DISCRETE GEOMETRIC AND PREDICTIVE NONLINEAR CONTROL

DISCRETE GEOMETRIC AND PREDICTIVE NONLINEAR CONTROL

By

CHRIS McCREADY, B.A.Sc

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Engineering

McMaster University

MASTER OF ENGINEERING (2002)
(Chemical)

McMaster University
Hamilton, Ontario

TITLE: Discrete Geometric and Predictive Nonlinear Control

AUTHOR: Chris McCready, B.A.Sc. (University of Waterloo)

SUPERVISOR: Professor J. F. MacGregor

NUMBER OF PAGES: x, 140

Abstract

The topic of study within includes the development and application of nonlinear control technologies on sampled systems. Discrete control structures are introduced that expand on existing differential geometric and predictive control methods. The differential geometric techniques are described from the error trajectory context, which are typically only derived for continuous application. The discrete error trajectory controllers introduced have one of two configurations. The first configuration requires satisfaction of the error trajectory objective at the next sampling interval through prediction of system behaviour over the controller sampling interval. This objective found limited success and it is observed that satisfaction of the error trajectory objective at discrete intervals does not generally result in the intended response. The second configuration minimizes the integrated distance from the error manifold defined by the error trajectory objective over the entire controller sampling interval. It is observed that this integrated error trajectory controller best emulates the intent of the continuous controller in the discrete domain. Techniques borrowed from predictive control are incorporated into the integrated error trajectory controller such as input move suppression and constraints to produce an optimal error trajectory controller, further improving performance.

The predictive control method introduced utilizes a transformation of the input space. The differentiating property of input transformation predictive control

(ITPC) from other methods is the prediction technique that is capable of estimating the future behaviour of nonlinear systems through elementary matrix operations similar to the dynamic matrix control (DMC) prediction technique. This is achieved by separation of the steady state and dynamic system properties and the introduction of an intermediate state prediction layer. This allows for the nonlinear prediction of system behaviour without the need to numerically integrate the system model.

Two example systems are used to demonstrate application of the discrete error trajectory and ITPC on nonlinear controllers. Performance for these control structures is compared to technologies accepted within the control community for a broad range for characteristics including, computation efficiency, design effort and other nonlinear performance criteria, with favourable results.

Acknowledgments

There are many individuals who contributed to the development of this report through inspiration, guidance and support. Special thanks are extended to Dr. John F. MacGregor who provided continual insights and direction and to the managers, process engineers and operators at the Petro-Canada Lubricants Centre in Mississauga, Ontario, Canada who provided the opportunity to test advanced control approaches on many physical systems. I would like to thank Robert Kuhne who first introduced me to model identification, process control and optimization.

Finally I'd like to thank my wife and daughter who provided me with patience and understanding and to my parents and family who continually supported my many diverse efforts no matter how strange.

Table of Contents

List of Figures

List of Tables

1.0 Introduction

Model based control (MBC) technologies are used to determine control policies for complex systems. Of the commercially available MBC packages, predictive methods are the most common. Model predictive control (MPC) encompasses a broad range of control algorithms that use an explicit system model to determine a control policy that minimizes an objective over a finite or infinite prediction horizon. Qin and Badgwell (2000) comment that model predictive control (MPC) has not yet penetrated into industries where process nonlinearities are strong and market demands require frequent changes in operating conditions. Recent advances in nonlinear (NL) MPC technologies are changing this perspective but many difficulties remain. The elegance and simplicity of linear control technologies make them the preferred choice for control applications. In a survey of commercially available model predictive control technology, Qin and Badgwell (1997) report over 2200 industrial MPC applications. They state that of these applications almost all use linear process models and are clustered in refinery and petrochemical processes. The reasons for the preference towards MPC based on linear system representations represents a good topic for debate but one distinction between linear and nonlinear MPC is clear. It is very difficult, if not impossible, to construct a model form that appropriately describes the behaviour of all feasible nonlinear systems yet many simple model forms exist for the linear case in both the continuous and discrete domains. Development of model identification procedures can not be completed without a generic model structure. Efficient nonlinear system identification remains a combination of art and science. Likewise, design of algorithms to determine a control policy for predictive controllers utilizing nonlinear system descriptions is highly dependent on the nonlinear characteristics. In general the jump from linear to nonlinear MPC requires a more in-depth understanding of the selected control technology, more knowledge of the plant behaviour, and more model identification effort. Why then is there interest in investigating and applying nonlinear control structures? The simple answer is necessity. Systems that contain severe nonlinearity or are operated over a wide range of conditions may require a control technology with capabilities greater than that provided by linear MPC to achieve the desired performance.

MPC however is just one of many MBC technologies. A sampling of those available for nonlinear control include, NL internal model control (NL-IMC) (Economou *et al.,* 1986), NL dynamic matrix control (NL-DMC) (Garcia, 1984), NL inferential control (Parrish and Brosilow, 1988), global linearizing control (GLC) (Kravaris and Kantor, 1990a and 1990b), feedback linearization (Isidori, 1991), generic model control (GMC) (Lee and Sullivan, 1988), NL programming (NLP) techniques (Jang *et al.*, 1987) and many ad hoc configurations. Some NL control techniques are computationally efficient, some handle constrained systems and some have restrictions on the system descriptions and/or dynamic properties. Algorithms well suited for the general non-square NL control problem with input constraints and non-minimum phase dynamics are typically computationally expensive and implementation on real systems often requires simplifications. Selection of the appropriate NL control technology therefore depends on the system to be controlled and to a certain extent the control hardware it is to be run on.

The area of interest in this study is the development and application of discrete nonlinear control technologies given a broad range of characteristics including, design effort, computational efficiency, constraint handling, objective function structure, servo capability and robustness at singular regions of operation. Two distinctly different control technologies are designed from existing differential geometric and predictive structures. The geometric control methods are presented using the error trajectory formulation. With respect to error trajectory descriptions in literature the methods described within have the unique characteristic of design for discrete implementation. A technique that is introduced here is the selection of input moves that are relevant over the controller execution interval rather than at the beginning of the interval as is the case with sampled implementation of the continuous error trajectory method. Two variants of this technique are described, the first selects inputs that satisfy the error trajectory objective at the next controller execution and the second minimizes the integrated distance from the error manifold defined by the error trajectory objective over the interval. Of these two methods the second was observed to best emulate the intent of the continuous controller. A drawback of the second method is its computational demand, requiring minimization of a nonlinear optimization function. It is shown that restructuring the error trajectory objective as an optimization problem allows for the inclusion of constraints, output weighting and input move suppression expanding the capability of the error trajectory controller.

The predictive control technique presented is built on the familiar quadratic dynamic matrix control (QDMC) structure (Garcia and Morshedi, 1986). An input transformation technique is introduced that separates the steady state and dynamic properties of the system. The result is the ability to estimate future behaviour of nonlinear systems through elementary linear matrix algebra and ultimately a computationally efficient nonlinear predictive controller.

The motivation behind development of the error trajectory and predictive methods stems from the author's need for a computationally efficient nonlinear control technology that is applicable in the discrete domain. Geometric methods in literature are primarily derived for continuous application yet implementation is typically performed on discrete computers with sampled state measurements. If the controller sampling time is fast compared to the system dynamics the discretization effects may be insignificant. In this study the problem of designing geometric control methods for application on systems where the controller execution frequency is significant is investigated. Nonlinear predictive control technologies are very powerful but in general are not computationally efficient. Real-time application often necessitates simplification techniques so that the set of future inputs can be found in time for implantation. The input transformation prediction method presented within eliminates the need for numerical integration of the system model, reducing the computational demands of the controller.

Organization of the following sections begins with a brief review of existing nonlinear MBC technologies in section 2.0. Section 3.0 describes discrete error trajectory control including an overview of literature references for discrete differential geometric control approaches. The error trajectory method is described from a predictive control perspective resulting in two discrete formulations of the objective. These methods are tested on an example system to assess their performance and comments are made on the suitability of the error trajectory objective in the discrete domain. Section 4.0 begins with a review of dynamic matrix control (DMC) followed by an explanation of the input transformation prediction control (ITPC) method. It is also shown that for the unconstrained case an explicit solution may exist for the ITPC controller. The performance of the ITPC controller is compared to the error trajectory controllers from section 3.0. Finally the ITPC method is used to optimize the operation of a continuously stirred tank reactor (CSTR) given a set of known disturbances. Section 5.0 contains a comparative discussion of discrete error trajectory, ITPC and other control methods accepted by the control community for a broad range of characteristics such as design effort, computational efficiency and performance. Sections 6.0 and 7.0 round out the main sections containing conclusions and suggested future research activities.

2.0 Nonlinear Control Methods

A comprehensive review of nonlinear control technology is beyond the scope of this report. Instead, a sampling of techniques that are representative of the range of common NL control methods are examined, focusing on those that are model based. Adaptive strategies and other control approaches such as sliding mode are not considered. The methods described include nonlinear model predictive control (NL-MPC), local linearization techniques and error trajectory methods. Their presentation is primarily directed at how the system model is utilized in the control law and their objectives.

The purpose of this section is to present a foundation of existing nonlinear control technologies that form the context in which the control structures developed later are derived. As well, a common nomenclature is established for use throughout.

2.1 Model Based Control Objectives

The control objective is a mathematical description of what the controller intends to do. The nonlinear control techniques described in this report have one of two basic objectives. The first objective involves designing an error trajectory function, E_f. The error trajectory is a *suitably well behaved* function of the output error (ε), and higher derivatives of this error.

$$E_f = \sum_{i=0}^{\alpha} \beta_i \cdot \frac{d^i \varepsilon}{dt^i} = 0 \qquad \text{where: } \varepsilon = y_d - y \qquad \textbf{(2-1)}$$

ε is the error, defined as the distance between the desired output or setpoint, y_d, and the actual or measured output y.

The error trajectory function defines a desired error manifold. The objective of the error trajectory controller is to select u such that the system is maintained on the error manifold. In general E_f is said to be *suitably well behaved* if the control law can be recovered implicitly or explicitly (McLellan *et al.*, 1990). In the SISO case ε, y_d and y are real valued parameters and $\beta \in \Re^{\alpha+1}$. β_i are pole placement tuning constants. α must be greater than or equal to the relative order, r, of the system. The concept of relative order is further explained in section 2.4, Continuous Error Trajectory Methods.

The second objective is an optimization problem, minimizing a quadratic function over a prediction horizon.

$$\min_{u_f} \quad J = \int_{t_o}^{t_o+t_p} e^T W_y e \cdot dt \qquad \text{where:} \quad \begin{aligned} e &= y_{Tr} - y \\ y_{Tr} &= f_{Tr}(y_d, y) \end{aligned} \qquad \textbf{(2-2)}$$

t_o is the present time and t_P is the length of the prediction horizon. $y_{Tr} \in \Re^{N_y}$ and is a trajectory towards y_d, where N_y is the number of outputs. W_y is a $\Re^{N_y \times N_y}$ weighting matrix that is usually diagonal and constant. The function f_{Tr} is the desired trajectory of the errors towards y_d. f_{Tr} and E_f are therefore similar in the sense that they both describe how the outputs dynamically track towards y_d. Of note, the error in (2-2) is defined slightly different than in (2-1). In (2-2) e is not really the error but rather the portion of error that is penalized in J. Penalizing the true error, ε, can be very aggressive and may lead to very large input moves unless the controller is detuned in some fashion (Qin and Badgwell, 2000).

In practice (2-2) is implemented in the discrete domain with some additional features such as input move suppression and constraints as in (2-3). Other common additions include state and output constraints, penalties for distances from input setpoints and economic costs. These are neglected in this analysis for simplicity.

$$\min_{u_f} \quad J = \frac{1}{2}\left\{e^T W_y e + \nabla u_f^T W_u \nabla u_f\right\} \tag{2-3}$$

$$\text{subject to: } u_{min} \le u_f \le u_{max}$$

$$|\nabla u_f| \le u_{roc}$$

In (2-3) e, y_{Tr}, y_d, and $y \in \Re^{N_y \cdot P}$ and ∇u_f, u_{min}, u_{max}, u_f and $u_{roc} \in \Re^{N_u \cdot C}$ where N_u is the number of inputs. ∇u_f are future input changes, u_{min} and u_{max} are minimum and maximum input constraints, u_{roc} are input rate of change limits and u_f are the absolute future input values. P is the number of discrete sampling intervals in the prediction horizon and C is the number of sampling intervals in the control horizon. In the discrete domain the weighting matrices $W_y \in \Re^{N_y \cdot P \times N_y \cdot P}$ and $W_u \in \Re^{N_u \cdot C \times N_u \cdot C}$. The weighting matrices most commonly contain positive values on the diagonal.

Even though the objectives in (2-1) and (2-3) are both rooted around a desired trajectory, y_d, they are fundamentally very different. The differences are most evident when considering controller tuning. A controller using the objective in (2-1) is tuned by defining an error manifold through pole placement that directs y towards y_d. The objective in (2-3) has much more flexible tuning capabilities containing not only a definition of the desired trajectory towards y_d but also weighting matrices W_y and W_u. The output weighting W_y, allows a relative priority to be given to specific outputs that require tighter control. Inclusion of move suppression W_u, imparts stabilizing qualities to the control performance especially in the presence of model mismatch and measurement noise or uncertainty (Gattu and Zaririou, 1991). Mathematically input weighting decreases ill conditioning, lowering the condition number of the model inversion (Qin and Badgwell, 1997). The weights eliminate large input moves that do not significantly reduce output errors. Excessively large input weights however will decrease controller performance; hence a balance must be achieved between robustness and aggressiveness. Zafiriou (1991) comments further on these tuning issues with respect to robustness.

The use of an optimization problem in objective (2-3) provides another beneficial property. The minimization of a set of penalties relaxes the requirement that input-output degrees of freedom are satisfied. The control law resulting from the objective in (2-1) is a set of N_y equations with N_u unknowns. Therefore a unique solution only exists for square systems where $N_y = N_u$. The control law resulting from (2-3) is a search over the feasible range of ∇u_f that minimizes *J*. The number of inputs and outputs may have an impact on the controller performance and controllability but they do not affect the existence of a solution. Furthermore, in the constrained case a non-square system utilizing the objective in (2-3) may be tuned in such a way that the controller is multi-structured depending on active constraints. As a simple example, a system with 1 output and 2 inputs can be tuned so that the output is primarily controlled with input 1. When input 1 becomes saturated the controller then uses input 2 to control the output.

2.2 Nonlinear Model Predictive Control

MPC refers to control technologies that use an explicit process model to estimate the future system behaviour over a prediction horizon. At a fundamental level nonlinearity can be introduced into MPC through the inclusion of input constraints, from actuator saturation and process limitations, or the use of nonlinear process models. In the context of this report NL-MPC denotes the use of nonlinear process models and MPC structures that use linear models will be called linear MPC even if they consider input constraints.

The control law for MPC controllers is defined as the solution to the objective in (2-3). At each control execution an optimal control policy is determined and the control moves for the present interval are implemented. This process is repeated at each subsequent control execution, $t_{[k+1]} = t_{[k]} + T_s$, where $t_{[k]}$ is the time of the kth control interval, $t_{[k+1]}$ is the time of the next control interval and T_s is the controller sample time. This recursive process results in moving horizon or receding horizon control as the prediction horizon shifts forward in time at each control execution.

y_{Tr} is typically defined as simply y_d, (2-4), or a first order error decay, (2-5). In general any appropriate function can be used. Qin and Badgwell (2000) describe other y_{Tr} design approaches used in commercial applications such as funnel or zone regions.

$$y_{Tr} = y_d \qquad \textbf{(2-4)}$$

$$y_{Tr} = \varepsilon + y_d \qquad \text{where: } \tau_c \cdot \frac{d\varepsilon}{dt} = \varepsilon \qquad \textbf{(2-5)}$$

In (2-5) τ_c is the desired closed loop time constant of the error decay. Use of (2-4) simplifies the definition of e so that $e = \varepsilon$.

Prediction of future output values requires the integration of the process model, (2-6), over the prediction horizon.

$$\frac{dx}{dt} = f(x, u_f)$$
$$y = h(x) \tag{2-6}$$

In (2-6) x is the system state vector on $\Re^{N_x}$. $u_f \in \Re^{N_u}$ and is typically assumed to be piecewise constant functions over the control horizon (Li and Biegler, 1989), sampled at future control executions. Normally all the states in (2-6) are not measured directly and the measurements of the system that are available contain noise. Filtering techniques such as the extended Kalman filter (EKF) are used to estimate the system states using the available process observations (Doucet, 1998). Gagnon and MacGregor (1991) and Gattu and Zafiriou (1995) provide design and tuning guidelines for Kalman filters for use in control applications. In particular these articles explain the value of adding meaningful nonstationary stochastic states to account for unknown disturbances and modeling errors. With the addition of these stochastic states and measurement noise the process model (2-6) becomes,

$$\frac{dx}{dt} = f_s(x, x^s, u_f)$$
$$\frac{dx^s}{dt} = \omega^s \qquad \text{where: } \omega^s \text{ and } v \text{ are white noise signals} \tag{2-7}$$
$$y = h_s(x, x^s) + v$$

The only requirement in augmenting the deterministic process model with these stochastic states is that they must be observable. This limits the number of these states to less than or equal to the number of independent process measurements (Gagnon and MacGregor, 1991) (Gattu and Zafiriou, 1995).

The process description (2-7) is a continuous relation however the objective function is discrete. This contradiction is rectified by outputting the results of the process model integration as a vector synchronized with the controller execution times.

For nonlinear systems the optimization problem in (2-3) is solved using nonlinear programming (NLP) techniques. The NLP uses iterative algorithms consisting of a sequence of linear programming (LP) or quadratic programming (QP) approximations, Newton's method or differential dynamic programming (Liao and Shoemaker, 1992). In some cases the solution to the optimization problem requires significant computer time. The controller however must find the solution in real-time, thus various commercial vendors have developed short-cut procedures to improve computational efficiency for complex or large systems (Qin and Badgwell, 1997) (Qin and Badgwell, 2000). The most common of these simplifying procedures is an assumption of linearity about the present operating point. The benefit of the linearity assumption is the nonlinear program is converted into a quadratic program (QP) that can be solved very efficiently (Li and Biegler, 1989) (Garcia, 1984). This topic is investigated further in sections 2.3, Local Linearization Techniques and 4.1, Dynamic Matrix Control.

Other simplification techniques are used to convert the system description into a more convenient form. An example of one of these techniques is orthogonal collocation on finite elements (OCFE). OCFE can be used to reduce the ordinary differential equations from (2-6) to algebraic equations (Sistu *et al.*, 1993).

The necessity for these simplifications is understood through a breakdown of a NLP algorithm. In general the NLP consists of an outer minimization procedure with an inner integration loop. The minimization procedure may consist of a gradient line search, sequential quadratic program (SQP) or other appropriate technique. The iterative nature of the search procedure may require hundreds or thousands of function evaluations (*J* from (2-3)) for even a seemingly simple problem. Each function evaluation requires the integration of (2-6), hence efficient numerical techniques are essential real-time implementation.

2.3 Local Linearization Techniques

Local linearization is a sub optimal extension of the NL-MPC structure. The nonlinear process model in (2-7) is only used to predict the system behaviour with no control action, (2-8).

$$\begin{aligned} \frac{dx}{dt} &= f_s\left(x, x_0^s, u_0\right) \\ \frac{dx^s}{dt} &= 0 \\ y_0 &= h_s\left(x, x_0^s\right) + v \end{aligned} \tag{2-8}$$

u_0 and x_0^s are the present input and stochastic state values and y_0 is the predicted outputs if no future input changes are implemented. A linear process model is derived from (2-8), normally centered around u_0. Garcia (1984) describes a linearizing procedure using (2-9).

$$\frac{dx_L}{dt} = F_0 \cdot x_L + G_0 \cdot u \qquad \text{where: } F_0 = \left.\frac{\partial f_x}{\partial x}\right|_{x_0, x_0^s, u_0}$$

$$y_L = H_0 \cdot x_L \qquad G_0 = \left.\frac{\partial f_x}{\partial u}\right|_{x_0, x_0^s, u_0} \qquad H_0 = \left.\frac{\partial h_y}{\partial x}\right|_{x_0, x_0^s} \tag{2-9}$$

x_0 is the present value of the states. x_L and y_L are the linearized change in future state and output values respectively. $F_0 \in \Re^{N_x \times N_x}$, $G_0 \in \Re^{N_x \times N_u}$ and $H_0 \in \Re^{N_y \times N_x}$ where N_x is the number of states. The 0 subscript on the linearized model parameters indicates that this model is time varying and is derived for the present operating point. The model is time varying in the sense that the model parameters are recalculated at every control execution. The continuous linear model is discretized to produce (2-10).

$$x_{L[k+1]} = \Phi_0 \cdot x_{L[k]} + \Gamma_0 \cdot u_{[k]}$$

$$y_{L[k]} = H_0 \cdot x_{L[k]} \tag{2-10}$$

Estimation of system behaviour beyond the k+1 time interval requires recursive iteration of (2-10).

Prediction of future outputs is accomplished by superposition of the integration of (2-8) and the linear model.

$$y = y_0 + y_L \quad \textbf{(2-11)}$$

This method is not rigorously valid for nonlinear systems but affords significant benefits. The advantage of this configuration is the integration of the nonlinear model is only performed once and the resulting optimization problem from (2-3) is reduced to a QP from a NLP. Furthermore the local linear method is guaranteed to converge to the optimal operating point so long as the method exhibits decent directions (Li and Biegler, 1989). This suggests convexity constraints on the full NL process model. Obviously the performance of this control method may deteriorate with severe nonlinearities. Quantification of nonlinearity and its effect on control performance is difficult and is a relatively new area of research. With respect to control, rules that defined the closed loop stability and performance of linear control, nonlinear control and local linear control technologies in the presence of various types of process nonlinearities would be immensely valuable. Development of nonlinear controllers typically requires greater effort than for the linear case. Guay *et al.* (1995) develop some useful measures of nonlinearity that attempt to assess when a process is sufficiently nonlinear to justify the development and use a nonlinear control law. This type of analysis may also be useful to assess if local linearization techniques are appropriate for a given nonlinear system.

An area that local linear techniques are limited in their ability is the prediction of system behaviour as the process is moved from one operating point to another. The predictive capability of the linear process model obviously will diminish the further the system moves from its linearization point in the prediction horizon.

Therefore prediction of future errors and constraint violations may be marginal during process transitions. Use of local linear techniques on transition applications such as optimal grade change trajectories or batch processes may not be appropriate depending on the type of process nonlinearity. This is not to say that local linear techniques will not perform adequately even in the presence of severe nonlinearity. Stack and Doyle (1997) suggest that linear controllers optimally control some classes of highly nonlinear systems, while other systems with much smaller measured nonlinearities require the use of a nonlinear control law. Therefore the linearization portion of this technique may not hinder the closed loop performance even if its predictive capability is compromised.

2.4 Continuous Error Trajectory Methods

In this report error trajectory methods refer to control technologies that utilize the E_f from (2-1) to design a control law. In nonlinear control literature there are a number of these, notably, reference systems synthesis (RSS) (Bartusiak *et al.*, 1989), global linearizing control (GLC) (Kravaris and Chung, 1987), generic model control (GMC) (Lee and Sullivan, 1988) and feeback linearization (Isidori, 1991) (Kravaris and Kantor, 1990a/b). McLellan *et al.* (1990) show that these techniques can be cast in the error trajectory form and that RSS, GLC, GMC and feedback linearization are special cases of the more general description in (2-1).

Error trajectory methods are most relevant for control affine systems, (2-12).

$$\frac{dx}{dt} = f_a\left(x^\sigma\right) + g_a\left(x^\sigma\right)\cdot u \qquad \text{where: } x^\sigma = \begin{bmatrix} x \\ x^s \end{bmatrix} \qquad \textbf{(2-12)}$$

$$\frac{dx^s}{dt} = \omega^s$$

$$y = h_s(x^\sigma)$$

The control law is designed using differential geometry and nonlinear inversion techniques along with a pole placement tuning procedure. This type of control design is often referred to as an output tracking problem which involves determining the control action required to force a system to follow a desired path or trajectory.

To develop the control law for a SISO system the objective function in (2-1) is rewritten as in (2-13) using Lie derivatives from differential geometry.

$$E_f = \sum_{i=0}^{\alpha} \beta_i \cdot \left(\frac{d^i y_d}{dt^i} - \left\{ L_f^i h_s(x^\sigma) + L_g L_f^{i-1} h_s(x^\sigma) \cdot u \right\} \right) = 0 \qquad \textbf{(2-13)}$$

$$\text{where: } \frac{d^i y}{dt^i} = L_f^i h_s(x^\sigma) + L_g L_f^{i-1} h_s(x^\sigma) \cdot u \qquad \textbf{(2-13a)}$$

$$L_f h_s(x^\sigma) = \frac{\partial h_s}{\partial x_1^\sigma} \cdot f_{a,1} + \frac{\partial h_s}{\partial x_2^\sigma} \cdot f_{a,2} + \ldots + \frac{\partial h_s}{\partial x_{N_\sigma}^\sigma} \cdot f_{a,N_\sigma} \qquad \textbf{(2-13b)}$$

$$L_f^i h_s(x^\sigma) = \frac{\partial L_f^{i-1} h_s}{\partial x_1^\sigma} \cdot f_{a,1} + \frac{\partial L_f^{i-1} h_s}{\partial x_2^\sigma} \cdot f_{a,2} + \ldots + \frac{\partial L_f^{i-1} h_s}{\partial x_{N_\sigma}^\sigma} \cdot f_{a,N_\sigma} \qquad \textbf{(2-13c)}$$

$$L_g L_f^i h_s(x^\sigma) = \frac{\partial L_f^{i-1} h_s}{\partial x_1^\sigma} \cdot g_{a,1} + \frac{\partial L_f^{i-1} h_s}{\partial x_2^\sigma} \cdot g_{a,2} + \ldots + \frac{\partial L_f^{i-1} h_s}{\partial x_{N_\sigma}^\sigma} \cdot g_{a,N_\sigma} \qquad \textbf{(2-13d)}$$

The relative order of the system is interpreted as the number of times y must be integrated to result in an expression containing the input u. Therefore, $L_g L_f^{i-1} h_s(x^\sigma) = 0$ for $i < r$, where r is the relative order of the system. For a more complete description of Lie derivatives and relative order refer to (Kravaris and Chung, 1987) (Kravaris and Kantor, 1990a).

Integral action can be included in error trajectory controller by setting $\alpha = r+1$ (McLellan *et al.*, 1990). With the system description in (2-12) however integral action will be provided by a state observer, such as a Kalman filter, as a result of including stochastic states in the process model. Stochastic states remove the steady state offset between the model predictions and measurements (McAuley and MacGregor, 1993). Using $\alpha = r$ and solving for u results in the continuous error trajectory control law (2-14).

$$u = \frac{\left[\sum_{i=0}^{\alpha} \beta_i \cdot \frac{d^i y_d}{dt^i}\right] - \left[\sum_{i=0}^{\alpha} \beta_i \cdot L_f^i h_s(x^\sigma)\right]}{\beta_r \cdot L_g L_f^{\alpha-1} h_s(x^\sigma)} \tag{2-14}$$

Kravaris and Soroush (1990) provide an extension of a nonlinear input/output linearization method, which bears similarity to the error trajectory method shown here, for multivariable systems. Notably the multivariable controller design procedure described will only yield an explicit control law for square systems. Essentially the error trajectory method results in N_Y equations with N_u unknowns. A unique solution to the control law therefore requires $N_u = N_Y$. The error trajectory equation for the jth output is shown in (2-15). The multivariable extension to (2-13) simply accounts for the influence of all inputs on the derivatives of each output.

$$E_{f,j} = \sum_{i=0}^{\alpha} \beta_{i,j} \cdot \left(\frac{d^i y_{d,j}}{dt^i} - \left\{ L_{f,j}^i h_{s,j}(x^\sigma) + \sum_{n=1}^{N_u} L_{g,n} L_{f,j}^{i-1} h_{s,j}(x^\sigma) \cdot u_n \right\} \right) = 0 \tag{2-15}$$

The model inversion used to design the error trajectory control law potentially presents problems. Since (2-12) is assumed to be nonlinear there may exist singular regions such that the denominator in (2-14) is zero. In these singular regions the system is no longer controllable as input changes have no affect on the system. A secondary problem is the inability to control systems possessing nonminimum phase dynamics. Nonminimum phase systems are noninvertible but an approximate inverse can be developed where the undesirable dynamics are removed to provide an approximate solution (McLellan *et al.*, 1990).

3.0 Discrete Error Trajectory Control

Differential geometric control approaches such as the error trajectory method described are naturally derived in the continuous domain. Application of this technology is typically implemented on digital computers at discrete sampling intervals. Digital control problems designed on the basis of linear continuous time models are solved with sampled dynamics interpreted as a discrete time model where the linear structure is preserved under sampling (Monaco and Normand-Cyrot, 1997). Nonlinear continuous systems however are not easily described in the discrete domain and structural properties are generally not preserved under sampling. Many authors (Grizzle and Kokotovic, 1988) (Bequette, 1990) (Ferreira and Agrawal, 2000) have noted that sampling can destroy the performance of geometric control technologies, yet few have prescribed solutions.

An area of research that has received attention is discrete control of feedback linearizable systems. The convenient property of these systems is the existence of an exact transformation into a linear form. Details on continuous feedback linearization are found in Isidori (1991) and Kravaris and Kantor (1990a and 1990b). McLellan *et al.* (1990) describe feedback linearization in the context of the error trajectory approach. The specifics of feedback linearization are unimportant in the context of this study but the end result is significant with respect to sampled systems. The system is linearized through state feedback and coordinate transformation to produce a system description of the form in (3-1) where z and υ are the transformed system states and inputs.

$$\frac{dz}{dt} = A_t \cdot z + B_t \cdot \upsilon \tag{3-1}$$

Linear system theory can then be used to control the transformed system, reducing a potentially difficult nonlinear control problem to a simple linear one. Arapostathis *et al.* (1989) describe the effect of sampling on the transformed system and summarize conditions for the existence of systems that are sampled feedback linearizable. Ferreira and Agrawal (2000) investigate discrete control of feedback linearizable systems through the design of an optimal planning strategy. The difficulty experienced with these techniques is the requirement that υ remain constant between sampling intervals. The transformed input can contain a combination of system inputs and continuous state variables invalidating the zero order hold condition. Ferreira and Agrawal quantify this error and point out that the effects of sampling are minor if the system transformation is weakly dependent on the states or if the states don't significantly change over the sampling interval. This indicates that if the sampling frequency is fast compared to the system dynamics the error for the sampled feedback linearized system representation is small. Grizzle and Kokotovic (1988) show that feedback linearizability is not generally preserved under sampling but describe a method of recovering a feedback linearizable discrete time model from a continuous one through an example. Jakubczyk and Sontag (1990) survey known results for controllability of nonlinear systems and present new characterizations using differential geometric ideas.

In this report the area of interest is the applicability of the continuous error trajectory objective on sampled systems and alternative error trajectory structures that are better suited for discrete implementation. Examples of discrete geometric control approaches similar to error trajectory appear in Spong *et al.* (1986) and Ferreira and Agrawal (2000). These applications include a reformulation of the error trajectory objective as an optimization problem over a limited prediction horizon. This suggests that at a minimum there is an intuitive awareness that the objective of error trajectory and other related nonlinear geometric continuous control approaches are not well suited for sampled systems. Direct application of the continuous error trajectory control law (2-14) on a sampled system can result in poor performance. The effects of discretization can be shown through Taylor series expansion of the sampled control law. Given the control law for the determination of u at the present time (3-2), the difference between the continuous and sampled implementation are the sum of the first and higher order terms in the Taylor series expansion, (3-3).

$$u_{[k]} = f_{et}\left(x^{\sigma}_{[k]}\right) = \frac{\left[\sum_{i=0}^{\alpha} \beta_i \cdot \left.\frac{d^i y_d}{dt^i}\right|_{[k]}\right] - \left[\sum_{i=0}^{\alpha} \beta_i \cdot L_f^i h_s\left(x^{\sigma}_{[k]}\right)\right]}{\beta_r \cdot L_g L_f^{\alpha-1} h_s\left(x^{\sigma}_{[k]}\right)} \tag{3-2}$$

$$\partial u_{[k]} = \frac{\partial\left\{f_{et}\left(x^{\sigma}_{[k]}\right)\right\}}{\partial x^{\sigma}} \cdot \left(x^{\sigma}(t) - x^{\sigma}_{[k]}\right) + \frac{\partial^2\left\{f_{et}\left(x^{\sigma}_{[k]}\right)\right\}}{\partial x^{\sigma 2}} \cdot \frac{\left(x^{\sigma}(t) - x^{\sigma}_{[k]}\right)^2}{2!} + \ldots \tag{3-3}$$

$\partial u_{[k]}$ is the difference between the continuous input u from (2-14) and the input resulting from the sampled control law $u_{[k]}$, for the set of continuous states x^{σ}. (3-3) indicates that as the states move from their value at the sampling time k $\partial u_{[k]}$

grows. Therefore the effects of discretization are magnified if the controller execution frequency is significant with respect to the system dynamics. It is also important to note that this effect is not a result of the system nonlinearity but rather system dynamics. Nonlinearity can magnify the problem however especially if the process gain, represented by $L_g L_f^{i-1} h_s(x_{[k]}^c)$, changes dramatically in magnitude or sign. Direct application of the continuous error trajectory objective in the discrete domain clearly may not perform as expected. There is value in the method and with some reformulation an error trajectory controller can be designed for the discrete domain.

In the discrete domain a controller must select a set of inputs to be implemented for the interval $t = t_{[k]}$ to $t_{[k]} + T_s$. Sampled implementation of the continuous error trajectory control law however selects the set of inputs that satisfy the objective at the time of the controller execution, given the present state values. Taylor series expansion of the sampled error trajectory control law indicates that the input values implemented at each sampling interval become less valid as the continuous states change. The idea that is introduced in this report is that discrete implementation of error trajectory control should be approached by attempting to select a set of inputs that are relevant over the sampled interval rather than at the beginning. Specifically two discrete error trajectory controllers are described, the first selects the set of inputs that satisfy the error trajectory objective at the end of the present execution interval and the second selects inputs to minimize the distance from the error manifold over the entire sampling interval. This involves prediction of system changes to the next execution time. Given that the control law is built on a system model, utilization of this model for predictive purposes is a natural extension. In this respect a discrete error trajectory controller can be designed as a predictive controller. In the next section a discrete error trajectory controller is designed in the context of MPC. This is useful as the derivation helps to bridge the gap between error trajectory and predictive control structures and sets up a strong basis for comparison of the techniques.

3.1 Error Trajectory Predictive Control

A SISO NL predictive controller is designed with prediction and control horizons of one sampling interval, $P = C = 1$, based on the error trajectory objective (2-1). The outputs for this predictive controller are the error and the first r derivatives of the error, where r is the relative order of the system. The predictive objective is to minimize the quadratic value of the error trajectory error function, E_f from (2-1) at the next controller execution as in (3-4). Noticeably input constraints, rate of change limits and input move suppression are not included in the objective.

$$\min_{u} J = \left(E_{f[k+1]}\right)^2 \tag{3-4}$$

The goal in (3-4) is simply to drive the absolute value of the error function to a minimum. For controllable systems the unconstrained solution exists at $J = 0$. Setting J to 0 and expanding $E_{f[k+1]}$ produces;

$$\sum_{i=0}^{r} \beta_i \cdot \left(\left.\frac{d^i y_{Tr}}{dt^i}\right|_{[k+1]} - \left.\frac{d^i y}{dt^i}\right|_{[k+1]} \right) = 0 \qquad \text{where: } y_{Tr} = y_d \tag{3-5}$$

Rewriting (3-5) using Lie algebra results in (3-6).

$$\sum_{i=0}^{r}\left\{\beta_i \cdot \left.\frac{d^i y_d}{dt^i}\right|_{[k+1]} - \beta_i \cdot L_f^i h_s\left(x_{[k+1]}^g\right)\right\} - \beta_r \cdot L_g L_f^{r-1} h_s\left(x_{[k+1]}^g\right) \cdot u_{[k]} = 0 \qquad \text{(3-6)}$$

(3-6) is identical to (2-13) with the exception that the desired output trajectory and the Lie terms are the projected values at the next sampling interval. This outcome is intuitive and really not the significant product of the derivation. More importantly the short derivation displays the error trajectory objective in the predictive context and can be used to highlight the significant differences between discrete error trajectory control and traditional predictive control approaches.

It has long been understood that in predictive control long predictive horizons are favoured over short ones. Infinite prediction horizons have been studied for decades and are found to have convenient properties, such as guaranteed closed loop stability for linear quadratic Gaussian (LQG) optimal control under general assumptions (Morari, 1993). Finite horizons are used in industrial predictive control technologies to simplify the control law computation and allow for the inclusion of constraints. A general rule of thumb however is to select the predictive horizon long enough that it approaches the settling time of the system. This approach captures the system's dominant dynamic properties and provides a means of controlling nonminimum phase transients. The proposed discrete error trajectory controller only has a predictive horizon of 1 therefore it has limited stability guarantees and is not appropriate for nonminimum phase systems. This is true of the continuous controller as well.

A second area the predictive error trajectory controller differs from traditional predictive control is tuning. The tuning constants β_i can be used to adjust the aggressiveness and dynamic properties of the closed loop response but they are not very useful in handling ill conditioned regions of operation. This is particularly important for NL systems as the system conditioning varies over points of operation. Predictive control effectively handles ill conditioning through the use of input move suppression via penalties on input changes.

Comparison of the predictive controller (3-6) to the sampled version of the equivalent continuous error trajectory controller provides further insight into problems associated with discretization. For instance one could make the unsophisticated and erroneous assumption that the future value of the Lie terms are best estimated as the present value.

$$\left.\frac{d^i y_d}{dt^i}\right|_{[k+1]} = \left.\frac{d^i y_d}{dt^i}\right|_{[k]}$$

$$L_f^i h_s\left(x_{[k+1]}^p\right) = L_f^i h_s\left(x_{[k]}^p\right) \tag{3-7}$$

$$L_g L_f^i h_s\left(x_{[k+1]}^p\right) = L_g L_f^i h_s\left(x_{[k]}^p\right)$$

Substituting these assumptions into (3-6), and solving for $u_{[k]}$ results in the control law, (3-8).

$$u_{[k]} = \frac{\left[\sum_{i=0}^{r} \beta_i \cdot \left.\frac{d^i y_d}{dt^i}\right|_{[k]}\right] - \left[\sum_{i=0}^{r} \beta_i \cdot L_f^i h_s\left(x_{[k]}^p\right)\right]}{\beta_r \cdot L_g L_f^{i-1} h_s\left(x_{[k]}^p\right)} \tag{3-8}$$

Note that (3-8) is identical to (3-2). This is because (3-8) is the actual control law executed by error trajectory controllers when implemented in the discrete domain. Practitioners of predictive control would no doubt have reservations of using (3-8), especially if sample times are significant compared to the system dynamics. The ZOH assumption nullifies the predictive ability of the controller.

The predictive error trajectory objective has a subtle difference from the predictive objective in (2-3). In (2-3) each of the terms in the error vector contributes to the objective function, regardless of sign, because each term is squared ($e^T W_y e$). In (3-4) only the weighted sum of the error is penalized. Therefore terms with positive error can be compensated by other terms with offsetting negative errors. These two error weightings enforce different closed loop properties. For instance using (3-4) a SISO system with relative order 1 accepts errors in the output with the condition that the error derivative is of the opposite sign, (3-9).

$$\varepsilon = -\frac{\beta_1}{\beta_0} \cdot \frac{d\varepsilon}{dt} \tag{3-9}$$

For the same system error weighting of the form (2-3) penalizes both the error and its first derivative no matter what the sign of each.

Of note the discrete error trajectory as presented only requires that the objective be satisfied at the sampling intervals. This is a common property of discrete control theory. For instance dynamic matrix control (DMC) only minimizes the error at discrete points along the predictive horizon. This qualification does however signify that the performance of the discrete error trajectory controller will not match the continuous equivalent. In fact it will be shown that the satisfaction of the error trajectory objective at discrete sampling intervals may lead to unacceptable performance.

One of the benefits of the continuous error trajectory control method is that often the control law consists of a set of algebraic equations. Therefore the controller can be implemented without sophisticated numerical integration or optimization routines. Unfortunately the solution to the predictive discrete error trajectory objective (3-6) requires integration of the system model. If computational efficiency is a concern intelligent assumptions can be used to simplify the problem. The next section includes some simplification techniques but any reasonable method of estimating the Lie algebraic terms for the next sampling interval can be utilized.

3.2 Sampled Error Trajectory Objectives

The results from the previous section are used to design a sampled error trajectory controller. Specifically there are three configurations presented. The first of these is referred to as the exact discrete error trajectory controller. The term exact is used to indicate that no simplification techniques are used to satisfy the sampled objective. In order to preserve the computational benefits of error trajectory control simplification techniques are employed for the remaining sampled controllers. The simplification techniques involve explicit methods of estimating the future system response to again produce an algebraic control law.

3.2.1 Exact Discrete Error Trajectory Objective

The basic structure of the sampled controller has already been derived in the Error Trajectory Predictive Control section. The error function, (3-6), selects the input at the present sampling interval that satisfies the objective at the next sampling interval.

The problem that arises is finding the solution to (3-6) since integration of the system model is required over the sampling interval. Assuming there is not an algebraic solution to the integration, the inputs that satisfy (3-6) can be found through a minimization algorithm using the predictive objective (3-4).

3.2.2 Euler Discrete Error Trajectory Objective

The first simplification technique applies the Euler method to estimate future differentials (3-10). Substituting (3-10) into the discrete objective function results in (3-11).

$$\left.\frac{d^i y}{dt^i}\right|_{[k+1]} = \left.\frac{d^i y}{dt^i}\right|_{[k]} + T_s \cdot \left.\frac{d^{i+1} y}{dt^{i+1}}\right|_{[k]} \qquad \textbf{(3-10)}$$

$$\sum_{i=0}^{r} \beta_i \cdot \left[\left.\frac{d^i y_d}{dt^i}\right|_{[k+1]} - \left(\left.\frac{d^i y}{dt^i}\right|_{[k]} + T_s \cdot \left.\frac{d^{i+1} y}{dt^{i+1}}\right|_{[k]} \right) \right] = 0 \qquad \textbf{(3-11)}$$

This method produces a more cumbersome objective and requires determination of $r+1$ differentials but still results in algebraic equations.

In (3-10) the Euler method is used to estimate the future differential terms. In some cases it may be more beneficial to estimate future state values and use the estimated states to predict the future differentials. This procedure preserves portions of the nonlinear system structure that may be lost by simply predicting the differentials.

3.2.3 Discrete Model Error Trajectory Control

The second simplification technique is really an extension of the exact error trajectory controller but a discrete model is used to provide an explicit control law. This technique therefore requires the development of a suitable discrete system model, (3-12).

$$\begin{aligned} x_{[k+1]} &= F_s\left(x^s_{[k]}, u_{[k]}\right) \\ x^s_{[k+1]} &= x^s_{[k]} + \omega^s_{[k+1]} \\ y_{[k]} &= H_s\left(x^s_{[k]}\right) + v_{[k]} \end{aligned} \tag{3-12}$$

Using the discrete model the Lie algebra terms can be estimated for the next sampling interval. Letting Ω_f and Ω_g represent the functions that define the Lie algebraic terms (3-13), the discrete control law is shown in (3-14).

$$\begin{aligned} \Omega_f\left(x^s_{[k]}, u_{[k]}\right) &= L^i_f h_s\left(x^s_{[k+1]}\right) \\ \Omega_g\left(x^s_{[k]}, u_{[k]}\right) &= L_g L^i_f h_s\left(x^s_{[k+1]}\right) \end{aligned} \tag{3-13}$$

$$\sum_{i=0}^{r}\left\{\beta_i \cdot \left.\frac{d^i y_d}{dt^i}\right|_{[k+1]} - \beta_i \cdot \Omega_f\left(x^s_{[k]}, u_{[k]}\right)\right\} - \beta_r \cdot \Omega_g\left(x^s_{[k]}, u_{[k]}\right) \cdot u_{[k]} = 0 \tag{3-14}$$

A complication that may present itself using the discrete model is the possible presence of squared or nonlinear input terms. This can happen if $\Omega_g\left(x^s_{[k]}, u_{[k]}\right)$ contains $u_{[k]}$ in any form or $\Omega_f\left(x^s_{[k]}, u_{[k]}\right)$ contains terms nonlinear in $u_{[k]}$. These problematic terms make determination of an explicit algebraic equation for $u_{[k]}$ difficult.

The effectiveness of these sampled error trajectory techniques are displayed through an example. The system for the sample problem is shown in (3-15).

$$\frac{dx}{dt} = F \cdot x + G(x) \cdot u \qquad \text{where: } F = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -2 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 \\ 0 & 0 & 0 & -\frac{1}{5} \end{bmatrix} \cdot \quad G(x) = \begin{bmatrix} 0 & 0 \\ 2 & 0 \\ 0 & \frac{x_1}{6} \\ \frac{2}{5} & 0 \end{bmatrix} \tag{3-15}$$

$$y = H \cdot x \qquad H = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For simplicity no stochastic states or noise terms are included in the system and all states are perfectly observed. Proper examination of noise and disturbance rejection necessitates the use and analysis of filtering techniques such as the extended Kalman filter.

The example system includes two nonlinear and two linear dynamic input state relationships. Nonlinearity enters into the system through the term $x_1/6$ in $G(x)$. Input step responses are shown in Figure 1. Figure 1 a) shows the linear second order under damped relationship between u_1 and x_1. Figure 1 b) displays the effect of u_1 on x_3. Included are three step changes, showing the different responses for varying values of u_2 (0, 1 and –1). Figure 1 c) displays the linear first order relationship between u_1 and x_4. Similar to Figure 1 b), Figure 1 d) includes three step responses displaying the relationship between u_2 and x_3 for varying values of x_1 (0, 1 and -1). The system nonlinearity is severe containing not only dramatic changes in process gain magnitudes but also sign changes, including singular points. The nonlinear steady state input-output and input-state maps are summarized in (3-16).

$$x_1 = u_1 \qquad x_3 = \frac{1}{2} \cdot x_1 \cdot u_2 \qquad \text{(3-16)}$$

$$\dot{x}_4 = 2 \cdot u_1$$

$$\text{where: } y_1 = x_1 + x_3 = u_1 + \frac{1}{2} \cdot u_1 \cdot u_2$$

$$y_2 = x_4 = 2 \cdot u_1$$

The system responses in Figure 1 and (3-16) effectively convey the steady state nonlinearities but do not clearly present the nonlinear dynamic properties of the system. A test in superposition is used to display dynamic nonlinearities. Figure 2 includes step responses in x_3 for the input changes, $u_1^0 = 1$ to $u_1^f = -1$ and $u_2^0 = 1$ to $u_2^f = 2$, where u_i^0 is the initial input value and u_i^f is the final input value. Both input changes are implemented at $t = 0$. Figure 2 a) shows the system response for simultaneous execution of these input changes. Figure 2 b) displays the response of x_3 for the same input changes in Figure 2 a) carried out independently. A characteristic of systems that exhibit linear dynamics is the validity of superposition. In this case the superposition of the independent step responses does not match the system response of the simultaneous input changes, as shown in Figure 2 c). In particular the actual system response includes nonminimum phase dynamics while the superposition response does not.

Servo control of the system with linear proportional integral (PI) controllers is shown in Figure 3. The discrete PI control algorithm is based on a velocity form of the continuous PI equivalent (3-17).

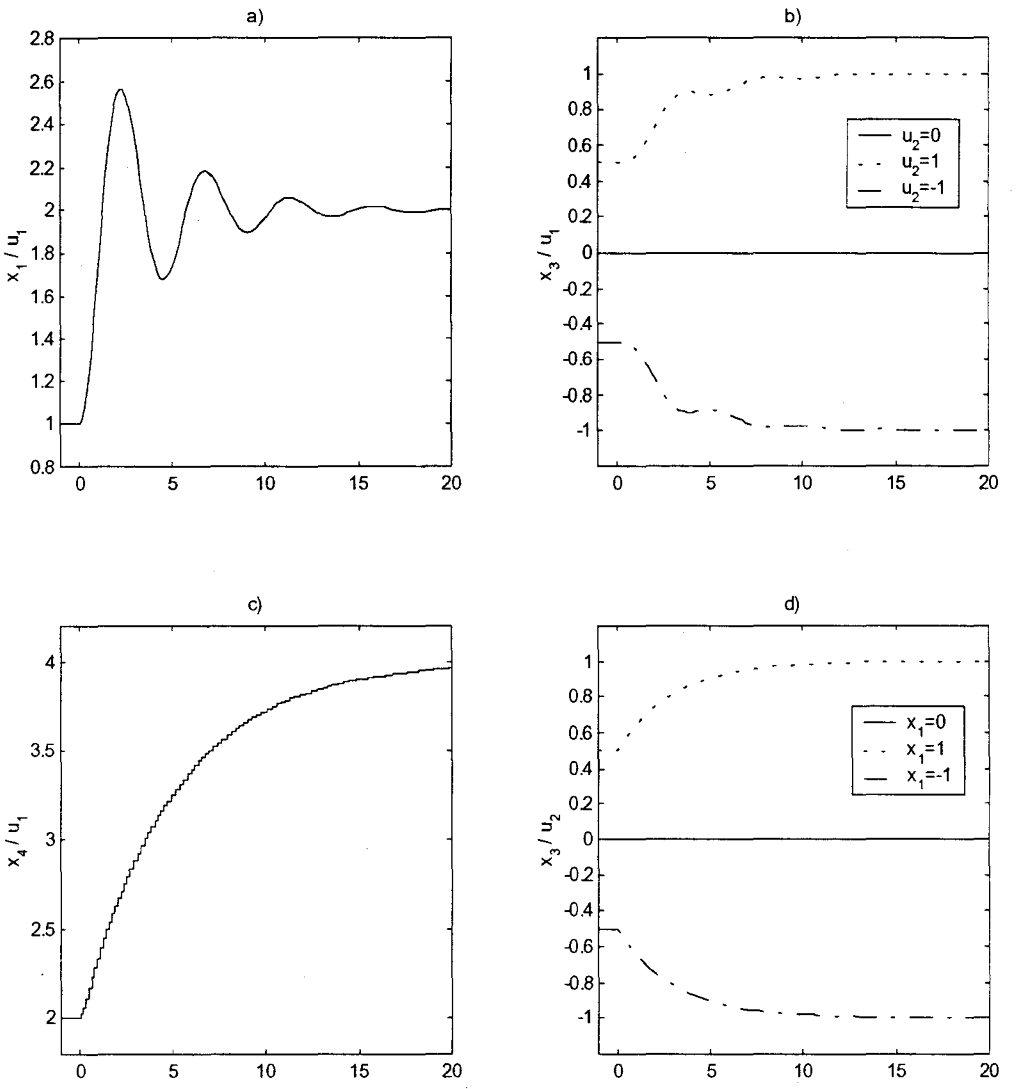

Figure 1, Open loop unit step responses for the sample system (3-15). For each step response the input is changed from 1 to 2 at t = 0.

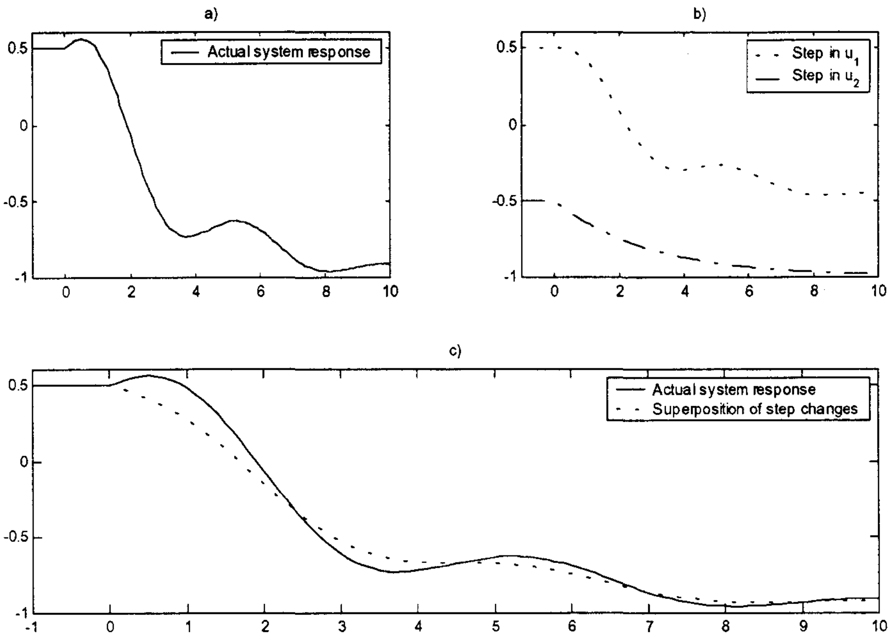

Figure 2, Dynamic response of x_3 in the sample system (3-15) to step changes of $u_1 = 1$ to -1 and $u_2 = 1$ to 2. a) simultaneous input step change (actual system response), b) independent step changes c) comparison of actual system response and superposition of independent input step changes.

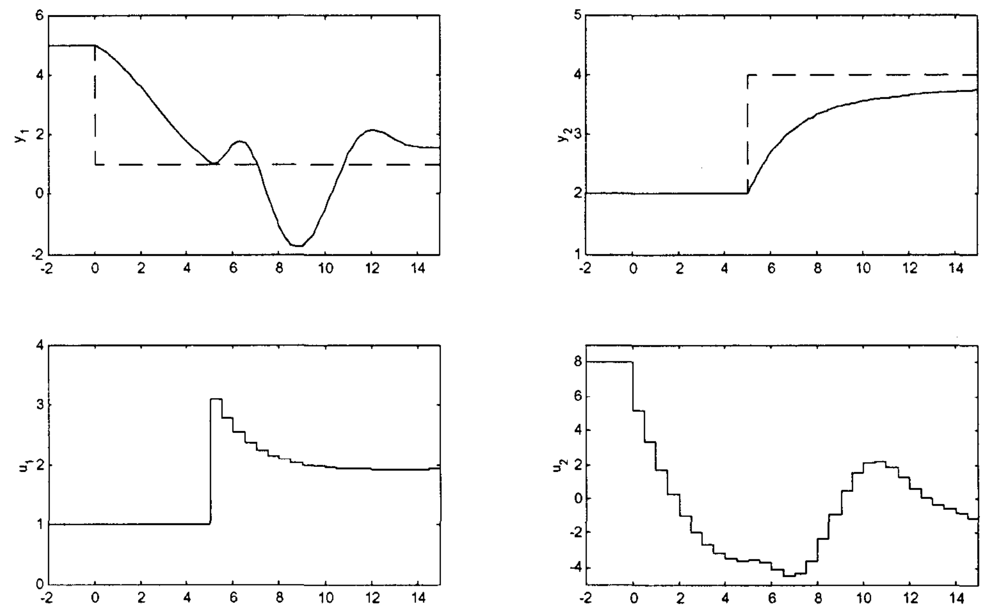

Figure 3, Setpoint tracking of the PI controller (3-17) on the example system with a sample time of $T_s = 0.5$.

$$u_{[k]} = u_{[k-1]} + K_c \cdot \left(\nabla \varepsilon_{[k]} + \frac{T_s}{\tau_I} \cdot \varepsilon_{[k]} \right) \qquad \text{where: } \nabla \varepsilon_{[k]} = \varepsilon_{[k]} - \varepsilon_{[k-1]} \tag{3-17}$$

The PI controllers are configured so that in the first loop y_1 is maintained by u_2 and in the second loop y_2 is maintained by u_1. The tuning of the first loop consists of $K_c = 10$ and $\tau_I = 0.2$ and for the second $K_c = 1$ and $\tau_I = 0.2$. The performance of the simple PI control scheme is not desirable. This example does however provide a nice baseline from which to compare the performance of nonlinear MBC control technologies such as error trajectory.

The Lie derivative terms for (3-15) are;

$$\begin{aligned} L_f h_1(x) &= x_2 - \frac{x_3}{3} & L_f h_2(x) &= -\frac{x_4}{5} \\ L_g h_1(x) \cdot u &= \frac{x_1}{6} \cdot u_2 & L_g h_2(x) \cdot u &= \frac{2}{5} \cdot u_1 \end{aligned} \tag{3-18}$$

Referring back to the definition of relative order, the system is relative order 1 since $L_g h_1(x)$ and $L_g h_2(x)$ are both nonzero. Substituting the terms in (3-18) into the MIMO error trajectory objective, (2-15) and solving for the inputs, produces the continuous error trajectory control law (3-19).

$$\begin{aligned} u_1 &= \frac{\beta_{0,2} \cdot (y_{d,2} - y_2) + \beta_{1,2} \cdot \left(\frac{dy_{d,2}}{dt} - \frac{x_4}{5} \right)}{\beta_{1,2} \cdot \frac{2}{5}} \\ u_2 &= \frac{\beta_{0,1} \cdot (y_{d,1} - y_1) + \beta_{1,1} \cdot \left(\frac{dy_{d,1}}{dt} - x_2 + \frac{x_3}{3} \right)}{\beta_{1,1} \cdot \frac{x_1}{6}} \end{aligned} \tag{3-19}$$

The exact discrete error trajectory controller is designed using (3-6). For the example system the error trajectory functions are shown in (3-20).

$$E_{f,1}\Big|_{[k]} = \beta_{0,1}\cdot\left(y_{d,1[k+1]} - y_{1[k+1]}\right) + \beta_{1,1}\cdot\left(\frac{dy_{d,1}}{dt}\bigg|_{[k+1]} - x_{2[k+1]} + \frac{x_{3[k+1]}}{3} - \frac{x_{1[k+1]}}{6}\cdot u_{2[k]}\right) = 0$$

$$E_{f,2}\Big|_{[k]} = \beta_{0,2}\cdot\left(y_{d,2[k+1]} - y_{2[k+1]}\right) + \beta_{1,2}\cdot\left(\frac{dy_{d,1}}{dt}\bigg|_{[k+1]} + \frac{x_{4[k+1]}}{5} - \frac{2}{5}\cdot u_{1[k]}\right) = 0 \qquad \text{(3-20)}$$

The solution to (3-20) requires integration of the system model over the interval $t_{[k]} \rightarrow t_{[k]} + T_s$ to estimate the state values at the next control execution. In general an algebraic solution to the integration will not exist but the solution can be found by formulating (3-20) as an optimization problem. A suitable formulation is shown in (3-21).

$$\min_{u_{[k]}} J = \left(E_{f,1}\Big|_{[k+1]}\right)^2 + \left(E_{f,2}\Big|_{[k+1]}\right)^2 \qquad \text{(3-21)}$$

The unconstrained solution is found at $J = 0$ through the use of a minimization algorithm. The set of inputs that minimize (3-21) are referred to as the exact discrete error trajectory control policy.

The discrete error trajectory controller using the Euler simplification method is designed similar to the continuous controller but includes additional terms for predictive ability as shown in (3-22). Replacing the differentials with Lie algebraic terms and grouping like terms results in the Euler objective, (3-23).

$$\beta_{0,j}\cdot\left[y_{d,j}\Big|_{[k+1]} - \left(y_j\Big|_{[k]} + T_s\cdot\frac{dy_j}{dt}\bigg|_{[k]}\right)\right] + \beta_{1,j}\cdot\left[\frac{dy_{d,j}}{dt}\bigg|_{[k+1]} - \left(\frac{dy_j}{dt}\bigg|_{[k]} + T_s\cdot\frac{d^2y_j}{dt^2}\bigg|_{[k]}\right)\right] = 0 \qquad \text{(3-22)}$$

$$\beta_{0,j}\cdot\left(y_{d,j[k+1]}-y_{j[k]}\right)+\beta_{1,j}\left.\frac{dy_{d,2}}{dt}\right|_{[k+1]}-\left(T_s\cdot\beta_{0,j}+\beta_{1,j}\right)\cdot\left[L_f h_j\left(x_{[k]}\right)+L_f h_j\left(x_{[k]}\right)\cdot u_{[k]}\right]$$
$$-T_s\cdot\beta_{1,j}\cdot\left[L_f^2 h_j\left(x_{[k]}\right)+L_g L_f h_j\left(x_{[k]}\right)\cdot u_{[k]}\right]=0 \tag{3-23}$$

The Euler control law for the error trajectory controller is found by substituting the terms in (3-18) along with the second order Lie derivatives, (3-24), into (3-23) and solving for the inputs to produce (3-25).

$$L_f^2 h_1(x)=-2\cdot x_1-\frac{1}{2}\cdot x_2+\frac{1}{9}\cdot x_3 \qquad L_f^2 h_2(x)=\frac{x_4}{25}$$
$$L_g\, L_f h_1(x)\cdot u=2\cdot u_1-\frac{x_1}{18}\cdot u_2 \qquad L_g\, L_f h_2(x)\cdot u=-\frac{2}{25}\cdot u_1 \tag{3-24}$$

$$u_1=\frac{\beta_{0,2}\cdot\left(y_{d,2}-y_2\right)+\beta_{1,2}\cdot\frac{dy_{d,2}}{dt}+\left(T_s\cdot\beta_{0,2}+\beta_{1,2}\right)\cdot\frac{x_4}{5}-T_s\cdot\beta_{1,2}\cdot\frac{x_4}{25}}{\left(T_s\cdot\beta_{0,2}+\beta_{1,2}\right)\cdot\frac{2}{5}-T_s\cdot\beta_{1,2}\cdot\frac{2}{25}}$$
$$u_2=\frac{\beta_{0,1}\cdot\left(y_{d,1}-y_1\right)+\beta_{1,1}\cdot\frac{dy_{d,1}}{dt}-\left(T_s\cdot\beta_{0,1}+\beta_{1,1}\right)\cdot\left(x_2-\frac{x_3}{3}\right)-T_s\cdot\beta_{1,1}\cdot\left(2\cdot\left(u_1-x_1\right)-\frac{x_2}{2}+\frac{x_3}{9}\right)}{\left(T_s\cdot\beta_{0,1}+\beta_{1,1}\right)\cdot\frac{x_1}{6}-T_s\cdot\beta_{1,1}\cdot\frac{x_1}{18}} \tag{3-25}$$

A discrete representation of (3-15) is required to design the second simplified discrete error trajectory controller. It is difficult to design a discrete system model that is general for any sample time, T_s, therefore for simplicity a sample time of 0.5 is assumed. The discrete model is shown in (3-26).

$$\hat{x}_{[k+1]} = \Phi \cdot x_{[k]} + \Gamma(x) \cdot u_{[k]} \qquad \text{where: } \Phi = \begin{bmatrix} 0.7789 & 0.4065 & 0 & 0 \\ -0.8130 & 0.5756 & 0 & 0 \\ 0 & 0 & 0.8465 & 0 \\ 0 & 0 & 0 & 0.9048 \end{bmatrix} \tag{3-26}$$

$$y_{[k]} = H \cdot x_{[k]}$$

$$\Gamma(x) = \begin{bmatrix} 0.2211 & 0 \\ 0.8130 & 0 \\ 0 & \dfrac{x_1}{13.028} \\ 0.1903 & 0 \end{bmatrix}$$

The objective of this simplified error trajectory controller is the same as for the exact discrete controller. The solution is easier to determine using a discrete model however since integration of the system model is not required for future predictions. Substituting the discrete model representation into the exact error trajectory objective produces (3-27).

$$E_{f1} = \beta_{0,1} \cdot \left(y_{d,1[k+1]} - H_1 \cdot \hat{x}_{[k+1]}\right) + \beta_{1,1} \cdot \left(\left.\frac{dy_{d,1}}{dt}\right|_{[k+1]} - \begin{bmatrix} 0 & 1 & -\frac{1}{3} & 0 \end{bmatrix} \cdot \hat{x}_{[k+1]} - \begin{bmatrix} 0 & \frac{x_{1[k]}}{6} \end{bmatrix} \cdot u_{[k]} \right) = 0$$

$$E_{f2} = \beta_{0,2} \cdot \left(y_{d,2[k+1]} - H_2 \cdot \hat{x}_{[k+1]}\right) + \beta_{1,2} \cdot \left(\left.\frac{dy_{d,2}}{dt}\right|_{[k+1]} - \begin{bmatrix} 0 & 0 & 0 & -\frac{1}{5} \end{bmatrix} \cdot \hat{x}_{[k+1]} - \begin{bmatrix} \frac{2}{5} & 0 \end{bmatrix} \cdot u_{[k]} \right) = 0 \tag{3-27}$$

$$\text{where: } \quad H_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}, \quad H_2 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

Substitution of $\hat{x}_{[k+1]} = \Phi \cdot x_{[k]} + \Gamma(x) \cdot u_{[k]}$ into (3-27) and solving for the inputs results in the control law, (3-28).

$$u_{1[k]} = \frac{\beta_0 \cdot \left(y_{d,2[k+1]} - H_2 \cdot \Phi \cdot x_{[k]}\right) + \beta_1 \cdot \left(\left.\frac{dy_{d,2}}{dt}\right|_{[k+1]} - \begin{bmatrix} 0 & 0 & 0 & -\frac{1}{5} \end{bmatrix} \cdot \Phi \cdot x_{[k]} \right)}{\beta_0 \cdot 0.1903 + \beta_1 \cdot \left(\frac{2}{5} - \frac{0.1903}{5} \right)}$$

(3-28)

$$u_{2[k]} = \frac{\beta_0 \cdot \left(y_{d,1[k+1]} - H_1 \cdot \Phi \cdot x_{[k]} - 0.2211 \cdot u_{1[k]}\right) + \beta_1 \cdot \left(\left.\frac{dy_{d,1}}{dt}\right|_{[k+1]} - \begin{bmatrix} 0 & 1 & -\frac{1}{3} & 0 \end{bmatrix} \cdot \Phi \cdot x_{[k]} - 0.8130 \cdot u_{1[k]} \right)}{\beta_0 \cdot \frac{x_{1[k]}}{3 \cdot 13.028} + \beta_1 \cdot \left(-\frac{x_{1[k]}}{3 \cdot 13.028} + \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \end{bmatrix} \cdot \Phi \cdot x_{[k]} + \frac{0.2211}{6} \cdot u_{1[k]} \right)}$$

The continuous error trajectory control algorithm is simulated for the example system in both the continuous and discrete domains. The sample time used in the discrete implementation is 0.5. Results are shown in Figure 4 and Figure 5.

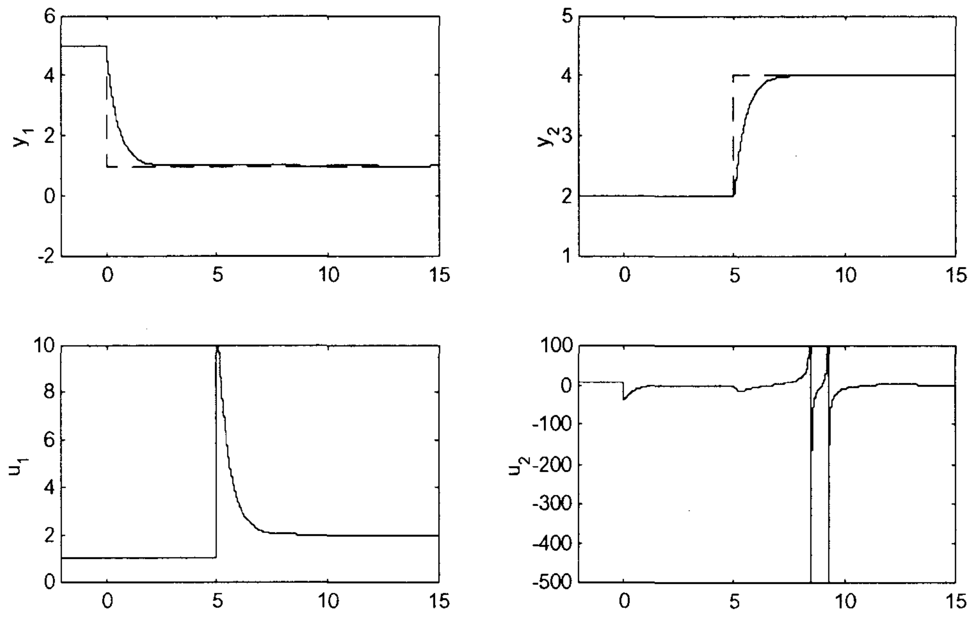

Figure 4, Setpoint tracking of the continuous error trajectory control law (3-19) on the example system in the continuous time domain.

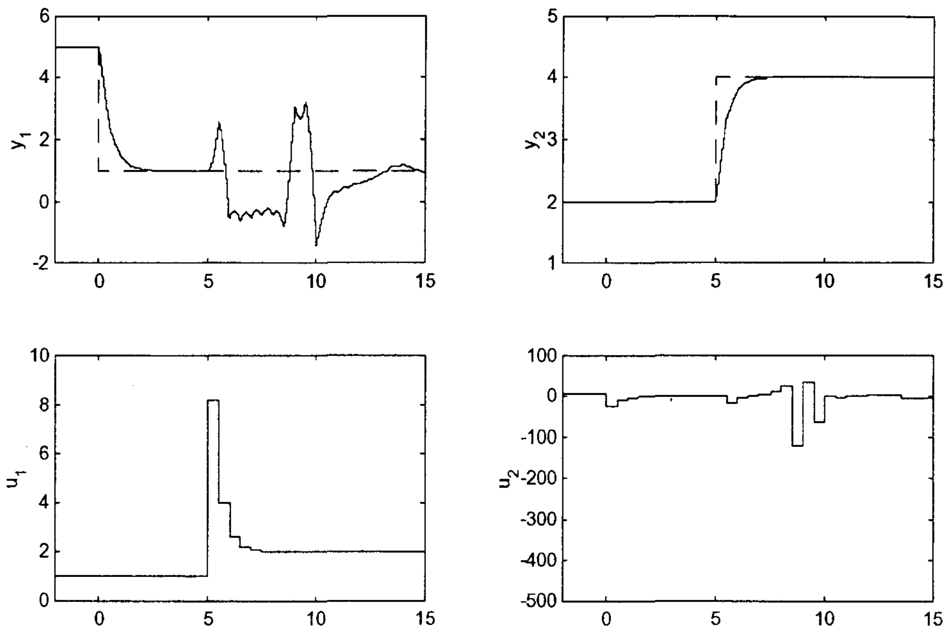

Figure 5, Setpoint tracking of the continuous error trajectory control law (3-19) on the example system in the discrete time domain ($T_s = 0.5$).

The controller is tuned with $\beta_{0,1} = \beta_{0,2} = 1$ and $\beta_{1,1} = \beta_{1,2} = 0.7$. The error trajectory control algorithm performs as expected in the continuous domain but its performance degrades significantly under sampled implementation. Intuitively, the response can be improved by using a faster sampling frequency. With a sampling interval of 0.1 the discrete effects are reduced as shown in Figure 6. In the limit as $T_s \to 0$ the discrete implementation approaches the continuous equivalent. This example agrees with earlier analysis that found the effects of discretization arise as the system states deviate from their value from the last sampling interval.

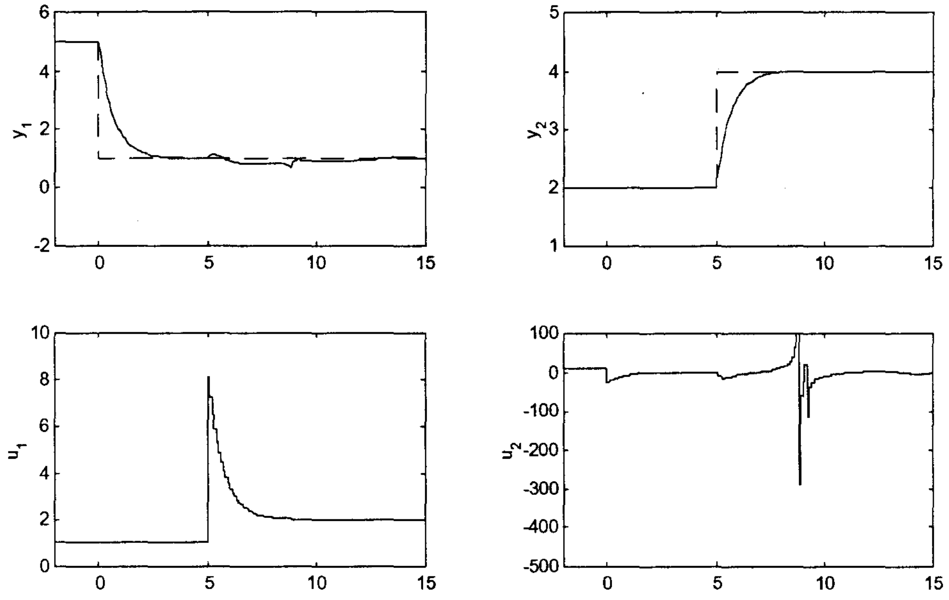

Figure 6, Setpoint tracking of the continuous error trajectory control law (3-19) on the example system in the discrete time domain ($T_s = 0.1$).

One may expect the performance of the exact discrete error trajectory to match or come close to the continuous controller, as is commonly expected for discrete implementations of linear control technologies. This is not the case however as shown in Figure 7. The nature of the discrete error trajectory objective permits excursions from y_d over the sampling interval so long as the output is heading dynamically towards y_d at the next sampling interval. Figure 8 provides a close-up view of y_1 in Figure 7 and shows that at each control execution a positive error is accompanied by a negative derivative and visa versa. The exact discrete error trajectory controller does outperform the discrete implementation of the continuous algorithm. This increase in performance comes at the cost of increased computational load and complexity including an online optimization routine requiring repetitive numerical integration of the system model.

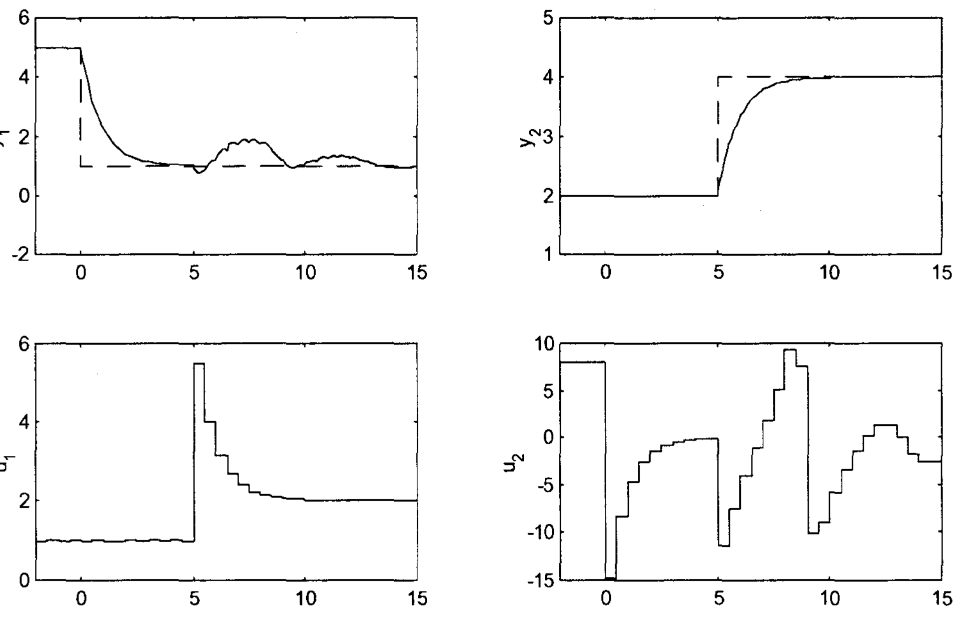

Figure 7, Setpoint tracking of the discrete exact error trajectory control law (3-20) on the example system ($T_s = 0.5$).

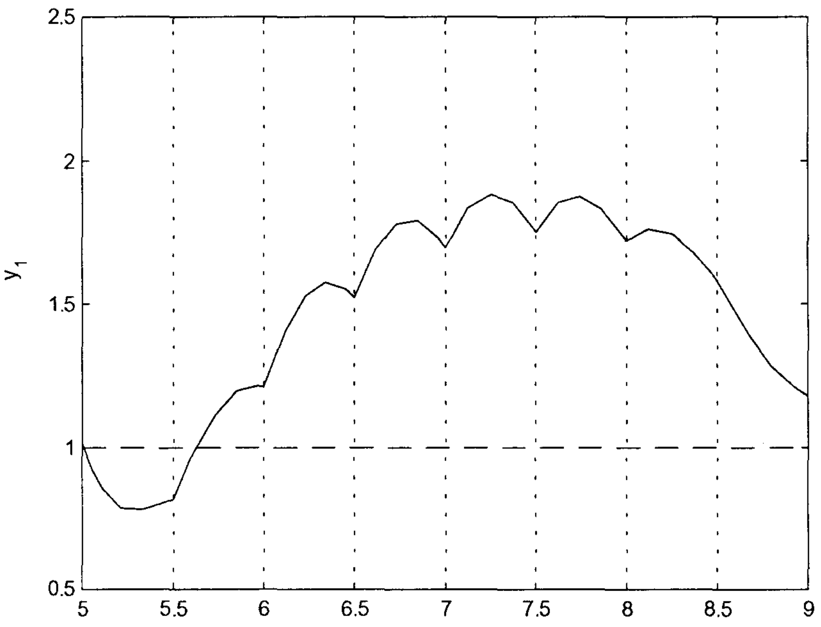

Figure 8, Close-up view of the setpoint tracking of y_1 for the discrete exact error trajectory control law (3-20) on the example system ($T_s = 0.5$) from Figure 7.

The Euler technique improves the predictive quality of the discrete error trajectory controller by using a first order approximation for estimation of future system behaviour. Simulation of the Euler error trajectory control law (3-25) on the example system is shown in Figure 9. The performance of the Euler error trajectory method is better than the discrete implementation of the continuous control law but worse than the exact discrete controller. This is also consistent with the amount of effort involved in the controller development and computational complexity.

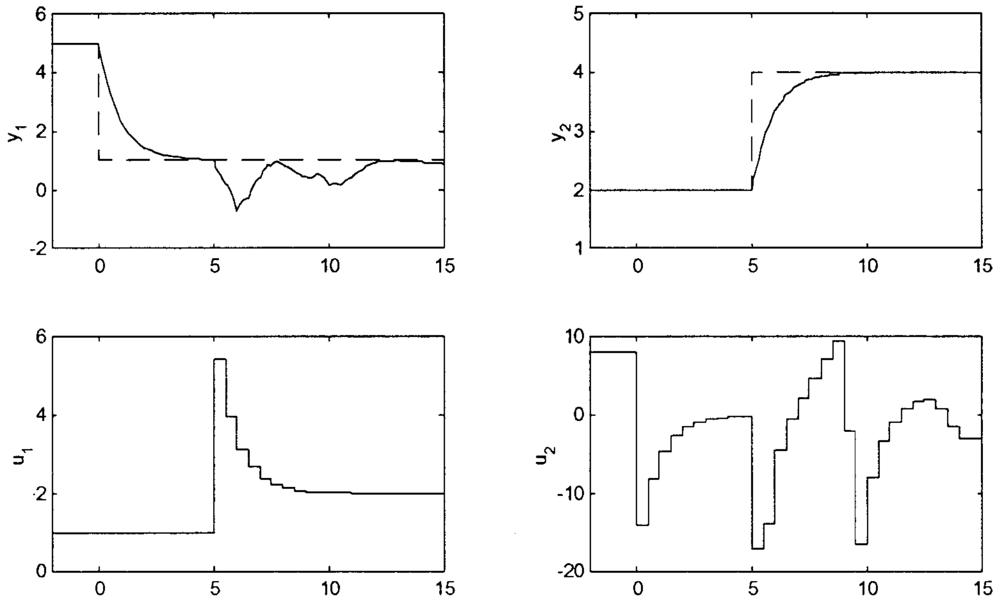

Figure 9, Setpoint tracking of the discrete Euler error trajectory method (3-25) on the example system using ($T_s = 0.5$).

Utilization of a discrete model can reduce the exact discrete error trajectory control algorithm into algebraic equations. Simulation of the discrete model simplification technique, (3-28), for the example system is shown in Figure 10. The system response does not match the exact discrete error trajectory response though due to a modelling error. $\Gamma(x)$ in the discrete NL model contains a term with x_1. In the controller development it was assumed that $x_1 = x_{1[k]}$. x_1 however does not remain constant over the control interval thus this ZOH assumption is a source of error. A convenient property of the example system is that y_1 is controllable only by u_1 therefore u_1 is determined from E_{f2} only. This allows estimation of $x_{1[k+1]}$ regardless of the condition of E_{f1}. Taking advantage of this property the value of x_1 used in $\Gamma(x)$ can be any combination of $x_{1[k]}$ and $x_{1[k+1]}$. Using a weighted average of $x_{1[k]}$ and $x_{1[k+1]}$ ($x_1 = 0.63 \cdot x_{1[k]} + 0.37 \cdot x_{1[k+1]}$) provided a closed loop system response very similar to the exact discrete error trajectory controller, Figure 11.

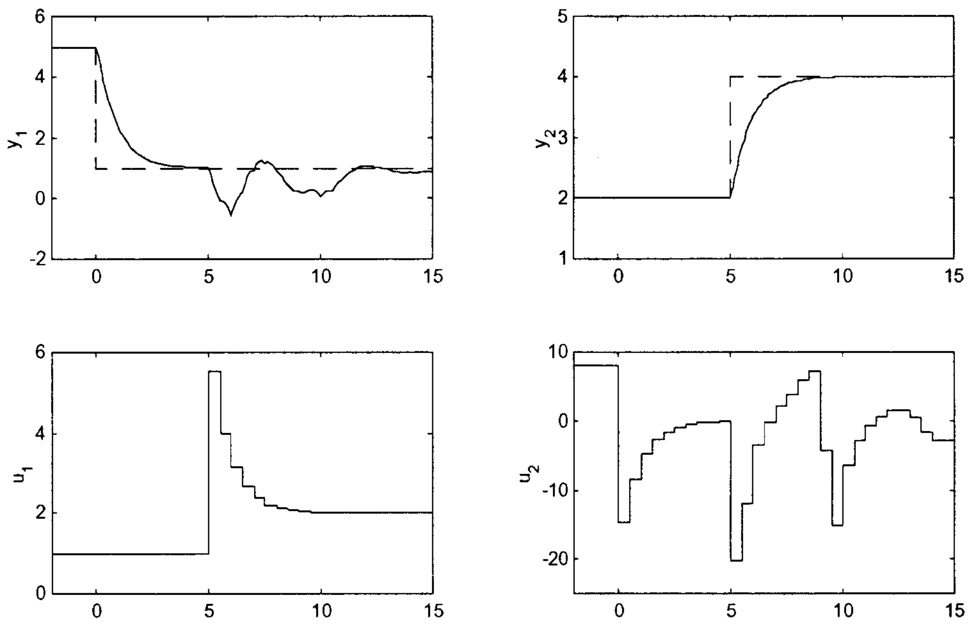

Figure 10, Setpoint tracking of the discrete model error trajectory control law (3-28) on the example system ($T_s = 0.5$).

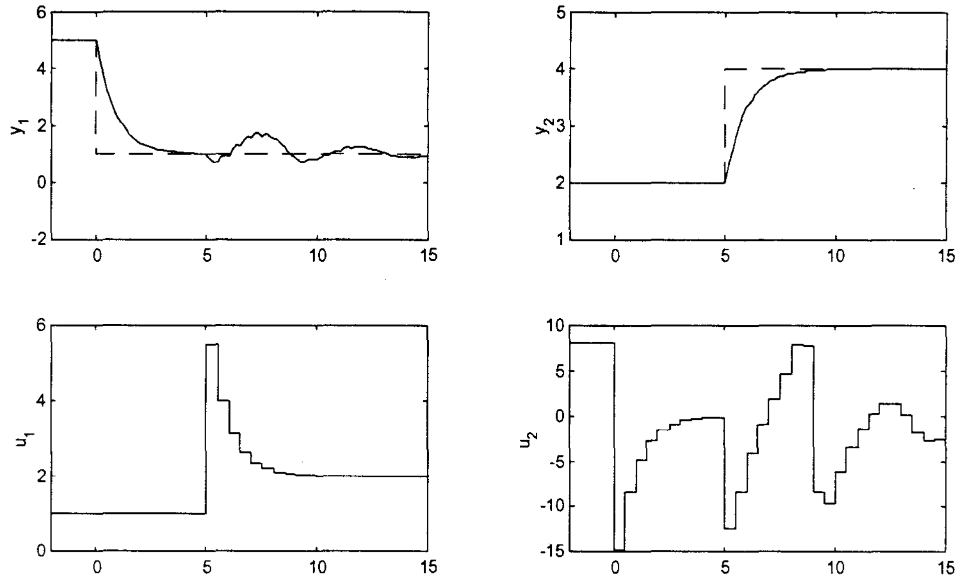

Figure 11, Setpoint tracking of the discrete model error trajectory control law (3-28) on the example system using ($T_s = 0.5$) and a weighed average state prediction for x_1 ($x_1 = 0.63 \cdot x_{1[k]} + 0.37 \cdot x_{1[k+1]}$).

The appropriateness of each of these error trajectory methods depends on the situation. Factors such as availability of a discrete model, computing power, system transient behaviour and controller execution frequency influence the ability and suitability of each of these methods. For instance if the controller execution frequency is 0.01 for the sample system there is no discernable difference in performance between the discrete implementation of the continuous error trajectory control algorithm and the discrete error trajectory control laws. Finding the solution to the exact discrete error trajectory may be difficult or infeasible at that sampling frequency given a set of computing hardware.

The applicability of the error trajectory objective in the discrete domain deserves some question. It was shown that satisfaction of the error trajectory objective at discrete sampling intervals is not necessarily desirable and can lead to unexpected or inappropriate control response. Unlike the predictive control objective (2-3), that is equally valid in its discrete or continuous form, the error trajectory objective does not translate well between the two time domains. In particular the performance of the error trajectory methods presented diminish as the controller sampling interval increases in duration.

3.3 Integrated Error Trajectory Objectives

Given the problems associated with the discrete application of the error trajectory objective a final objective is offered that perhaps best combines the premises of predictive and error trajectory control technologies, to further improve performance. Instead of satisfaction of the objective at the next sampling interval the objective is restructured to minimize the integrated error trajectory over this interval, (3-29).

$$\min_{u[k]} J = \frac{1}{T_s} \cdot \sum_{j=1}^{N_y} \int_{t_k}^{t_k+T_s} E_{f,j}^2 \cdot dt \tag{3-29}$$

For the example system the error trajectory terms, E_f, are shown in (3-30).

$$\begin{aligned} E_{f,1} &= \beta_{0,1} \cdot (y_{d,1} - y_1) + \beta_{1,1} \cdot \left(\frac{dy_{d,1}}{dt} - x_2 + \frac{x_3}{3} - \frac{x_1}{6} \cdot u_2 \right) \\ E_{f,2} &= \beta_{0,2} \cdot (y_{d,2} - y_2) + \beta_{1,2} \cdot \left(\frac{dy_{d,1}}{dt} + \frac{x_4}{5} - \frac{2}{5} \cdot u_1 \right) \end{aligned} \tag{3-30}$$

Simulation of the example system for (3-29) demonstrates the advantages of the discrete integrated error trajectory objective as shown in Figure 12. (3-29) emulates the continuous error trajectory objective in the discrete domain much better than the previous discrete methods. The complexity and computational load are comparable with the exact discrete error trajectory technique yet the performance is superior.

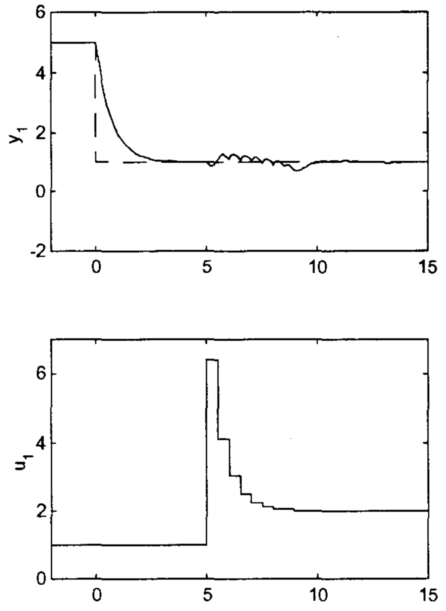

Figure 12, Setpoint tracking of the discrete integrated error trajectory objective (3-29) on the example system using ($T_s = 0.5$).

3.3.1 Optimal Error Trajectory Control

Some deficiencies of the error trajectory method in its continuous or discrete forms are its inability to deal with ill conditioning, enforce system constraints and apply to nonsquare systems. A convenient way of handling ill conditioning in predictive controllers is to include a penalty in the objective function for input move sizes. Also since the objective function for predictive controllers is solved in an NLP or sequence of QPs, constraints are easily incorporated into the control law. Additionally the squareness of a system is not an issue in finding a solution to the control law if set up as optimization problem. Expanding the integrated objective (3-29) with the inclusion of input move suppression, constraints and error weightings is used as a method of removing these aforementioned deficiencies. This expanded integrated error trajectory objective is referred to as optimal error trajectory control. The MIMO optimal error trajectory control law takes the form of (3-31).

$$\min_{u} J = E^T \cdot W_y \cdot E + \nabla u^T \cdot W_u \cdot \nabla u \tag{3-31}$$

subject to: $u_{min} \le u \le u_{max}$

$$|\nabla u| \le u_{roc}$$

where: $$E = \frac{1}{T_s} \cdot \left[\int_{t_k}^{t_k+T_s} E_{f,1}^2 \cdot dt \quad \int_{t_k}^{t_k+T_s} E_{f,2}^2 \cdot dt \quad \ldots \quad \int_{t_k}^{t_k+T_s} E_{f,N_y}^2 \cdot dt \right]^T$$

$$\nabla u = \left[\nabla u_1 \quad \nabla u_2 \quad \ldots \quad \nabla u_{N_u}\right]^T$$

In (3-31) $W_u \in \Re^{N_u \times N_u}$ and is a matrix of input move suppression weightings and $W_y \in \Re^{N_y \times N_y}$ and is the output trajectory error penalty matrix. u_{roc} is a vector of input rate of change constraints. The continuous version of (3-31) simplifies E to $\left[E_{f,1}^2 \quad E_{f,2}^2 \quad \ldots \quad E_{f,N_y}^2\right]^T$ and $\nabla u = \left[du_1/dt \quad du_2/dt \quad \ldots \quad du_{N_u}/dt\right]^T$ with $T_s = 0$.

In the unconstrained continuous case setting $W_u = \underline{0}$ reduces (3-31) to the original trajectory objective, (2-1). The tuning procedures used for predictive controllers can also be applied to W_u and W_y in (3-31). Keep in mind however that W_y does not penalize the error between y and y_d but the sum of the error derivatives. W_y therefore penalizes the distance from the error trajectory state manifold described by $E_{f,j}$.

To demonstrate the power of the optimal error trajectory method the simulation from the previous example is repeated with the input constraint $u_2 \geq 0$. Results for the integrated (3-29) and optimal (3-31) error trajectory objectives are shown in Figure 13 and Figure 14 respectively. The optimal controller is simulated with the three different tunings shown in (3-32). The first tuning represents a balanced tuning in which both output error trajectories are controlled equally. The second and third tunings preferentially control the first and second outputs respectively. A small input move suppression weighting is used for each input to impart stabilizing qualities to ill conditioned regions of operation.

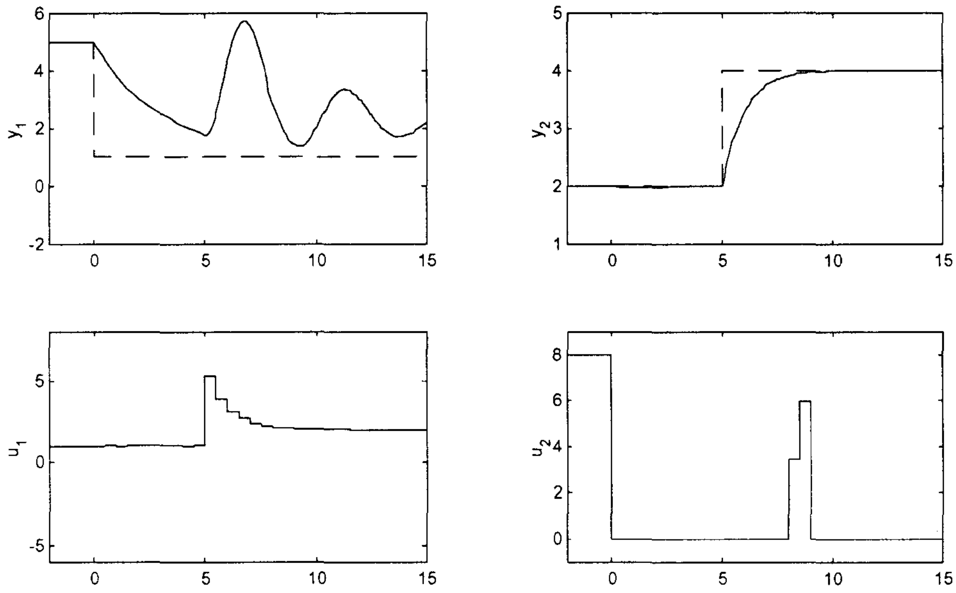

Figure 13, Setpoint tracking of the discrete integrated error trajectory objective (3-29) **on the example system using (** $T_s = 0.5$ **) with input constraint** $u_2 \geq 0$.

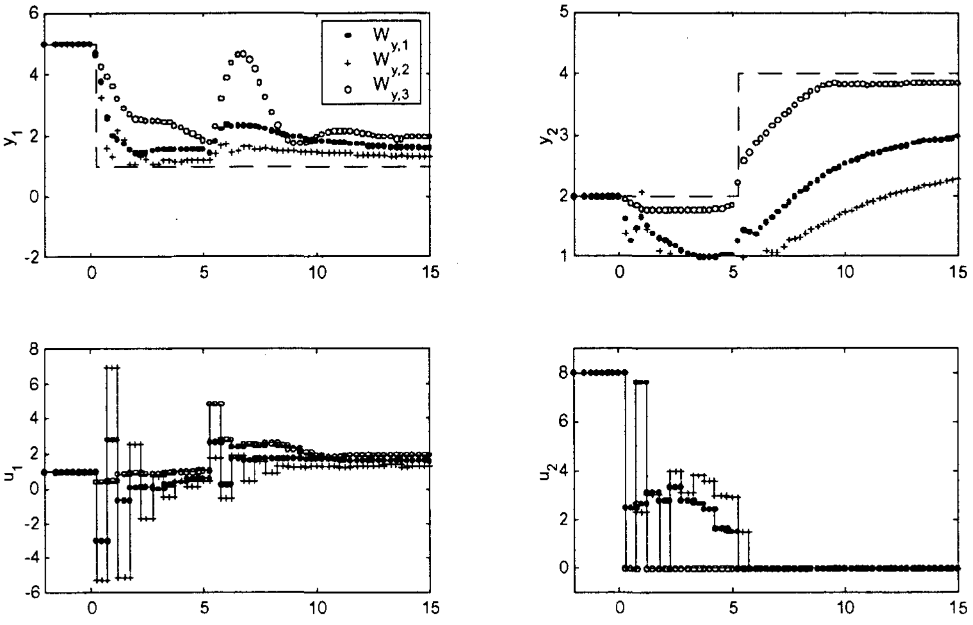

Figure 14, Setpoint tracking of the discrete optimal error trajectory objective (3-31) **on the example system using (** $T_s = 0.5$ **) with input constraint** $u_2 \geq 0$ **and three different output weightings.**

$$W_{y,1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad W_{y,2} = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix} \quad W_{y,3} = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \quad W_u = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \qquad \text{(3-32)}$$

The interesting quality of the optimal error trajectory controller is its multi-structured functionality during constrained operation. Depending on the controller tuning output 1 or 2 can be controlled while input 1 is constrained. Constraint handling capability is not present in MIMO error trajectory methods in literature.

3.4 Summary

There are six error trajectory controllers described in this section; one continuous, and five discrete. The discrete controllers are divided into two categories. The first selects a set of inputs that satisfy the error trajectory objective at the next controller execution using various predictive techniques. The second minimizes the integrated distance from the error manifold defined by the error trajectory objective over the controller execution interval. Using this segregative approach there are three types of error trajectory objectives described, continuous, sampled and integrated. These controllers are summarized in Table 3-1.

Table 3-1, Summary of error trajectory controllers.

Error Trajectory Controller	Design Equation	Sample System Error Function	Objective	Comment
Continuous	(2-13)	(3-19)	C	Derived for continuous domain
Exact Discrete	(3-6)	(3-20)	S	No simplification techniques are used to satisfy the objective
Euler Discrete	(3-11)	(3-25)	S	Uses the Euler method to estimate future system behaviour
Discrete Model	(3-14)	(3-27)	S	Uses a discrete system model to estimate future system behaviour
Integrated	(3-29)	(3-30)	I	Best emulates the continuous error trajectory in discrete domain
Optimal	(3-31)	(3-30)	I	Incorporates tools from predictive control for increased robustness and constraint handling

The objective function column in Table 3-1 refers to the structure of the objective where the C, S and I represent continuous, sampled and integrated respectively. Investigation of the error trajectory methods described within provides several insights into its appropriateness in the discrete domain. It was found that the sampled objective, that satisfies of the error trajectory objective at discrete intervals, does not provide the expected or desired response. Since the error trajectory objective describes a manifold of the error and its derivatives, errors between y and y_d are acceptable provided the error derivatives are directed towards y_d. The exact, Euler and discrete model error trajectory controllers presented are able to maintain the system on the specified error manifold defined by E_f satisfying their objective, yet performance is undesirable.

The integrated error trajectory objective (3-29) was introduced to better emulate the continuous objective over the sampling interval. Simulation for the example system displayed the effectiveness of the technique especially relative to the other discrete algorithms. The problem with (3-29) is its computational complexity. An advantage of the continuous error trajectory method is the simplicity of the resulting control law, which in general consists of a set of algebraic equations. (3-29) requires minimization of an optimization problem necessitating repetitive integration of the system model. This makes the computational load comparable to predictive control methods that have constraint handling and comparably better stability qualities. The increase in complexity therefore decreases the incentive to use the integrated error trajectory.

An optimal error trajectory control structure is introduced to expand the applicability of error trajectory control for both the discrete and continuous time domains. The benefit of the optimal error trajectory controller is its ability to handle ill conditioned regions of operation, input constraints and nonsquare systems.

A general observation of all error trajectory methods is that performance decreases as the controller execution frequency increases in duration. This is an interesting deduction because this is not necessarily true of other control methods such as predictive control. The fundamental difficulty of discrete implementation of the error trajectory method is the inappropriateness of its objective in the discrete domain. If the controller can be implemented such that the sampling time is insignificant with respect to the system dynamics error trajectory control in its continuous form is a very attractive control approach. If the sampling time is relatively large the discrete configurations introduced within provide a means of handling the discretization effects but other control technologies better suited for the discrete domain may be a superior alternative.

4.0 Input Transformation Predictive Control

A variable transformation technique is described that allows for the future prediction of a nonlinear systems through elementary matrix algebraic operations. This technique provides another tool for the control of NL plants and provides potential advantages over other NL predictive methods. This technique referred to as input transformation predictive control (ITPC), is tailored for implementation in the discrete domain and is built on the familiar dynamic matrix control (DMC) architecture.

The impetus behind the author's original development of the ITPC structure was to provide predictive control of a system that known to be nonlinear from empirical experience but may not be well understood on a mechanistic level. Secondly, the computational load and complexity had to be comparable to or less than existing NL predictive control techniques. The design procedure of the ITPC method is started with a review of the DMC algorithm.

4.1 Dynamic Matrix Control

Dynamic matrix control (DMC) (Cutler and Ramaker, 1980) is a popular model predictive control method for linear systems. It is based on the minimization of a quadratic objective as in (2-3). Estimation of future errors, e, is performed using straightforward matrix algebra. The prediction of future outputs for the SISO case is shown in (4-1).

$$y_P = y_f + y_h + y_e + y_{[k]} \qquad \text{where:} \qquad \begin{aligned} y_f &= A_f \cdot \nabla u_f \\ y_h &= A_h \cdot \nabla u_h \end{aligned} \tag{4-1}$$

y_P is the predicted future behaviour of the output for $y_{[k+1]}$ to $y_{[k+P]}$. y_f and y_h are the changes in y for future and past input changes respectively. y_e contains future stochastic disturbances and $y_{[k]}$ is the present value of y. ∇u_h is a vector of past input changes and ∇u_f is a vector of future control moves. y_P, y_f, y_h, y_e and $\nabla u_h \in \Re^P$ where P is the number of control executions in the prediction horizon. The length of ∇u_f is C, where C is the number of control executions in the control horizon. A_f and A_h contain the model parameters derived from discrete impulse response coefficients, m_j, (4-2).

$$A_f = \begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & 0 & \cdots & 0 \\ a_3 & a_2 & a_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_C & a_{C-1} & a_{C-2} & \cdots & a_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_P & a_{P-1} & a_{P-2} & \cdots & a_{P-C+1} \end{bmatrix} \qquad A_h = \begin{bmatrix} b_{2,2} & b_{3,3} & b_{4,4} & \cdots & b_{P+1,P+1} \\ b_{2,3} & b_{3,4} & b_{4,5} & \cdots & b_{P+1,P+1} \\ b_{2,4} & b_{3,5} & b_{4,6} & \cdots & b_{P+1,P+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{2,P+1} & b_{3,P+1} & b_{4,P+1} & \cdots & b_{P+1,P+1} \end{bmatrix} \tag{4-2}$$

$$\text{where:} \qquad \begin{aligned} a_i &= \sum_{j=1}^{i} m_j \\ b_{k,i} &= \sum_{j=k}^{i} m_j \end{aligned}$$

For simplicity no stochastic model is included thus $y_e = 0$. This is a common assumption and represents a random walk distribution of the future disturbances.

Estimation of system behaviour for MIMO systems is accommodated by expanding A_f and A_h as in (4-3).

$$A_f = \begin{bmatrix} A_{f:1,1} & A_{f:1,2} & \cdots & A_{f:1,N_u} \\ A_{f:2,1} & A_{f:2,2} & \cdots & A_{f:2,N_u} \\ \vdots & \vdots & \ddots & \vdots \\ A_{f:N_y,1} & A_{f:N_y,2} & \cdots & A_{f:N_y,N_u} \end{bmatrix} \qquad A_h = \begin{bmatrix} A_{h:1,1} & A_{h:1,2} & \cdots & A_{h:1,N_u} \\ A_{h:2,1} & A_{h:2,2} & \cdots & A_{h:2,N_u} \\ \vdots & \vdots & \ddots & \vdots \\ A_{h:N_y,1} & A_{h:N_y,2} & \cdots & A_{h:N_y,N_u} \end{bmatrix} \quad \textbf{(4-3)}$$

$A_{f:i,j}$ and $A_{h:i,j}$ refers to the dynamic matrix for the jth input's affect on the ith output. Using the MIMO dynamic matrices A_f and A_h, the length of y_p, y_f, y_h, y_e and $y_{[k]}$ become $P \cdot N_y$, $\nabla u_h \in \Re^{P \cdot N_u}$ and $\nabla u_f \in \Re^{C \cdot N_u}$.

Future control moves are determined by minimizing the objective, (2-3). The unconstrained optimal DMC control law, found by substituting (4-1) into (2-3) and solving for ∇u_f, is shown in (4-4) (Seborg *et. al*, 1989).

$$\nabla u_f = K_c \cdot e \qquad \text{where:} \quad \begin{aligned} e &= y_{Tr} - A_h \cdot \nabla u_h - y_{[k]} \\ K_c &= \left(A_f^T \cdot W_y \cdot A_f + W_u\right)^{-1} \cdot A_f^T \cdot W_y \end{aligned} \quad \textbf{(4-4)}$$

K_c is a static control matrix that contains known, static parameters, hence it can be determined once off line. e is a vector of the predicted future deviations from y_{Tr} if no control action is taken and must be calculated at each control execution. The unconstrained control law for DMC controllers is nice in that it results in an explicit solution. This is important for ease of implementation and performance analysis.

Garcia and Morshedi (1986) expanded the DMC algorithm for the constrained case with quadratic programming (QP) techniques. The expanded algorithm termed quadratic dynamic matrix control (QDMC) does not result in an explicit solution but efficient constrained QP optimization algorithms exist to minimize the objective. Qin and Badgwell (1997), report over 2200 industrial applications of controllers similar in structure to the QDMC algorithm, displaying its acceptance in industry and proven ability to control industrial systems.

4.2 Nonlinear Prediction of Future Outputs

An advantage of the DMC algorithm over the general predictive controller described in (2-3) is that the prediction of future system behaviour is estimated through matrix algebra rather than numerical integration of the system model. A variable transformation technique is presented that provides the ability to estimate nonlinear system behaviour through matrix operations.

Prediction of future outputs begins with the derivation of a steady state model, $\psi(x,u)$. Starting with the nonlinear model in (2-7) a steady state system representation results from setting the differentials to zero. For the example system in (3-15) development of the steady state model follows.

Given $y_1 = x_1 + x_3$ and $y_2 = x_4$, the states of interest are x_1, x_3 and x_4.

The steady state value of these states is given by;

$$x_1^{ss} = u_1 \qquad \frac{dx_2}{dt} + \frac{1}{2}\cdot\frac{dx_1}{dt} + 2\cdot x_1 = 2\cdot u_1 \tag{4-5a}$$

$$x_3^{ss} = \frac{1}{2}\cdot x_1^{ss}\cdot u_2 \qquad \text{for:} \quad \frac{dx_3}{dt} + \frac{x_3}{3} = \frac{1}{6}\cdot x_1\cdot u_2 \tag{4-5b}$$

$$x_4^{ss} = 2\cdot u_1 \qquad \frac{dx_4}{dt} + \frac{x_4}{5} = \frac{2}{5}\cdot u_1 \tag{4-5c}$$

$$\text{where:} \quad \frac{dx_1}{dt} = \frac{dx_2}{dt} = \frac{dx_3}{dt} = \frac{dx_4}{dt} = 0$$

The steady state model, ψ, is therefore given by;

$$y^{ss} = \psi(x,u) = \begin{bmatrix} u_1 + \frac{1}{2} \cdot \overset{ss}{x_1} \cdot u_2 \\ 2 \cdot u_1 \end{bmatrix} \tag{4-6}$$

With $\psi(x,u)$ the change in the outputs can be estimated by differencing the steady state model at the present operating condition, $\psi(x_{[k]}, u_{[k]})$, from a new operating point, $\psi(x_{[k]} + \nabla x, u_{[k]} + \nabla u)$. For the example system this results in (4-7).

$$\begin{aligned} \nabla y^{ss} &= \psi(x_{[k]} + \nabla x, u_{[k]} + \nabla u) - \psi(x_{[k]}, u_{[k]}) \\ &= \begin{bmatrix} (u_{1[k]} + \nabla u_1) + \frac{1}{2} \cdot (x_{1[k]} + \nabla x_1) \cdot (u_{2[k]} + \nabla u_2) \\ 2 \cdot (u_{1[k]} + \nabla u_1) \end{bmatrix} - \begin{bmatrix} u_{1[k]} + \frac{1}{2} \cdot x_{1[k]} \cdot u_{2[k]} \\ 2 \cdot u_{1[k]} \end{bmatrix} \\ &= \begin{bmatrix} \nabla u_1 + \frac{1}{2} \cdot (x_{1[k]} \cdot \nabla u_2 + u_{2[k]} \cdot \nabla x_1 + \nabla u_2 \cdot \nabla x_1) \\ 2 \cdot \nabla u_1 \end{bmatrix} \end{aligned} \tag{4-7}$$

Dynamics are incorporated into the nonlinear model much in same way as they are for the DMC controller. The equivalent steady state system representation for linear models used in DMC is;

$$\nabla y^{ss} = \begin{bmatrix} k_{1,1} \cdot \nabla u_1 + k_{1,2} \cdot \nabla u_2 \\ k_{2,1} \cdot \nabla u_1 \end{bmatrix} \qquad \text{where: } k_{i,j} = \text{steady state gain} \tag{4-8}$$

Given a vector of discrete impulse response coefficients for each of the input/output relationships in (4-8) the dynamic system behaviour for an input change is given in (4-9).

$$y_f = \begin{bmatrix} k_{1,1} \cdot \hat{A}_{f:1,1} \cdot \nabla u_{f:1} + k_{1,2} \cdot \hat{A}_{f:1,2} \cdot \nabla u_{f:2} \\ k_{2,1} \cdot \hat{A}_{f:2,1} \cdot \nabla u_{f:1} \end{bmatrix} \qquad \text{where: } A_{f:i,j} = k_{i,j} \cdot \hat{A}_{f:i,j} \tag{4-9}$$

$$y_f = \begin{bmatrix} A_{f:1,1} & A_{f:1,2} \\ A_{f:2,1} & 0 \end{bmatrix} \cdot \begin{bmatrix} \nabla u_{f:1} \\ \nabla u_{f:2} \end{bmatrix}$$

Note that $\hat{A}_{f:i,j}$ is the linear dynamic matrix normalized by the steady state gain $k_{i,j}$. Also $\nabla u_{f:j}$ in (4-8) is a single input change while in (4-9) $\nabla u_{f:j}$ is a vector of input changes of length C.

Utilizing the same procedure to integrate dynamic information into the nonlinear steady state model (4-7) results in (4-10).

$$y_f = \begin{bmatrix} 1 \cdot \hat{A}_{f:1,1} \cdot \nabla u_{f:1} + \frac{1}{2} \cdot \left(\hat{A}_{f:1,2} \cdot x_{f:1} \circ \nabla u_{f:2} + \hat{A}_{f:1,3} \cdot u_{f:2} \circ \nabla x_{f:1} + \hat{A}_{f:1,4} \cdot \nabla u_{f:2} \circ \nabla x_{f:1} \right) \\ 2 \cdot \hat{A}_{f:2,1} \cdot \nabla u_{f:1} \end{bmatrix} \quad \textbf{(4-10)}$$

The $\circ$ operator represents element by element multiplication of the associated vectors.

Estimation of the nonlinear system dynamics requires an increased level of complexity resulting from the inclusion of dependent states in (4-10). These states are important for retention of the nonlinear dynamic properties of the system. Referring back to Figure 2, it was shown that superposition of the independent input step changes for the example system (3-15) did not replicate the actual system response for the same input changes implemented simulateously. It is this dynamic nonlinearity that is retained with the addition of the states in (4-10).

Taking advantage of the sequential nature of the optimization algorithms used to solve the controller objective provides an affective method of dealing with these dependent states. Optimization algorithms iteratively select and test a set of control inputs until a minimum is reached. The role of the process model therefore is to provide an estimated output behaviour given a set of control moves. Introducing an intermediate state prediction layer enables implementation of (4-10). The predicted trajectory of x_1 over the prediction horizon is found using (4-11).

$$x_{p:1} = x_{f:1} + x_{h:1} + x_{1[k]} \qquad \text{where:} \quad \begin{array}{l} x_{f:1} = 1 \cdot \hat{\Xi}_{f:1,1} \cdot \nabla u_{f:1} \\ x_{h:1} = 1 \cdot \hat{\Xi}_{h:1,1} \cdot \nabla u_{h:1} \end{array} \qquad \textbf{(4-11)}$$

$\hat{\Xi}_{f:1,1}$ and $\hat{\Xi}_{h:1,1}$ are the state dynamic matrices describing the dynamic response of x_1 to future and past changes in u_1. To be consistent with the output dynamic matrices $\hat{\Xi}_{f:1,1}$ is normalized by the steady state gain. Given $x_{f:1}$, $\nabla x_{f:1}$ is found by subtracting $x_{f:1[k]}$ from $x_{f:1[k-1]}$ for $k=1$ to P.

The discrete impulse coefficients for the dynamic matrices in (4-10) are found using the differential equations in (4-5). (4-5a) is used to determine the coefficients for $\hat{\Xi}_{f:1,1}$. The dynamics of y_1 are governed by the addition of x_1 and x_3 dynamics. $\hat{\Xi}_{f:1,1}$ describes x_1 dynamics hence $\hat{A}_{f:1,1} = \hat{\Xi}_{f:1,1}$. The differential equation in (4-5b) describes the response of x_3 to changes in $x_1 \cdot u_2$. The change in $x_1 \cdot u_2$ is represented by the sum of the terms $x_{f:1} \circ \nabla u_{f:2}$, $u_{f:2} \circ \nabla x_{f:1}$ and $\nabla u_{f:2} \circ \nabla x_{f:1}$ therefore the impulse coefficients for these terms are derived from (4-5b) and $\hat{A}_{f:1,2} = \hat{A}_{f:1,3} = \hat{A}_{f:1,4}$. The final dynamic matrix, $\hat{A}_{f:2,1}$ is designed using (4-5c).

Setting $\hat{A}_{f:1,2} = \hat{A}_{f:1,3} = \hat{A}_{f:1,4}$ and collecting like terms in (4-10) results in the prediction equation (4-12).

$$y_f = \begin{bmatrix} 1 \cdot \hat{A}_{f:1,1} \cdot \{\nabla u_{f:1}\} + \frac{1}{2} \cdot \hat{A}_{f:1,2} \cdot \{x_{f:1} \circ \nabla u_{f:2} + u_{f:2} \circ \nabla x_{f:1} + \nabla u_{f:2} \circ \nabla x_{f:1}\} \\ 2 \cdot \hat{A}_{f:2,1} \cdot \{\nabla u_{f:1}\} \end{bmatrix} \quad \textbf{(4-12)}$$

The bracketed terms in (4-12), $\{\nabla u_{f:1}\}$ and $\{x_{f:1} \circ \nabla u_{f:2} + u_{f:2} \circ \nabla x_{f:1} + \nabla u_{f:2} \circ \nabla x_{f:1}\}$ are referred to as the transformed intputs represented by $\upsilon_{f:1}$ and $\upsilon_{f:2}$ as shown in (4-13).

$$y_f = \begin{bmatrix} \hat{A}_{f:1,1} & \frac{1}{2} \cdot \hat{A}_{f:1,2} \\ 2 \cdot \hat{A}_{f:2,1} & 0 \end{bmatrix} \cdot \begin{bmatrix} \upsilon_{f:1} \\ \upsilon_{f:2} \end{bmatrix} \quad \textbf{(4-13)}$$

Prediction of system behaviour from past system changes is simpler to compute than future control moves because there is no need for an intermediate state estimation layer. The past changes in state variables are either measured or estimated through filtering techniques. Therefore the transformed inputs contain all known quantities. Prediction of output response from past system changes is shown in (4-14).

$$y_h = \begin{bmatrix} 1 \cdot \hat{A}_{h:1,1} & \frac{1}{2} \cdot \hat{A}_{h:1,2} \\ 2 \cdot \hat{A}_{h:2,1} & 0 \end{bmatrix} \cdot \begin{bmatrix} \upsilon_{h:1} \\ \upsilon_{h:2} \end{bmatrix} \quad \textbf{(4-14)}$$

There is a subtle detail that must be addressed when using continuous states in the transformed inputs. Assuming the present time is $t_{[k]}$ and the time at the next control execution is $t_{[k+1]}$, the state values measured or estimated at time $t_{[k]}$ is $x_{[k]}$. This terminology matches that for inputs in predictive control. Consider DMC control where a set of input changes are used to predict future output values. The change in u at $t_{[k]}$ is the difference between the input over the present control interval, $t_{[k]} \to t_{[k+1]}$, and the last interval, $t_{[k-1]} \rightarrow t_{[k]}$, as shown in **Figure 15**. Applying this same convention the state changes at $t_{[k]}$ produces $\nabla x_{[k]} = x_{[k]} - x_{[k-1]}$.

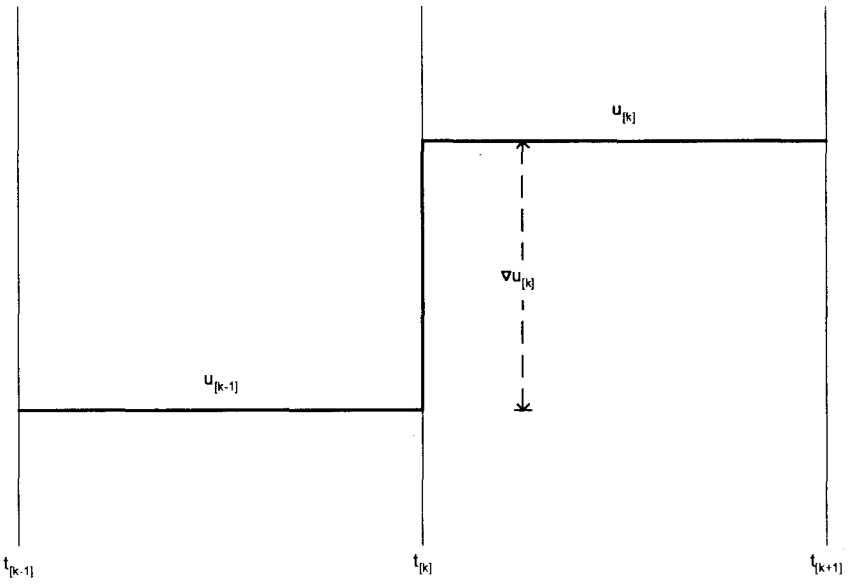

Figure 15, Change in input at time $t_{[k]}$.

For a given state the predicted state vector x_f, and state change vector ∇x_f are defined in (4-15). This convention therefore projects a ZOH structure on the state dynamics. This isn't rigorously valid because the states are continuous and represents a source of error in system dynamic predictions.

$$x_f = \begin{bmatrix} x_{[k]} \\ x_{[k+1]} \\ \vdots \\ x_{[k+P-1]} \end{bmatrix} \qquad \nabla x_f = \begin{bmatrix} \nabla x_{[k]} \\ \nabla x_{[k+1]} \\ \vdots \\ \nabla x_{[k+P-1]} \end{bmatrix} = \begin{bmatrix} x_{[k]} \\ x_{[k+1]} \\ \vdots \\ x_{[k+P-1]} \end{bmatrix} - \begin{bmatrix} x_{[k-1]} \\ x_{[k]} \\ \vdots \\ x_{[k+P-2]} \end{bmatrix} \tag{4-15}$$

Notice that x_f and ∇x_f are vectors of length P. This implies that υ_f must also be of length P to incorporate future changes in x irrespective of the control horizon length. ITPC controllers therefore require A_f to be designed for input vectors of length P independent of C. This does not mean that $C = P$ for but simply that the prediction of future system behaviour requires the inclusion of future state dynamics that contribute to dynamics in the transformed inputs.

The complete prediction of the output behaviour is found by summing up the changes due to past and future system changes (4-16). The output vectors in (4-16) are the same as those described in (4-1).

$$y_p = y_f + y_h + y_e + y_{[k]} \qquad \textbf{(4-16)}$$

The prediction ability of ITPC method is tested using the example system. Previously it was shown that the example system contained both steady state and dynamic nonlinearities. Prediction of the nonlinear state, x_3, using the ITPC prediction method for the same input step changes shown in Figure 2 is shown in Figure 16. Comparison of the ITPC prediction to the superposition of the independent input step changes from Figure 2 displays the ability of the ITPC method to retain both steady state and dynamic nonlinear system properties. The ITPC prediction however does not match exactly the actual system response. The source of error is the ZOH condition applied to the intermediate state x_3. The error can be reduced using a weighted average of the predicted state values over the sampling intervals. For instance at time $t_{[k]}$, the predicted state value can be estimated using (4-17).

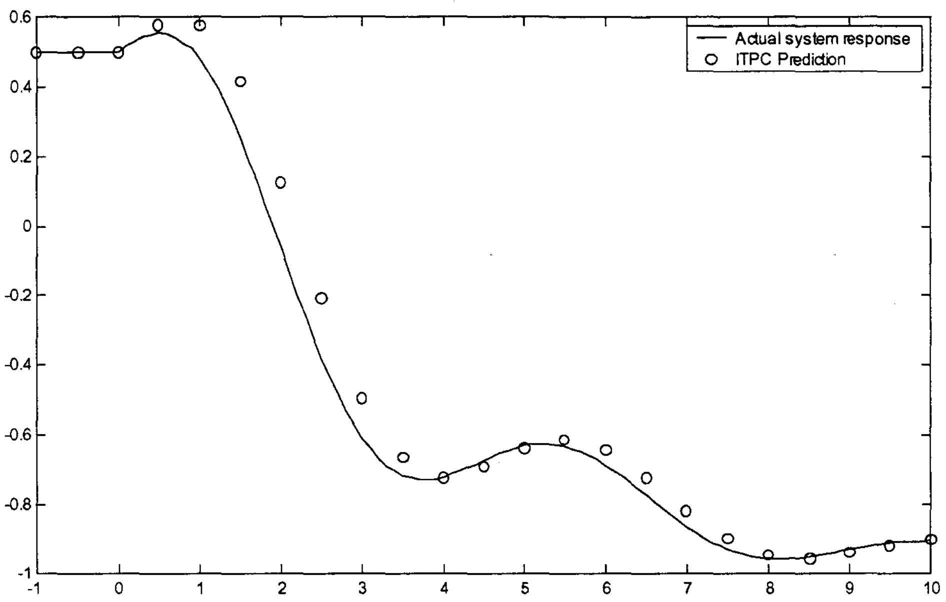

Figure 16, Prediction of the nonlinear state x_3, from the example system (3-15), for the input step changes, $u_1 = 1$ to -1 and $u_2 = 1$ to 2.

$$\overline{x}_{[k]} = (1-\theta)\cdot x_{[k]} + \theta \cdot x_{[k+1]} \qquad \textbf{(4-17)}$$

Using (4-17) with $\theta = 0.35$ the prediction of x_3 for the same input step changes is shown in Figure 17. The error is reduced producing a very good prediction. Note that this predictive ability that retains both steady state and more importantly dynamic nonlinear system characteristics is achieved through linear matrix algebra. Computationally the ITPC prediction method is very efficient and does not require any iterative algorithmic procedures.

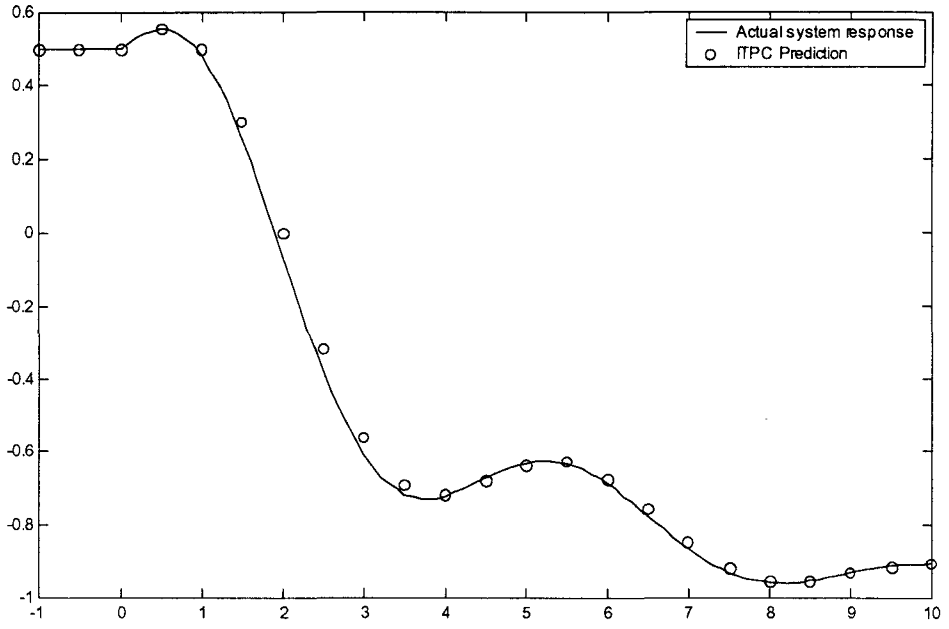

Figure 17, Prediction of the nonlinear state x_3, from the example system (3-15), **for the input step changes, u_1 = 1 to –1 and u_2 = 1 to 2 using a weighted average intermediate state prediction** (4-17), **with θ=0.35.**

The nonlinear prediction technique outlined for the example system describes a specific application of the input transformation method but does not fully reflect its general nature. The general ITPC model design procedure consists of the following steps.

i) *Define a steady state system model* $y^{ss} = \psi(x,u)$.

The steady state model can be derived from a mechanistic white box model, an empirical black box model identified through plant experimentation, or a combination of the two.

ii) *Find the steady state change in the outputs for a change in operating conditions using* $\nabla y^{ss} = \psi(x+\nabla x, u+\nabla u) - \psi(x,u)$.

The inputs are independent variables and the states are dependent. In the previous example the states were simply responses of the system inputs. The states however can be any known or measurable quantity such as a disturbance variable.

iii) *Group the terms in the resulting steady state change function by those that have common dynamics.*

The grouped terms are referred to as the transformed inputs. An important property of the transformed inputs is they are linear in y. This enables the prediction of output dynamics through simple matrix algebra. A difficulty that may arise is the prediction of future state values. This was not a problem for the example system but in special circumstances an analytical state prediction may be difficult or impossible to formulate. In the event that an exact solution can not be found sub-optimal or empirical simplifications can be employed. This problem will be explored through an additional example in section 4.5, ITPC Reactor Design Methodology.

iv) *Determine impulse response coefficients for the design of dynamic matrices.*

As in DMC a dynamic description of each input-output relationship is required to complete the controller model development. The inputs to ITPC control however are not the system inputs, u, but rather the transformed inputs, υ.

Similar to the last step, an analytical representation of system dynamics may be difficult or impossible to develop. Again simplification techniques, such as empirical descriptions identified through experimentation, can be utilized to provide a solution.

4.3 Solving the ITPC Objective Function

Prediction of system behaviour is only half of the ITPC controller's task. Finding the set of inputs to implement that satisfy the objective completes the controller function. Linear predictive controllers with quadratic objectives determine the new set of inputs using quadratic programming techniques. The ITPC objective is quadratic but the process model is not linear thus more sophisticated minimization approaches are required. Solution to the example problem objective is found using the `fmincon` function in MATLAB Version 5.3.0 (R11). The `fmincon` function uses a sequential quadratic programming (SQP) algorithm. SQP is an iterative algorithm consisting of the sequential solution to QP subproblems. Included in the SQP algorithm are major and minor iterations. During major iterations gradient information is collected to form a quadratic model of the optimization problem. The minor iterations include finding the solution to the QP subproblem derived from the quadratic optimization model. Details of SQP are not covered within and the reader is referred to Fletcher and Powell (1963), Shanno (1970), Boggs and Tolle (1989), Dennis *et al.* (1998) and the MATLAB optimization toolbox documentation (Coleman *et al.*) for additional information. Many other powerful optimization algorithms exist and may outperform SQP for specific applications. SQP is used here because it is an accepted general purpose nonlinear constrained optimization tool. The goal of this section is not to describe optimization tools. The algorithm used is transparent to the controller so long as the proper solution is found for the objective. Any appropriate minimization algorithm can be used in place of SQP. In some applications factors such as problem size, convexity of the objective, controller execution frequency and implementation hardware drive the selection of optimization software. Proper matching of the software and application can dictate the success of a controller. For instance a large problem may require the use of simplification techniques to solve the objective in real time.

The set of inputs for implementation are found using a multi-step process. Steps that are not required in the iterative SQP are completed outside the algorithm to improve computational efficiency. These steps include those that determine the predicted error given past system changes including, e_h in (4-18) and the prediction of the intermediate state x_1 given changes in u_1. The simplest of these is the change in x_1, which is estimated by, $x_{h:1} = 1 \cdot \hat{\Xi}_{h:1,1} \cdot \nabla u_{h:1}$, as described in (4-11).

$$e_h = y_{Tr} - \left(y_h + y_e + y_{[k]}\right) \tag{4-18}$$

y_{Tr} is the trajectory toward y_d. The error trajectory controller used an error function of the form in (4-19).

$$\frac{d\varepsilon}{dt} = -0.7 \cdot \varepsilon \tag{4-19}$$

The ITPC y_{Tr} for the example system is designed using this same relation to enforce the equivalent servo trajectories from one operation point to another. The error function (4-19) is projected over the prediction horizon and sampled at the controller sample time ($T_s = 0.5$), to produce y_{Tr}. Given the present measured error the resulting y_{Tr} is shown in (4-20).

$$y_{Tr} = E_{Tr}^{s} \qquad \text{where:} \quad E_{Tr} = \frac{1}{0.7 \cdot s + 1} \cdot \varepsilon[k] + y[k] \tag{4-20}$$

E_{Tr}^{s} is the sampled version of E_{Tr} over the prediction horizon.

The predicted output trajectory from past changes in the transformed inputs y_h, is determined using (4-14). Assuming a random walk disturbance structure simplifies the projected stochastic error term y_e to a vector of zeros.

The resulting error vector e_h is passed to the optimization algorithm. The portion of error that is penalized in the objective is (4-21). This relation is produced through some algebraic manipulation of (4-16) and the penalized error definition $e = y_{Tr} + y_p$.

$$e = e_h + y_f \tag{4-21}$$

The complete ITPC objective function passed to the SQP solver is shown in (4-22).

$$\min_{u_f} \quad J = \frac{1}{2}\left\{ (e_h + y_f)^T W_y (e_h + y_f) + \nabla u_f^T W_u \nabla u_f \right\} \tag{4-22}$$

$$\text{subject to:} \quad u_{min} \le u_f \le u_{max}$$

$$|\nabla u_f| \le u_{roc}$$

$$y_f = \begin{bmatrix} \hat{A}_{f:1,1} & \frac{1}{2}\cdot\hat{A}_{f:1,2} \\ 2\cdot\hat{A}_{f:2,1} & 0 \end{bmatrix} \cdot \upsilon_f$$

$$\upsilon_f = \begin{bmatrix} \nabla u_{f:1} \\ \overline{x}_{f:1} \circ \nabla u_{f:2} + u_{f:2} \circ \nabla \overline{x}_{f:1} + \nabla u_{f:2} \circ \nabla \overline{x}_{f:1} \end{bmatrix}$$

$$\overline{x}_{[k]} = (1-\theta)\cdot x_{[k]} + \theta\cdot x_{[k+1]}$$

$$x_{f:1} = 1\cdot\hat{\Xi}_{f:1,1}\cdot\nabla u_{f:1} + x_{h:1} + x_{1[k]}$$

The SQP solver iterates through values of ∇u_f until the minimum of the objective function is found.

Simulation of the example system for the described ITPC controller is shown in Figure 18. The controller sample time is 0.5, the prediction horizon is 20 ($P = 40$) and the control horizon is 2.5 ($C = 5$). The tuning parameters used in the simulation are shown in (4-23).

$$W_y = \begin{bmatrix} 4\cdot I_P & 0_P \\ 0_P & 10\cdot I_P \end{bmatrix} \qquad W_u = \begin{bmatrix} 1\cdot I_C & 0_C \\ 0_C & 0.05\cdot I_C \end{bmatrix} \tag{4-23}$$

where:
I_i is an identity matrix of size $i \times i$
0_i is a zero matrix of size $i \times i$

In this example the performance of the ITPC controller is superior to all the error trajectory controllers with the exception of the integrated discrete error trajectory controller. One distinction between these controllers is evident. The ITPC controller is able to match the performance with much smaller input changes. This is generally considered a desirable property of a control technology for intangible reasons such as longer actuator life. The optimal error trajectory controller can be utilized with input weighting to suppress the input changes to match that of the ITPC controller. This detuning however results in significant degradation of performance as shown in Figure 19.

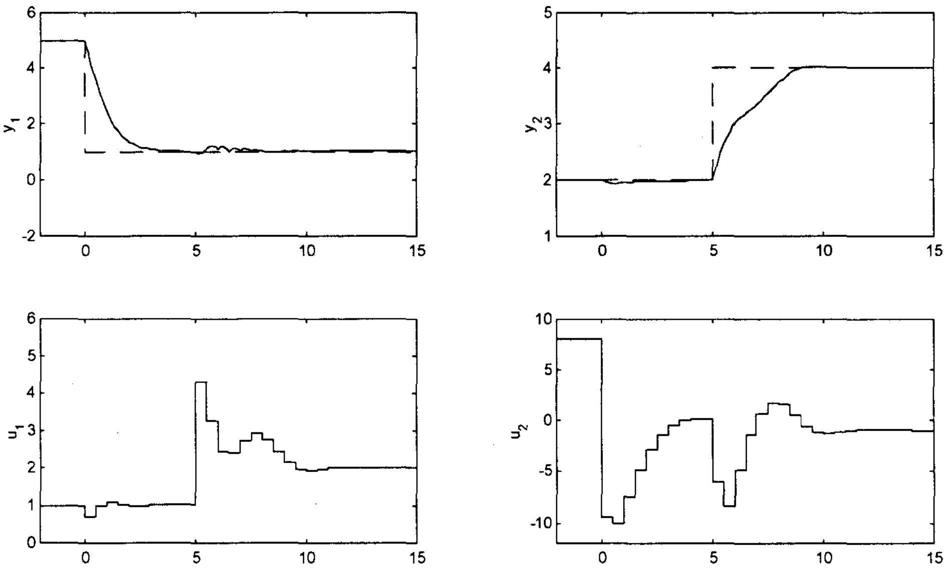

Figure 18, Setpoint tracking of the ITPC controller from section 4.2 on the example system using ($T_s = 0.5$).

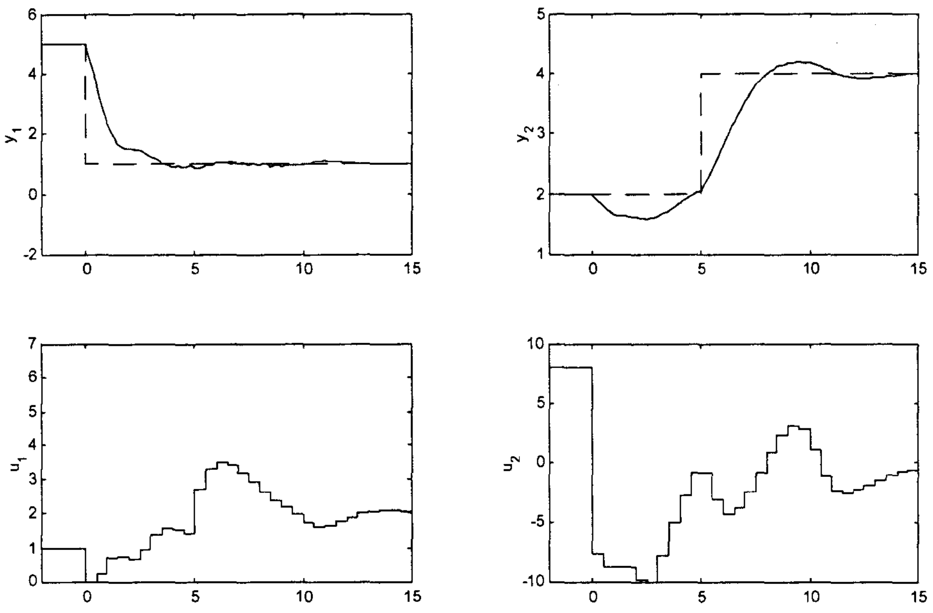

Figure 19, Setpoint tracking of the discrete optimal error trajectory controller (3-31) **detuned to match the magnitude of the ITPC input changes on the example system using (** $T_s = 0.5$ **).**

Application of the nonlinear predictive controller described in section 2.2 is shown in Figure 20. The performance is very similar to ITPC including the selection of control inputs. The only algorithmic difference between the ITPC method and the general nonlinear predictive controller is the prediction method. The similarity in the control response suggests that the predictive ability of the ITPC method is comparable to the general nonlinear controller that includes numerical integration of the full system model.

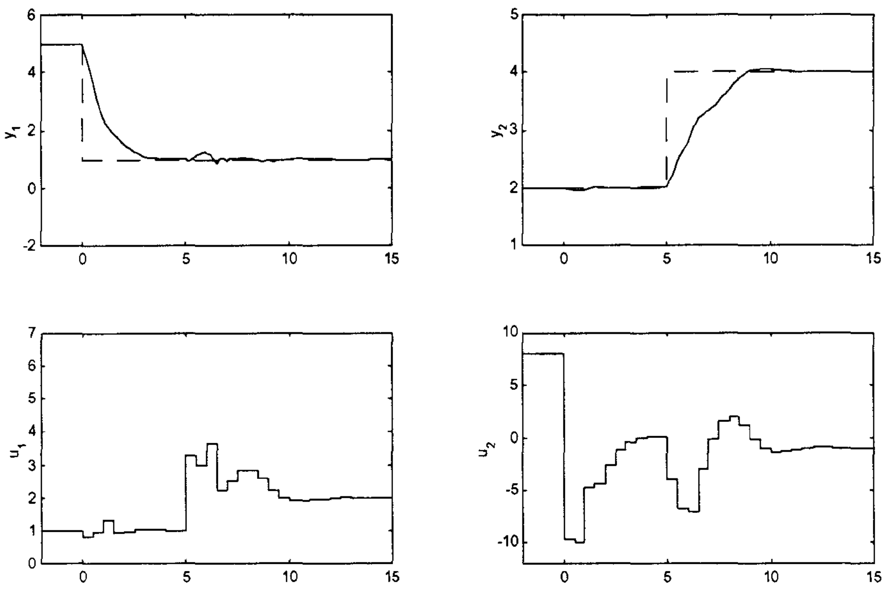

Figure 20, Setpoint tracking of the nonlinear predictive controller from section 2.2 on the example system using ($T_s = 0.5$).

4.4 Explicit ITPC Solution (Unconstrained)

In general explicit solutions for nonlinear predictive controllers do not exist. A special case of the NL ITPC controller can however be expressed in an explicit form. The conditions that must be satisfied to produce an explicit ITPC controller are:

i) The inputs must be unbounded (unconstrained).

ii) An input inverse function, $\nabla u = \phi(\nabla \upsilon)$, must exist that translates the ITPC transformed inputs into a unique set of physical system inputs on $\Re^{N_u}$.

The objective of an ITPC controller that satisfies these criteria is shown in (4-24).

$$\min_{\upsilon_f} \quad J = \frac{1}{2}\left\{e^T W_y e + \nabla\upsilon_f{}^T W_\upsilon \nabla\upsilon_f\right\} \tag{4-24}$$

$$\text{subject to: } \nabla u = \phi(\nabla\upsilon)$$

A property that differentiates (4-24) from the general nonlinear predictive control objective is the replacement of the physical system inputs with the ITPC transformed inputs. With this structure physical input changes are not penalized but rather the transformed input changes. This requires that the input inverse function be well conditioned for all regions of operation. An ill-conditioned input inverse function may lead to erratic input moves for very small perturbations in υ_f.

Borrowing a technique from DMC, the solution to this objective is;

$$\nabla\upsilon_f = K_c \cdot e \qquad \text{where:} \qquad \begin{aligned} e &= y_{Tr} - A_h \cdot \nabla\upsilon_h - y_{[k]} \\ K_c &= \left(A_f^T \cdot W_y \cdot A_f + W_\upsilon\right)^{-1} \cdot A_f^T \cdot W_y \end{aligned} \tag{4-25}$$

Even though υ_f must translate into a unique set of future inputs, only the set that is implemented at the present control execution need to be calculated. This is a convenient and powerful result because using the ZOH representation for the continuous state dynamics, the present state values and changes are known. This eliminates the need for an intermediate state prediction layer required for the more general constrained ITPC controller.

Inspection of the explicit form of the ITPC control law reveals that it is really the implementation of a linear control law on a transformed linear system. This procedure draws many parallels to geometric techniques such as feedback linearization.

Furthermore results for stability of linear predictive controllers may be extended to the explicit ITPC and general ITPC.

Implementation of (4-24) is the same as for DMC controllers with an additional input transformation inversion step. The explicit ITPC controller is executed in the following sequences.

i) Present outputs and states are measured or estimated through filtering techniques.
ii) The error vector e is calculated using (4-25).
iii) υ_f is calculated using the controller gain, K_c, in (4-25). The controller gain contains static known quantities therefore it can be determined once off-line.
iv) The set of inputs to be implemented at the present control interval are calculated from υ_f using the input inverse function, $\nabla u = \phi(\nabla \upsilon)$.

Design of the explicit ITPC controller for the example system is identical to the general ITPC design up to solving the objective function. Given the nonlinear system model (4-13) and (4-14), the controller gain is calculated using (4-25). The error vector e is determined by substituting current measured outputs and past measured transformed inputs into (4-25). The product of K_c and e results in a set of future transformed input control moves, $\nabla \upsilon_f$. Only the control moves implemented at the current controller execution are of interest. These are the first and $C+1$ elements of $\nabla \upsilon_f$. Notice that the length $\nabla \upsilon_f$ of need not be equal to P. Substituting these results into the transformed input variable definition from (4-12) results in (4-26). Solving for the physical system inputs produces (4-27), the input inverse function.

$$\nabla \upsilon_f = \begin{bmatrix} \nabla \upsilon_{f(1)} \\ \nabla \upsilon_{f(c+1)} \end{bmatrix} = \begin{bmatrix} \nabla u_{1[k]} \\ x_{1[k]} \cdot \nabla u_{2[k]} + \nabla x_{1[k]} \cdot u_{2[k]} + \nabla x_{1[k]} \cdot \nabla u_{2[k]} \end{bmatrix} \tag{4-26}$$

$$\begin{bmatrix} \nabla u_{1[k]} \\ \nabla u_{2[k]} \end{bmatrix} = \begin{bmatrix} \nabla \upsilon_{f(1)} \\ \dfrac{\nabla \upsilon_{f(2)} - \nabla x_{1[k]} \cdot u_{2[k]}}{x_{1[k]} + \nabla x_{1[k]}} \end{bmatrix} \tag{4-27}$$

Simulation of the explicit ITPC controller on the example system is shown in Figure 21. The controller tuning for the simulation are shown in (4-28). The controller sample time is 0.5, the projection horizon is 20 ($p = 40$) and the control horizon is 2.5 ($c = 5$) to match the previous simulations.

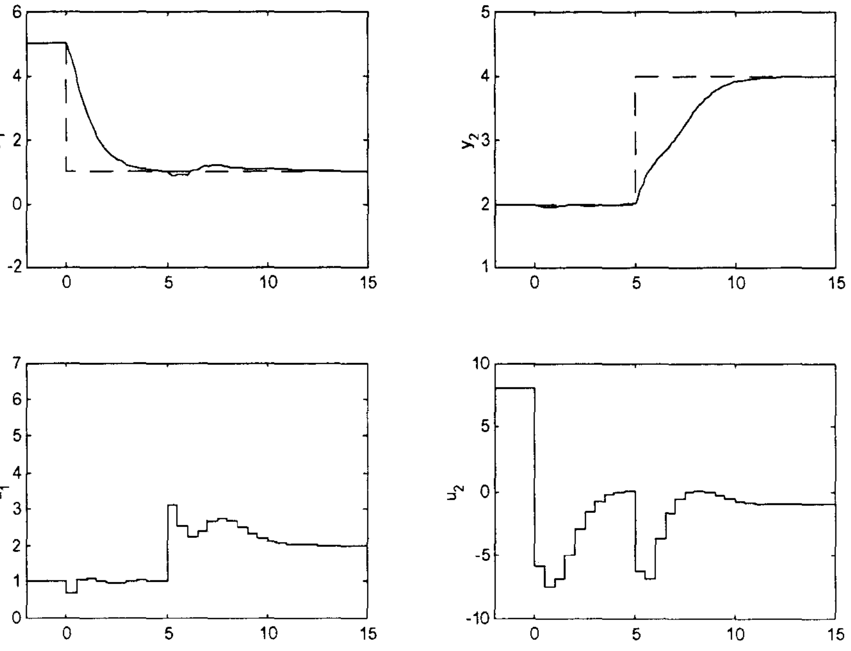

Figure 21, Setpoint tracking of the explicit form of the ITPC controller from section 4.2 on the example system using ($T_s = 0.5$).

$$W_y = \begin{bmatrix} 10 \cdot \mathbf{I}_P & \mathbf{0}_P \\ \mathbf{0}_P & 10 \cdot \mathbf{I}_P \end{bmatrix} \qquad W_\upsilon = \begin{bmatrix} 1 \cdot \mathbf{I}_C & \mathbf{0}_C \\ \mathbf{0}_C & 0.05 \cdot \mathbf{I}_C \end{bmatrix} \tag{4-28}$$

Performance of the explicit ITPC controller is comparable to the general ITPC but not identical. The performance difference primarily arises from two sources. The first is the alteration in move suppression weighting. In the general IPTC controller the actual system input are penalized and the explicit controller penalizes the transformed inputs. The second difference results from setting the transformed input changes to zero past the control horizon. In the general ITPC formulation the system input changes are zero past the control horizon, where the dynamics of the continuous states are modelled in υ_f for the duration of the prediction horizon.

There is an interesting extension to the explicit ITPC structure for constrained systems in which there exists an input inverse function. If the input constraints can be expressed in terms of υ, the ITPC objective can be designed as a QP with nonlinear input constraints. This is significant because the general ITPC objective function requires the solution of a NLP. In optimization theory QP problems are much easier to solve than NLP and much of the existing predictive control technologies used in industry are based on QP objectives. This topic is not explored within but is pointed out as a possible source for future study.

4.5 ITPC Reactor Design Methodology

A second example system is introduced to display additional properties of the ITPC controller. The second example models a physical system consisting of a continuous stirred tank reactor (CSTR) with a first order exothermic reversible reaction

$(A \leftrightarrow B)$ and a cooling jacket for thermal management. A diagram of the system is shown in Figure 22. The kinetic model including coefficients is derived in Fogler (1992) and the complete system model is shown in (4-29).

$$\frac{dC_a}{dt} = \frac{F}{V}\cdot(C_{ai} - C_a) - r_a$$

$$\frac{dC_b}{dt} = -\frac{F}{V}\cdot C_b + r_a$$

$$\frac{dT}{dt} = \frac{F}{V}(T_i - T) + \frac{(-\Delta H_R)}{\rho\cdot c_p}\cdot r_a + \frac{Q}{V\cdot\rho\cdot c_p} \tag{4-29}$$

$$\frac{dT_J}{dt} = \frac{F_C}{V_C}(T_C - T_J) + \frac{Q_C}{V_C\cdot\rho_C\cdot c_{p,C}}$$

where: $r_a = k(T)\cdot\left(C_a - \frac{C_b}{K_e}\right)$

$$K_e = 10^5\cdot\exp\left[-33.78\cdot\frac{(T-298)}{T}\right]$$

$$k(T) = k_0\cdot\exp\left[-\frac{E}{R\cdot T}\right]$$

$$Q = -Q_C = U\cdot A\cdot(T_J - T)$$

Over the range of CSTR operating conditions the heat of reaction, heat capacity and density are considered constant and have values of $\Delta H_R = -20000$, $c_p = 1$ and $\rho = 1000$. Other system constants are the reactor volume $(V = 5)$, rate constant $(k_0 = 1000)$, activation energy $(E = 40000)$, gas constant $(R = 8.314)$, heat transfer coefficient $(U = 1000)$, cooling jacket area $(A = 5)$ and cooling jacket volume $(V_c = 1)$.

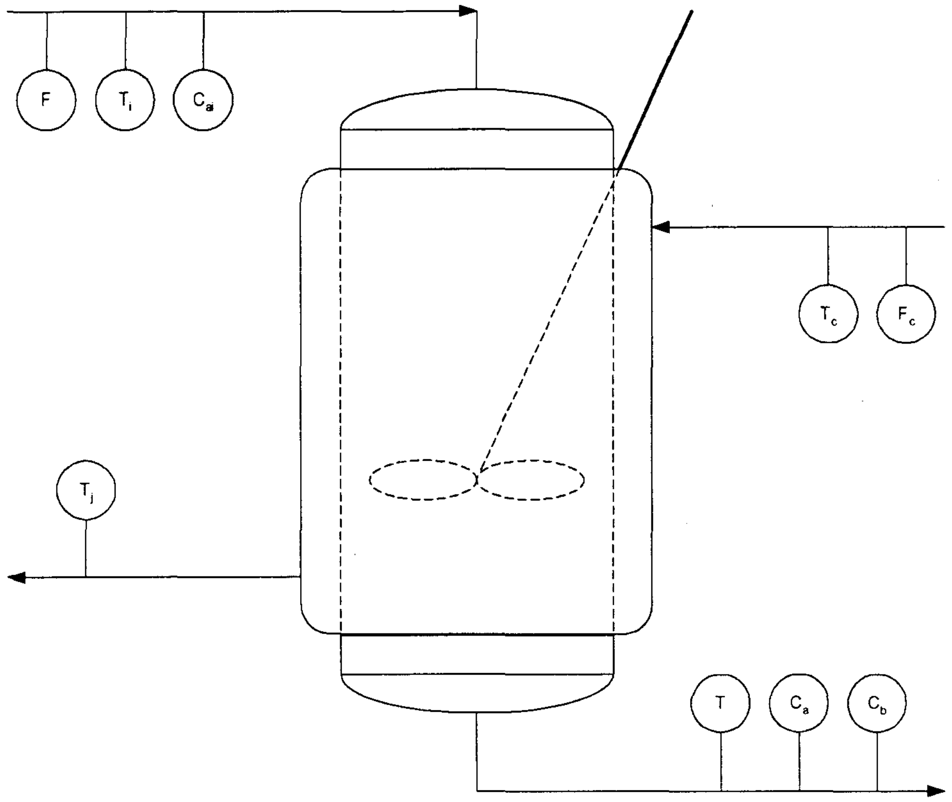

Figure 22, Diagram of the CSTR system in (4-29).

The system states are the reactant concentration C_a, product concentration C_b, reactor temperature T and cooling jacket temperature T_J. The goal of the controller is to adjust the coolant flow F_C, to maximize the product concentration subject to operational constraints, leading to the objective function (4-30). The emphasis of the controller is on rejection of low frequency disturbances.

$$\min_{F_c} J = -W_y \cdot C_{b,f} + \nabla F_{c,f}^T \cdot W_u \cdot \nabla F_{c,f} \quad \textbf{(4-30)}$$

subject to: $F_{c,min} \leq F_c \leq F_{c,max}$

$T \leq T_{max}$

The system constraints are $F_{c,min} = 0$, $F_{c,max} = 10$ and $T_{max} = 550$. As it will turn out these constraints are not an issue for control. They are included in the system objective for the purpose of displaying the flexibility ITPC objective structure. In (4-30) we see the use of two different types of parameters from the previous predictive control objective function description (2-3). The new parameters are the optimization term $-w_y.C_{b,f}$ and the state constraint. $-w_y.C_{b,f}$ is referred to as an optimization term because the controller is minimizing the parameter rather than regulating around a desired value or trajectory. Previously the objective function only included input constraints but state constraints can also be included in the ITPC objective provided they are determined in an intermediate state layer or as an output.

The reactant flow F, inlet reactant concentration C_{ai}, inlet reactant temperature T_i and coolant inlet temperature T_c are stochastic variables that can not be controlled. These are treated as disturbance variables used to predict the future system behaviour not included in u_f.

For a given set of process inputs F and C_{ai} there is an optimal reactor temperature that maximizes the product concentration. This optimal temperature results from the reversibility of the exothermic reaction. At low temperatures the equilibrium drives the reaction forward but the kinetics are slow. At high temperatures the kinetics are sped up but the equilibrium shifts, favouring the reverse reaction. Figure 23, Figure 24 and Figure 25 display the nonlinear steady state map for the expected operating range of T, F and C_{ai}. These opposing driving forces create a singular point where the coolant flow has a zero gain with respect to the output C_b. In addition the gain between F_c and C_b changes sign across the singular operating point.

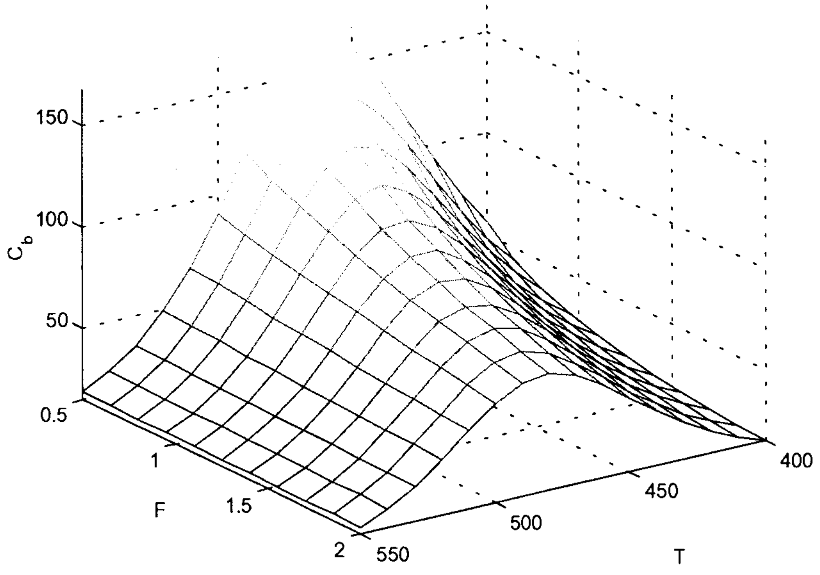

Figure 23, Nonlinear steady state map of the CSTR product concentration C_b in (4-29) **for reactant flow F and reactor temperature T variations.**

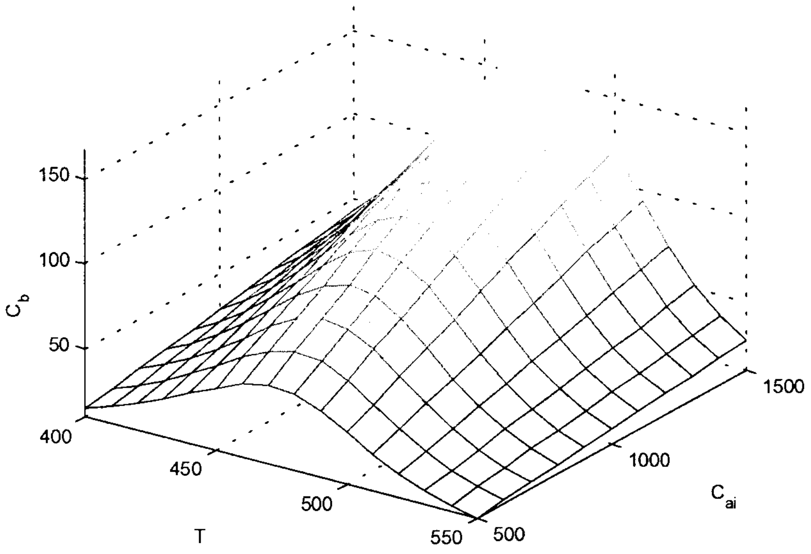

Figure 24, Nonlinear steady state map of the CSTR product concentration C_b in (4-29) **for reactant inlet concentration C_{ai} and reactor temperature T variations.**

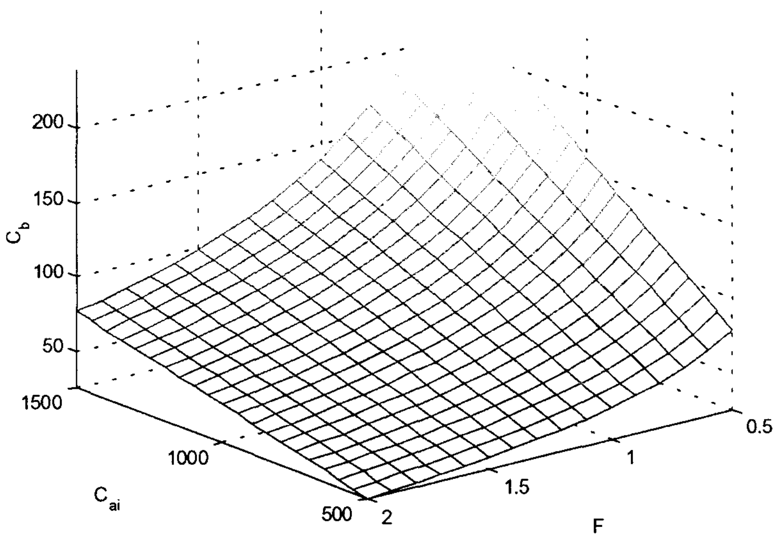

Figure 25, Nonlinear steady state map of the CSTR product concentration C_b in (4-29) **for reactant inlet concentration C_{ai} and inlet flow F, variations.**

The controller can be thought of a system optimizer outer loop with an inner regulatory loop. The outer loop determines the optimal CSTR temperature and the inner loop manipulates the coolant flow to achieve this desired temperature.

The controller design begins with the derivation of a steady state model for the product concentration, C_b. This is found by setting the derivative of the ODE for C_b in (4-29) to zero. This results in the steady state relationship (4-31) where the ss superscript indicates steady state value of a state.

$$0 = -\frac{F}{V}\cdot C_b^{ss} + r_a \qquad \textbf{(4-31)}$$

Substitution of the equation for r_a from (4-29) and solving for C_b^{ss} produces (4-32).

$$C_b^{ss} = \frac{k(T) \cdot C_a}{\frac{F}{V} + \frac{k(T)}{K_e}} \tag{4-32}$$

(4-32) contains parameters that are influenced by temperature changes and disturbances including $k(T)$, K_e and C_a. The change in $k(T)$ and K_e will be dealt with later. The steady state change in C_a is found by setting its derivative to zero and solving for C_a^{ss} as shown in (4-33) and (4-34).

$$0 = \frac{F}{V} \cdot \left(C_{ai} - C_a^{ss}\right) - r_a \tag{4-33}$$

$$C_a^{ss} = \frac{\frac{F}{V} \cdot C_{ai} + k(T) \cdot \frac{C_b^{ss}}{K_e}}{\frac{F}{V} + k(T)} \tag{4-34}$$

At this point two approaches can be taken to deal with the presence of C_a in (4-32). C_a could be treated as an intermediate state to be calculated prior to C_b or the relation for C_a^{ss} from (4-34) could be substituted into (4-32). The first option provides more flexibility in describing the system dynamics since the dynamics of C_a and C_b can be identified separately. In this example there is a flaw in this approach however because C_b appears in (4-34). This creates a circular reference between C_a and C_b. For this reason the second approach is used. Substituting (4-34) into (4-32) produces the function (4-35). Solving for C_b^{ss} results in the steady state function Ψ_{C_b}, (4-36).

$$C_b^{ss} = \frac{k(T)\cdot\left[\dfrac{\frac{F}{V}\cdot C_{ai} + k(T)\cdot\frac{C_b^{ss}}{K_e}}{\frac{F}{V}+k(T)}\right]}{\frac{F}{V}+\frac{k(T)}{K_e}} \qquad \textbf{(4-35)}$$

$$C_b^{ss} = \Psi_{C_b} = \frac{F\cdot k(T)\cdot C_{ai}\cdot K_e}{V\cdot\left(K_e\cdot\theta_1 - k(T)^2\right)} \qquad \text{where: } \theta_1 = \left(\frac{F}{V}+\frac{k(T)}{K_e}\right)\cdot\left(\frac{F}{V}+k(T)\right) \qquad \textbf{(4-36)}$$

If it can be assumed that the dynamic response of C_b for changes in C_{ai}, F, $k(T)$ and K_e can be lumped through the Ψ_{C_b} transformation then the steady state function in (4-36) can also be assumed to be the transformed input, υ_{C_b}. As in the first example analytical methods could be used to determine dynamic properties and verify this assumption but in this case empirical testing is used. The objective of the empirical test is to sufficiently excite υ_{C_b} from changes in C_{ai}, F and T to verify the lumped dynamics assumption and identify a linear model between C_b and υ_{C_b}. For simplicity the dynamics of interest are of relatively low frequency. The experiment therefore focuses on low frequency input changes. This approach is not ideal in real situations but is sufficient for this perfectly known system without unknown disturbances or noise. The factorial experiment shown in Table 4-1, designed in C_{ai}, F and F_c, is chosen as the experimental test procedure.

Table 4-1, Factorial experiment for the CSTR example.

Time	C_{ai}	F	F_c
-100	1000	1.0	1.5
0	1050	1.5	2.0
60	1050	1.5	1.0
120	1050	0.5	2.0
180	1050	0.5	1.0
240	950	1.5	2.0
300	950	1.5	1.0
360	950	0.5	2.0
420	950	0.5	1.0
480	1000	1.0	1.5

Results for the system states C_a, C_b, T and T_J for this experiment are shown in Figure 26. Figure 27 contains a plot of actual and predicted ($\hat{C}_b$) product concentration for the experiment as well as υ_{C_b}. The model for estimation of C_b from υ_{C_b} is;

$$\hat{C}_b = \frac{0.9833 \cdot z^{-1}}{1 - 0.0156 \cdot z^{-1}} \cdot \upsilon_{C_b} \qquad \text{where:} \quad \begin{aligned} \upsilon_{C_b}[k] &= \frac{F[k] \cdot k(T)_{[k]} \cdot C_{ai}[k] \cdot K_e[k]}{V \cdot \left(K_e[k] \cdot \theta_1[k] - k(T)_{[k]}^2\right)} \\ \theta_1[k] &= \left(\frac{F[k]}{V} + \frac{k(T)_{[k]}}{K_e[k]}\right) \cdot \left(\frac{F[k]}{V} + k(T)_{[k]}\right) \end{aligned} \qquad \textbf{(4-37)}$$

(4-37) is a linear discrete time model with a sampling time of 2. Theoretically the steady state gain between υ_{C_b} and C_b is 1. This is verified as the empirical model (4-37) has a gain of 1.00. The lumped dynamic assumption used in setting $\upsilon_{C_b} = \Psi_{C_b}$ appears reasonable from the result in Figure 27. Note that (4-37) is the dynamics of $\hat{C}_b$ from υ_{C_b} and not the physical inputs.

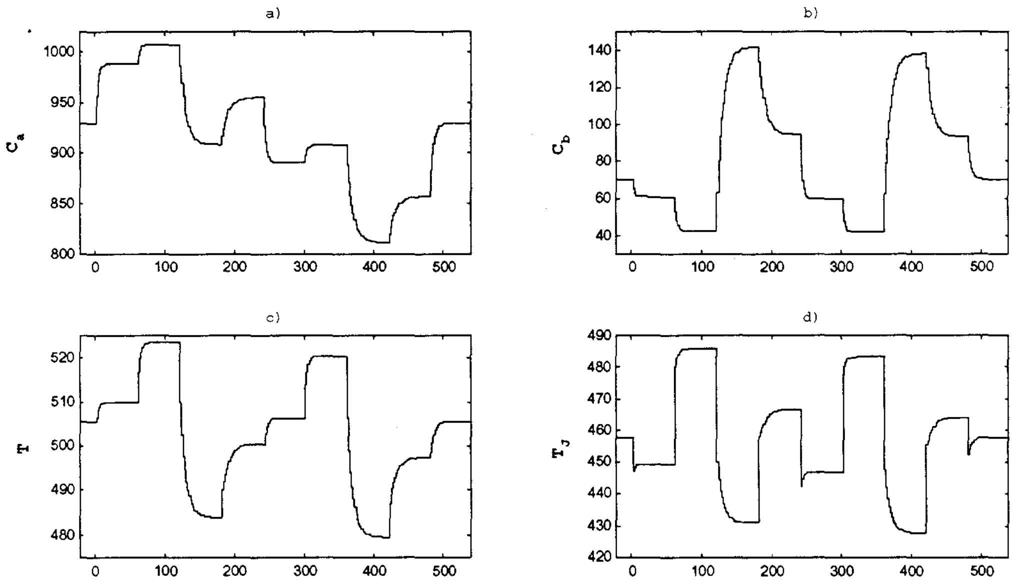

Figure 26, Results for the factorial experiment on the CSTR example.

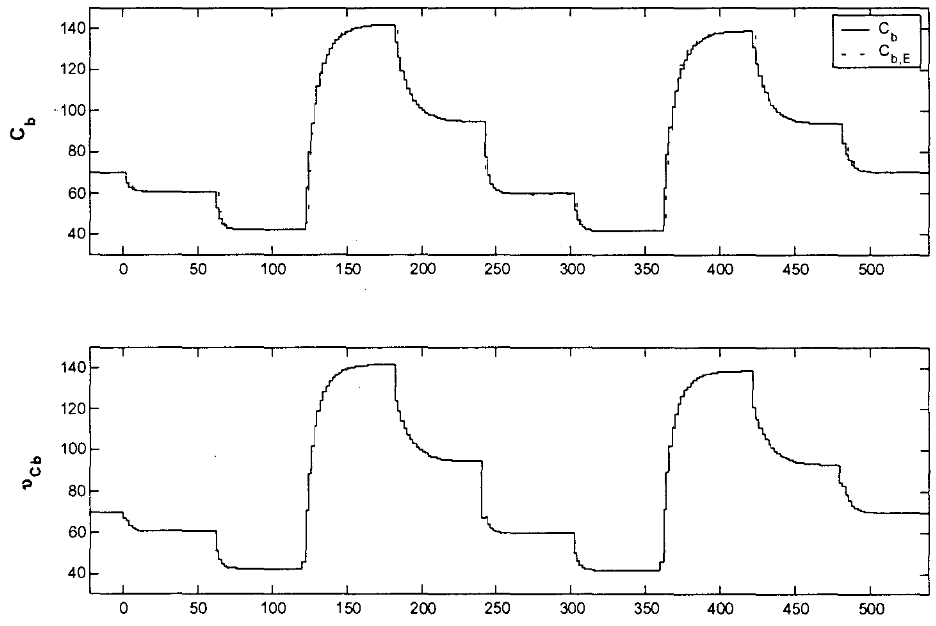

Figure 27, C_b and the transformed input υ_{C_a}, including the estimated product concentration ($C_{b,E}$) using (4-37), for the CSTR factorial experiment.

Thus far the steady state function Ψ_{C_b} has been designed that incorporates the reactant concentration. Empirical testing of the system verified that prediction of C_b with suitable accuracy is possible with the transformed input $\upsilon_{C_b} = \Psi_{C_b}$. Prediction of future C_b values entails estimation of future υ_{C_b} values. This requires knowledge of F, C_{ai}, $k(T)$ and K_e over the prediction horizon. F and C_{ai} are stochastic input variables that can not be manipulated and are treated as disturbances. If knowledge of future changes in these variables is available it can be incorporated into the prediction of υ_{C_b}. Typically future disturbances are not known however. It is assumed here that the disturbances follow a random walk structure, therefore the best estimation of future disturbance values is the present value.

Estimation of $k(T)$ and K_e requires knowledge of the reactor temperature. T will be treated as an intermediate state that is determined prior to estimation of υ_{C_b}. The steady state function for T can be derived from the system model (4-29) using the same method for determination of C_b^{ss}; by setting the temperature derivative to zero and solving for T^{ss} as in (4-38) and (4-39).

$$0 = \frac{F}{V} \cdot \left(T_i - T^{ss}\right) + \frac{(-\Delta H_R)}{\rho \cdot c_p} \cdot r_a + \frac{Q}{V \cdot \rho \cdot c_p} \tag{4-38}$$

$$T^{ss} = T_i + \frac{V \cdot (-\Delta H_R)}{F \cdot \rho \cdot c_p} \cdot k_0 \cdot \exp\left[-\frac{E}{R \cdot T^{ss}}\right] \cdot \left(C_a - \frac{C_b}{K_e}\right) + \frac{Q}{F \cdot \rho \cdot c_p} \tag{4-39}$$

There are a couple of obvious problems with this formulation. First there is a nonlinear T^{ss} term on the right hand side of (4-39). This prevents an algebraic solution for T^{ss}. The second problem is future knowledge of C_a and C_b is required. Since T is an intermediate state used to predict the future behaviour of C_b it is assumed that no knowledge of future C_b values is available. A simple predictive I/O relationship could be formulated for C_a and C_b given changes in the system independent variables but this will just add another layer to the prediction sequence. This would resemble a circular iterative structure in which C_a and C_b values are used to estimate T and T values are used to estimate C_a and C_b. Iteration loops could be continued with a stopping criterion. This solution is a reasonable path but the extra layer(s) of trajectory estimation are not computationally efficient. Observing the test data a reasonably good estimation of T can be found by using the inverse of the coolant and CSTR flows, (4-40).

$$T^{ss} = \frac{g_{Fc}}{F_c} + \frac{g_F}{F} + e_{T^{ss}} \tag{4-40}$$

In (4-40) g_{Fc} and g_F are empirical gain coefficients.

A plot of the actual and predicted CSTR temperature values along with the inverse flows is shown in Figure 28. The discrete linear predictive model with a sampling period of 2 is (4-41) and the resulting gains are $g_{F_c} = 27.0$ and $g_F = -17.5$.

$$\begin{aligned}\hat{T} = {} & \frac{12.4 - 20.89 \cdot z^{-1} + 8.488 \cdot z^{-2}}{1 - 2.058 \cdot z^{-1} + 1.144 \cdot z^{-2} + 0.02843 \cdot z^{-3} - 0.1153 \cdot z^{-4}} \cdot \upsilon_{T,1} \\ & + \frac{-5.533 + 7.599 \cdot z^{-1} - 1.132 \cdot z^{-2} - 0.9321 \cdot z^{-3}}{1 - 2.058 \cdot z^{-1} + 1.144 \cdot z^{-2} + 0.02843 \cdot z^{-3} - 0.1153 \cdot z^{-4}} \cdot \upsilon_{T,2} + e_{T^{ss}}\end{aligned} \tag{4-41}$$

where: $\upsilon_{T,1}[k] = \frac{1}{F_c[k]}$

$$\upsilon_{T,2}[k] = \frac{1}{F[k]}$$

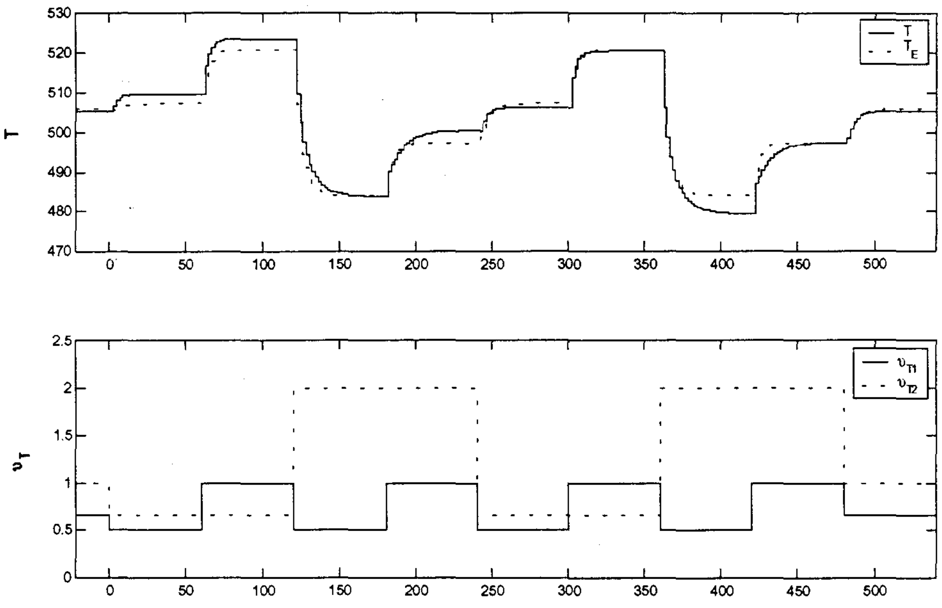

Figure 28, Actual and predicted CSTR temperatures (T and T_E), and the inverse coolant and CSTR flows, $1/F_c$ (υ_{T1}) and $1/F$ (υ_{T2}), for the CSTR factorial experiment.

The intermediate prediction of T is important for a number of reasons. First it increases the predictive ability of the controller. It also provides a path towards including the manipulated variable F_c in the predictive algorithm.

With these models an ITPC controller can be designed. The sequence used to predict the future C_b trajectory is implemented as follows.

i) Determine the predicted T trajectory from past changes in the transformed input variables.

$$T_h = \begin{bmatrix} 27.0231 \cdot \hat{A}_{T,1h} & -17.5247 \cdot \hat{A}_{T,2h} \end{bmatrix} \cdot \begin{bmatrix} \nabla \upsilon_{T,1h} \\ \nabla \upsilon_{T,2h} \end{bmatrix} \tag{4-42}$$

$\hat{A}_{T,1h}$ and $\hat{A}_{T,2h}$ are both of size $P \times P$ and are designed using (4-41). $\nabla\upsilon_{T,h}$ is a vector of length $2 \cdot P$. The ITPC controller in this example has a prediction horizon of 40 and a sample time of 2 ($P = 20$).

ii) Determine the predicted C_b trajectory from past changes in the transformed input υ_{C_b}.

$$C_{b,h} = 1 \cdot \hat{A}_{C_b h} \cdot \nabla\upsilon_{C_b h} \tag{4-43}$$

$\hat{A}_{C_b h}$ is a $P \times P$ matrix designed from (4-37) and $\nabla\upsilon_{C_b h}$ is of length P. Historical values for $\nabla\upsilon_{C_b}$ contain all known quantities. This is significant because it means that an accurate estimation of $C_{b,h}$ is available regardless of the ability to estimate the intermediate state T trajectory.

iii) Determine the future input transformation vectors $\nabla\upsilon_{T,1f}$ and $\nabla\upsilon_{T,1f}$. For a control horizon of 10 ($C = 5$) these vectors are;

$$\nabla\upsilon_{T,1f} = \begin{bmatrix} \upsilon_{T,1[k]} - \upsilon_{T,1[k-1]} \\ \upsilon_{T,1[k+1]} - \upsilon_{T,1[k]} \\ \upsilon_{T,1[k+2]} - \upsilon_{T,1[k+1]} \\ \upsilon_{T,1[k+3]} - \upsilon_{T,1[k+2]} \\ \upsilon_{T,1[k+4]} - \upsilon_{T,1[k+3]} \end{bmatrix} \qquad \nabla\upsilon_{T,2f} = \left\lfloor \upsilon_{T,2[k]} - \upsilon_{T,2[k-1]} \right\rfloor \tag{4-44}$$

$\nabla\upsilon_{T,2f}$ is only of length 1 as a result of the random walk assumption of the stochastic variable F.

iv) Calculate the trajectory of T for past and future transformed input changes using (4-45).

$$T_p = T_h + T_f + T_{[k]} \qquad \text{where: } T_f = \begin{bmatrix} 27.0231 \cdot \hat{A}_{T,1f} & -17.5247 \cdot \hat{A}_{T,2f} \end{bmatrix} \cdot \begin{bmatrix} \nabla\upsilon_{T,1f} \\ \nabla\upsilon_{T,2f} \end{bmatrix} \tag{4-45}$$

$T_{[k]}$ is the present measured temperature. $\hat{A}_{T,1f}$ and $\hat{A}_{T,2f}$ are matrices of size $P \times C$ and $P \times 1$ respectively and are designed using (4-41).

v) Estimate future υ_{C_b} values given the temperature trajectory from iv) and the future changes in F_C.

$$\nabla\upsilon_{C_b f} = \begin{bmatrix} \upsilon_{C_b}[k] - \upsilon_{C_b}[k-1] \\ \upsilon_{C_b}[k+1] - \upsilon_{C_b}[k] \\ \upsilon_{C_b}[k+2] - \upsilon_{C_b}[k+1] \\ \vdots \\ \upsilon_{C_b}[k+P-1] - \upsilon_{C_b}[k+P-2] \end{bmatrix} \tag{4-46}$$

vi) Calculate the predicted C_b trajectory using $\nabla\upsilon_{C_b f}$ from v), $C_{b,h}$ from ii) and the present measured value of C_b.

$$C_{b,p} = C_{b,h} + C_{b,f} + C_{b[k]} \qquad \text{where: } C_{b,f} = 1 \cdot \hat{A}_{C_b f} \cdot \nabla\upsilon_{C_b f} \tag{4-47}$$

$\hat{A}_{C_b f}$ is a $P \times P$ matrix designed from (4-37).

vii) Reset the $\nabla\upsilon_{C_b h}$, $\nabla\upsilon_{T,1h}$ and $\nabla\upsilon_{T,2h}$ vectors.

$$\nabla\upsilon_{C_b h} = \begin{bmatrix} \upsilon_{C_b}[k] - \upsilon_{C_b}[k-1] \\ \upsilon_{C_b}[k-1] - \upsilon_{C_b}[k-2] \\ \upsilon_{C_b}[k-2] - \upsilon_{C_b}[k-3] \\ \vdots \\ \upsilon_{C_b}[k-P+1] - \upsilon_{C_b}[k-P] \end{bmatrix} \tag{4-48}$$

$$\nabla \upsilon_{T,1h} = \begin{bmatrix} \upsilon_{T,1}[k] - \upsilon_{T,1}[k-1] \\ \upsilon_{T,1}[k-1] - \upsilon_{T,1}[k-2] \\ \upsilon_{T,1}[k-2] - \upsilon_{T,1}[k-3] \\ \vdots \\ \upsilon_{T,1}[k-P+1] - \upsilon_{T,1}[k-P] \end{bmatrix} \qquad \nabla \upsilon_{T,2h} = \begin{bmatrix} \upsilon_{T,2}[k] - \upsilon_{T,2}[k-1] \\ \upsilon_{T,2}[k-1] - \upsilon_{T,2}[k-2] \\ \upsilon_{T,2}[k-2] - \upsilon_{T,2}[k-3] \\ \vdots \\ \upsilon_{T,2}[k-P+1] - \upsilon_{T,2}[k-P] \end{bmatrix}$$

Steps i), ii) and vii) are completed only once at each control exectution. Steps iii) through vi) are included in an iterative optimization algorithm that minimizes the objective function (4-30).

Simulation results of the ITPC control for a set of known disturbances is shown in Figure 29. Included in the figure is the maximum possible product concentration ($C_{b,max}$) given the disturbance values. Comparison of $C_{b,max}$ and the actual C_b resulting from the ITPC manipulation of F_c displays the controller's ability to achieve the objective for this nonlinear system. This a particularly difficult task to ask of the controller since the objective requires that the system be maintained around the operating point where the input output relationship is singular. The disturbances for this simulation are included in Figure 30. The tuning of the controller is shown in (4-49).

$$W_y = [1 \quad 1 \quad \cdots \quad 1] \qquad W_u = I_C \cdot 10 \qquad \text{where: } I_c \text{ is an identity matrix of size } C \times C \qquad \textbf{(4-49)}$$

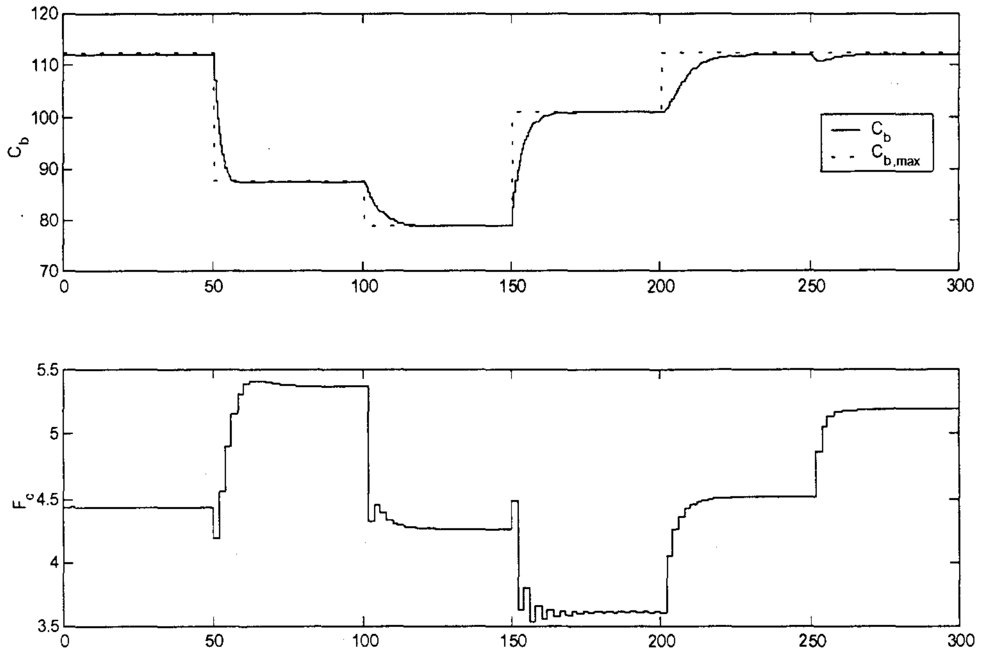

Figure 29, Simulation results for the ITPC controller from section 4.3 on the CSTR example for a set of known disturbances.

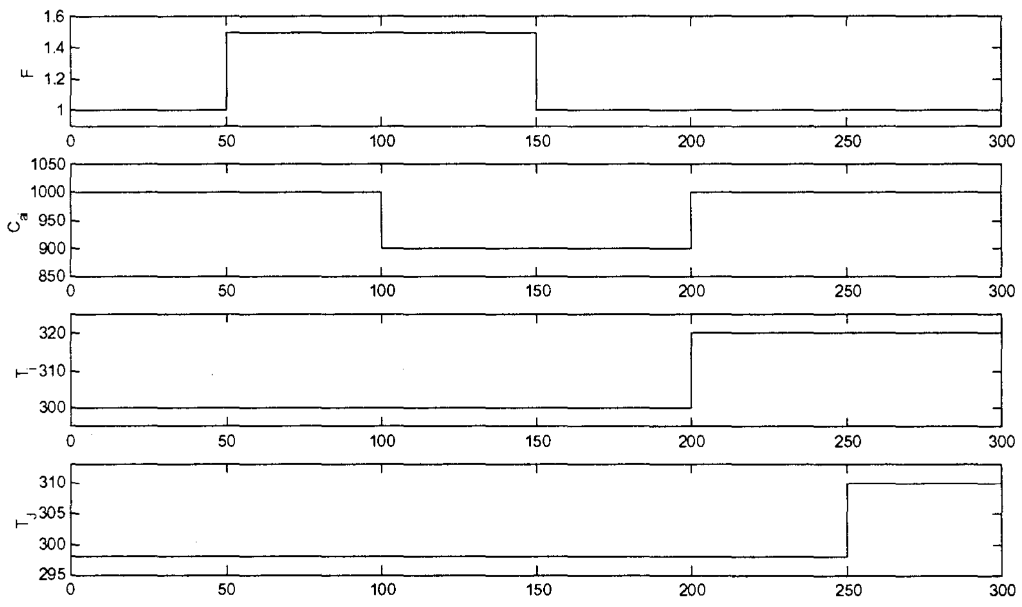

Figure 30, Known disturbances for the ITPC simulation.

Implementation of the ITPC control law on the system necessitated one modification from the design as shown. This modification was required due to the sensitivity of the system to overcooling. The plot of C_b verses F_c and T verses F_c in Figure 31 displays this sensitivity.

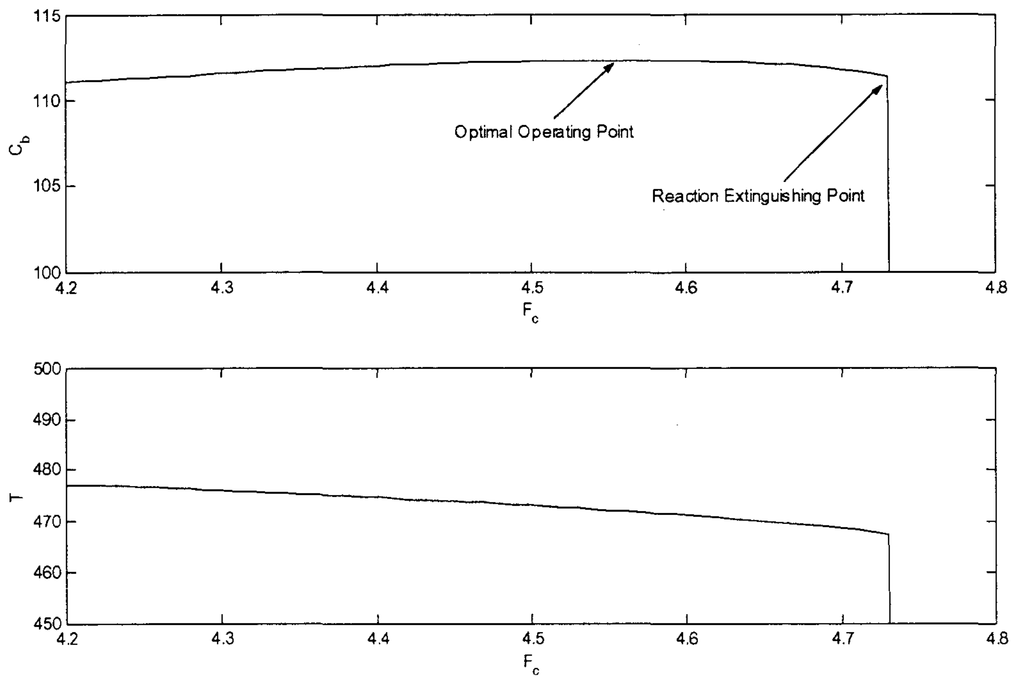

Figure 31, Sensitivity of the CSTR system to overcooling.

Figure 31 shows that small changes in coolant flow create large changes in the product concentration and CSTR temperature. This abrupt change in operation results from an extinguishing of the reaction. The sensitivity exists inconveniently close to the optimal operation point. If fact, there is only a 0.17 difference in coolant flow between the optimal operating point and the reaction extinguishing point. Operation too close to the extinguishing temperature is not robust as disturbances can extinguish the reaction before the controller can reject their action through manipulation of F_c. A temperature bias term is introduced that forces the ITPC controller to operate the CSTR slightly warmer than the optimal temperature to increase robustness. The bias term is incorporated by setting the temperature that is visible to the controller to the measured temperature minus the bias, (4-50).

$$T = T_m - T_\delta \qquad \textbf{(4-50)}$$

T_m is the measured reactor temperature and T_δ is the temperature bias. The simulation from Figure 29 uses a bias of 2.

It is important to observe and understand that the controller is capable of predicting the system behaviour outside of the testing region of operation. The factorial experiment used to identify the empirical models in (4-37) and (4-41) included coolant flow changes from 1.0 to 2.0. Over this range the CSTR temperature varied from 523 to 480. These levels were consciously chosen to assure operation on the hot side of the optimal temperature to avoid extinguishing the reaction. The intent of the controller however is to operate the system at the optimal temperature to maximize C_b. The ITPC controller is therefore expected to maintain the plant outside the operating region the system models are identified. In general empirical models do not extrapolate well for nonlinear systems. The models in ITPC really consist of two distinct submodels. The first submodel contains the system steady state I/O map and the second contains dynamic information. The steady state map describes the operating point a system will move to given a change in inputs and the dynamic model describes how it will get there. Even though the dynamic model is linear the introduction of intermediate states allows for the retention of nonlinear dynamics as eluded to in the first example system. It is this separation of steady state and dynamic information that allows a nonlinear system to be represented through elementary linear matrix algebra. It also provides the user with increased flexibility for the development of the controller models.

In the case of this CSTR example the steady state map is designed from a perfect mechanistic model. Therefore the ITPC controller is capable of determining the exact CSTR temperature that maximizes C_b as demonstrated in Figure 29. Empirical models are used to estimate the coolant flow required to adjust the CSTR temperature and the system dynamics. As a result the controller does not know the exact coolant flow required to achieve the desired CSTR temperature or the path the system will take to move from one operating point to another. With respect to the controller objective and implicit performance requirements feedback can be used to correct for model mismatch between (4-41) and the real plant for all regions of operation. This appropriate coupling of mechanistic and empirical information enables suitable control to be achieved outside the range of empirical model identification.

Understanding the information used to design the models of the ITPC controller are essential to assess its capabilities and limitation. The dual of this statement is; the control objective defines the required capabilities of a controller and hence the type of information needed for the controller design. For instance if the objective of the CSTR controller was to maintain the reactor temperature around a desired setpoint or the rejection of high frequency disturbances the structure, required information and capabilities would no doubt be very different from the design presented. These statements are valid for any control technology but the ITPC framework provides more opportunity for their application via the separation of the steady state and dynamic descriptions.

Implementation of control technologies such as error trajectory and QDMC are infeasible for this application. The objective for error trajectory controllers does not provide the flexibility to include optimization variables, meaning that system conditions can neither be minimized nor maximized. The error trajectory objective is only capable of regulating process variables around a desired point. Linear control structures like QDMC are very powerful for linear systems and can be successfully applied to mildly nonlinear systems but are not effective for controlling systems in which the process gains change sign such as in the CSTR. A NL-MPC structure could be applied with similar results to the ITPC controller but with increased computational demands. The local linearization techniques described in section 2.3 are also useful for control of this system. Simulation of a local linear predictive controller with the same execution frequency, disturbances, P, C and temperature bias as the ITPC controller is shown in Figure 32. The local linear controller however was found to be more difficult to tune and became unstable without large input move suppression weights, ($W_u = I_C \cdot 200$). This results from the inability of the local linear controller to recognize future crossings of the singular point. This is also evident in the chattering of the coolant flow for times 150 through 200.

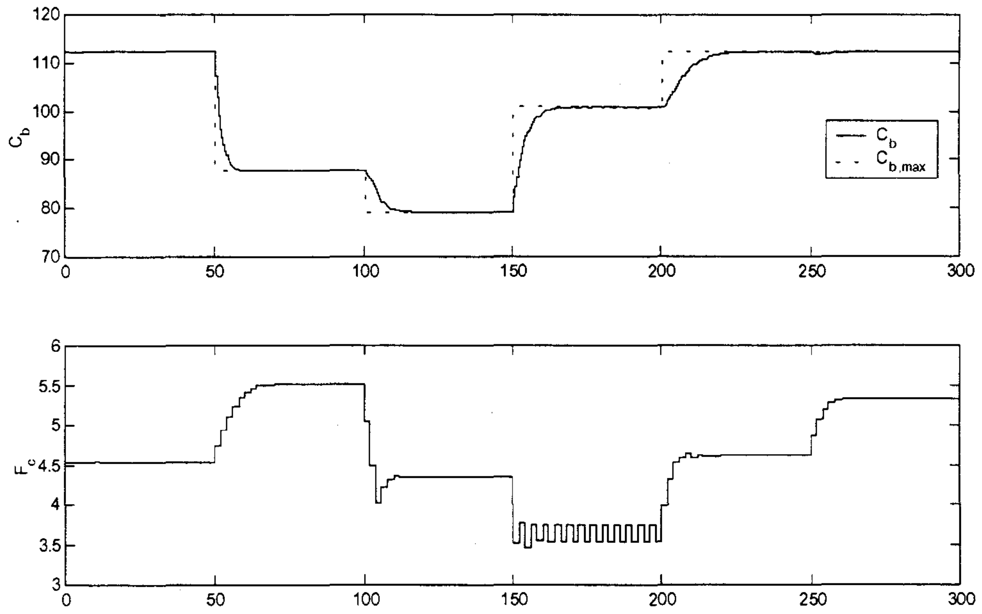

Figure 32, Simulation results for the local linear predictive controller from section 2.3 on the CSTR example system for a set of known disturbances.

This CSTR example system displays some additional properties of the ITPC control structure over and above those investigated in the first example system. The formulation of the objective for the CSTR is very different. The CSTR control problem is really an optimization problem whereas the objective of the first example was servo control around a desired operating point. The suitability of ITPC for both control problems stems from the open structure of its objective function. Inclusion of economic or optimization variables and various constraints in the objective is seamless. The CSTR example also provided some insight regarding the marriage of mechanistic and empirical system information. Most importantly the example demonstrated the separation of the steady state and dynamic system descriptions and how appropriate attention should be directed towards the design and identification of system models depending on the intent of the controller.

5.0 Discussion of Controller Performance

This section contains a direct comparison of the different control approaches described within, highlighting the strengths and weaknesses of each. The properties that are investigated include, the ability to execute setpoint changes, the ability to maintain a system at of over a singular point, computational burden, system constraint handling, the objective function structure and development effort. In previous sections specific issues of controller performance are discussed. This section begins by summarizing the previous results with a broader scope of analysis. New results are introduced in the following subsections in an expanded discussion of servo ability and singular point operation.

The first example system (3-15) was used to display the design procedure for a number of discrete error trajectory control structures and an ITPC controller. Each of the discrete error trajectory controllers presented have either a sampled or integrated error trajectory objective. The sampled version satisfies the error trajectory objective at the next controller execution and the integrated objective minimizes the distance from the error manifold over the controller execution interval. It was found that satisfaction of the error trajectory objective at discrete sampling intervals did not generally result in acceptable performance. This was true for both the sampled error trajectory designs and discrete implementation of the continuous error trajectory control law. The unexpectedly poor performance was explained to result from the inappropriateness of the error trajectory objective in the discrete domain. Taylor series expansion was used to quantify the effects of discretization of the continuous error trajectory objective when applied to sampled systems. The analysis showed that discretization effects arise when the controller sample time is significant with respect to system dynamics.

An input transformation predictive control structure was described. The ITPC structure contains nonlinear predictive abilities while retaining many of the computational properties of linear predictive control technologies. The ITPC structure was explained for both constrained and unconstrained applications including an explicit solution. It was shown that the ITPC prediction method, which consists of only simple linear matrix algebra procedures, is capable of retaining both steady state and dynamic system characteristics. Implementation of the controller on the example system (3-15), displayed the ITPC performance on a nonlinear system. Comparatively the ITPC controller performed as well or better than all forms of the discrete error trajectory controller.

A second example was introduced that displayed some additional properties of ITPC. The second example consisted of an adiabatic CSTR with a reversible exothermic reaction. The controller objective was to maximize the product concentration through temperature management. The temperature was maintained by adjustment of a coolant flow. Of particular concern, the system contained an inversion of the gain between the coolant flow and the product concentration. It was shown that the ITPC controller was able to bring the CSTR to its theoretical optimal operating point for a number of measurable disturbances. An interesting property of this control problem is, the goal of the controller is to maintain the system at a singular point where the gain between the coolant flow and the product concentration is zero. In addition to the ITPC controller a local linearized controller was simulated. It was found that very large move suppression weights were required for stable operation and chattering was observed around the optimal operating point. The difficulties experienced with the local linearized controller were attributed to its inability to predict future crossings of the singular point.

The application of a linear control technology or error trajectory controller on the exothermic reversible CSTR example is not feasible. The models used in linear controllers render them incapable of determining the optimal reactor temperature that maximizes the product concentration. The error trajectory objective is not condusive for the inclusion of optimization variables since there is no real error to speak of. Error trajectory control could be used to maintain the reactor temperature or product concentration around a desired value or trajectory but is not the appropriate technique for optimization applications.

Qualitatively performance of each of the aforementioned controllers was correlated to the design effort and computational demands. In general, the greater the design effort and computational burden the greater the performance. The error trajectory control method was found to be very good at controlling nonlinear systems when the controller execution frequency is fast compared to the system dynamics. The sampled error trajectory controllers presented within reduced the effects of discretization in the sense that the discrete objective was satisfied but the resulting performance was not as intended. The discrete integrated error trajectory control formulation showed the most promise with respect to performance but are as computationally expensive as nonlinear predictive control strategies, requiring the solution of a NLP. Predictive strategies however possess advantages over error trajectory methods, such as, effective handling of ill conditioned regions of operation, constraint handling and a flexible objective function that seamlessly allows for the incorporation of additional objectives such as economic parameters. An optimal error trajectory controller was presented that provides a solution to many of these problem areas. The performance of the optimal controller was acceptable but the method still suffers from yet other limitations of the error trajectory objective including, the inability for application on nonminimum phase systems and the difficulty of application on systems that can not be represented in the control affine representation, (2-12).

The servo changes used in the first example system effectively identified control approaches that were good or poor candidates for controlling the system. The next two subsections present results for a new set of setpoint changes to gain further insight into the capability of the control methods for a broader set of nonlinear control issues. In addition results are shown for control technologies not investigated in the previous analysis such as local linear and liner predictive techniques. The first setpoint change displays the ability of the controllers to bring the system from one operating point to another. This is similar to the previous analysis except that the magnitude of the change is increased. The second setpoint change forces the controllers to cross a singular point in the system, including an inverse in process gains.

5.1 Process Operating Point Changes

The first experiment used as a basis for expanded comparison of control technologies is a setpoint change in y_2 from 2 to 10. This is similar to one of the setpoint changes used in the first set of setpoint changes that included a change from 2 to 4. The difference between these two setpoint changes being the severity of plant nonlinearity the controller must drive the system through. This refers to the degree of change in the system's Jacobian from one operating point to another. The values of the Jacobians for the three associated operation points are shown in (5-1).

$$\begin{bmatrix} 5 & 0.5 \\ 2 & 0 \end{bmatrix} \qquad \text{for: } y_1 = 5, \quad y_2 = 2 \qquad \textbf{(5-1a)}$$

$$\begin{bmatrix} 2.5 & 1 \\ 2 & 0 \end{bmatrix} \qquad \text{for: } y_1 = 5, \quad y_2 = 4 \qquad \textbf{(5-1b)}$$

$$\begin{bmatrix} 1 & 2.5 \\ 2 & 0 \end{bmatrix} \qquad \text{for: } y_1 = 5, \quad y_2 = 10 \qquad \textbf{(5-1c)}$$

From (5-1) a setpoint change in y_2 from 2 to 4 requires that the controller move the process over a region in which the steady state gain between u_1 and y_1 decreases by half and the gain between u_2 and y_1 doubles. For a setpoint change of 2 to 10 these same gains are changed by a factor of 5 rather than 2. Expressing the open loop nonlinearity as the relative change in the Jacobian, the degree of open loop nonlinearity for a setpoint change in y_2 of 2 to 10 is 2.5 times greater than that for a setpoint change of 2 to 4. Use of a linear controller clearly displays the increase in difficulty of the large verses small setpoint changes. Figure 33 displays the closed loop response of the example system to a large and small step increase in y_2. In both cases the linear controller is able to move y_2 from 2 to its final setpoint. The controller's ability to maintain y_1 at its setpoint during these changes in y_2 is inadequate in both cases but especially poor for the change from 2 to 10.

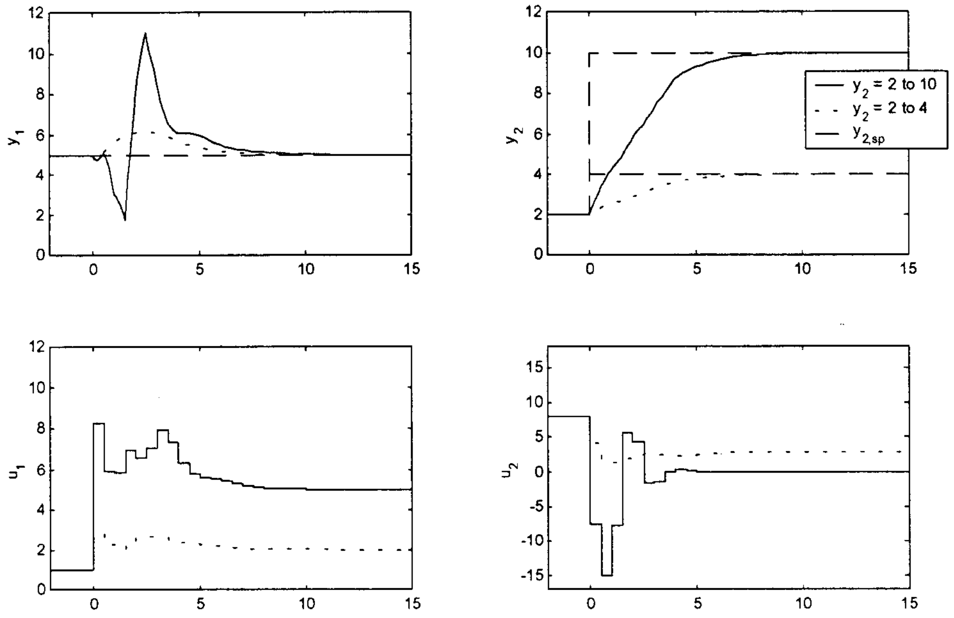

Figure 33, Closed loop response for a small and large setpoint change in y_2 from the example system (3-15), **using the linear QDMC controller from section 4.1 calibrated around $u_1 = 1$ and $u_2 = 1$ with a sample time, $T_s = 0.5$.**

Results for the optimal error trajectory and the ITPC controllers are shown in Figure 34 and Figure 35. These nonlinear control technologies are able to execute the setpoint changes without large deviations in y_1. The performance of these controllers is similar and neither is distinctly better than the other. The maximum deviation of y_1 from its setpoint is smaller for the optimal error trajectory controller however the ITPC controller has a smaller mean squared y_1 error and does not overshoot the y_2 setpoint.

Qualitatively the ITPC controller appears to settle the system faster than the optimal error trajectory and requires smaller input changes.

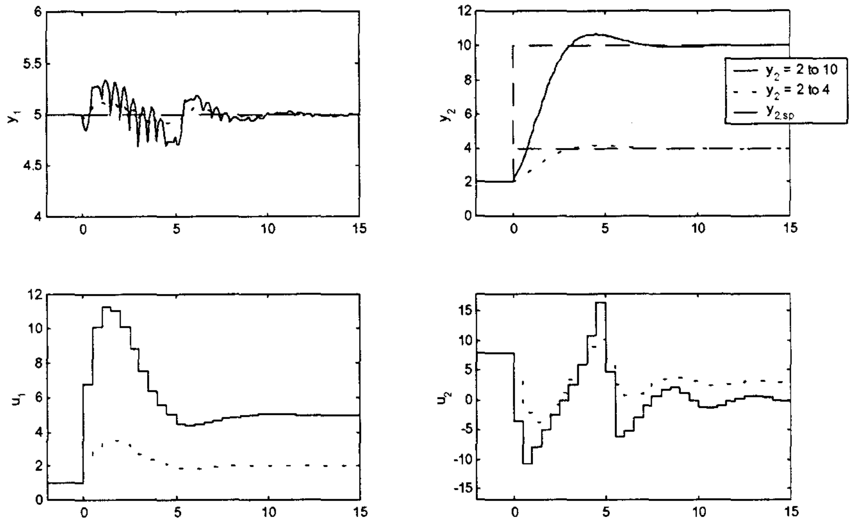

Figure 34, Closed loop response for a small and large setpoint change in y_2 from the example system (3-15), using the discrete optimal error trajectory controller (3-31) with a sample time, $T_s = 0.5$.

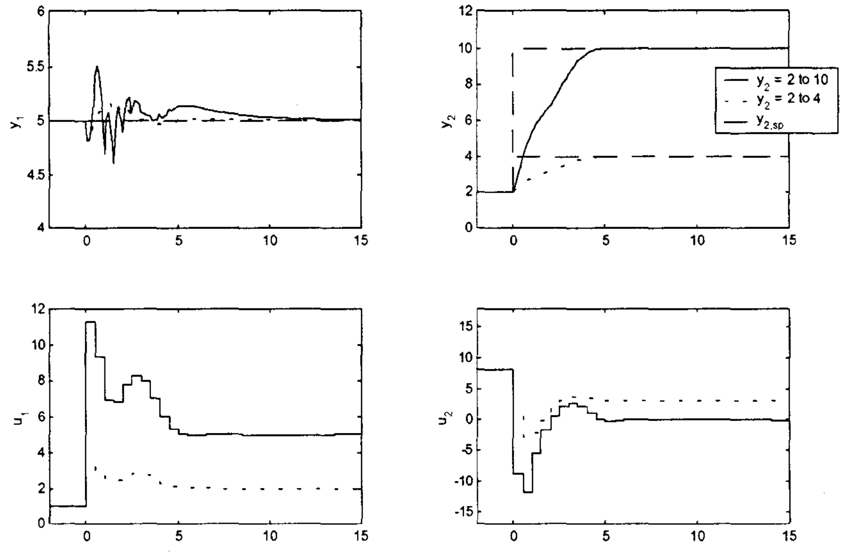

Figure 35, Closed loop response for a small and large setpoint change in y_2 from the example system (3-15), using the ITPC controller from section 4.2 with a sample time, $T_s = 0.5$.

During the development of the local linearization predictive controller in section 2.3 it is mentioned that one of its shortcomings is predictive ability as the process moves from one operating point to another. Implementation of the local linerized control method on the example system is shown in Figure 36. The technique is able to successfully execute the setpoint change of y_2 from 2 to 4 without large deviations in y_1, but is not as effective in performing the large setpoint change. Large deviations in y_1 are experienced and there is a significant overshoot in the y_2 setpoint. Comparison of the local linearized controller to ITPC performance demonstrates effect of the diminished predictive ability on closed loop response during operating point transitions.

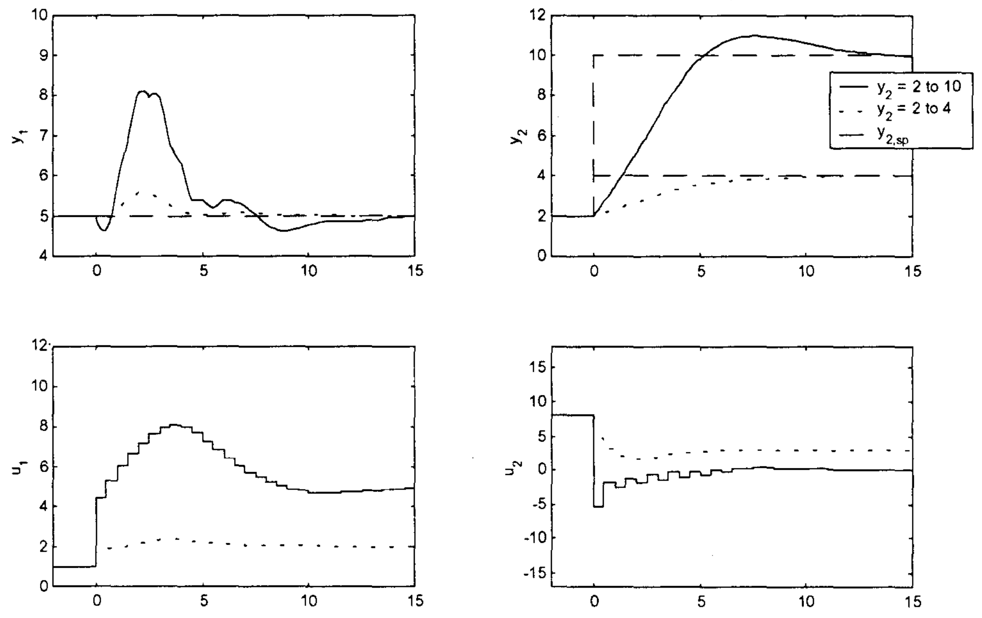

Figure 36, Closed loop response for a small and large setpoint change in y_2 from the example system (3-15), using the local linear predictive controller from section 2.3 with a sample time, $T_s = 0.5$)

5.2 Singular Point Crossings

Forcing a system to cross a singular point is a difficult task to demand of a controller. A singular point represents a position where the system is no longer controllable as the gain between an input and output becomes zero therefore its inverse does not exist. With respect to the continuous error trajectory controller (2-14), the control law *blows up* at the singular point, as its denominator equals zero ($L_g L_f^{\alpha-1} h_s(x^\sigma) = 0$). In the discrete domain the system crosses the singular point at some instant but often does not remain identically singular over the complete sampling interval therefore the issue is not so much how to deal with an uncontrollable system but rather how to handle sign changes in process gains and suppression of large input moves. Penalizing input changes through weighting of input moves in the controller objective as in DMC is an accepted means for reducing ill conditioning. In general, model based controllers tend to handle sign changes in process gains provided the system model is capable of detecting the sign change. Both the optimal error trajectory and ITPC controllers include accurate nonlinear system descriptions and input weighting in their objectives therefore they have all the necessary tools for acceptable handling of singular point crossings.

Singular point crossings for the ITPC and local linear controllers were demonstrated in the CSTR example problem. It was found that both controllers were able to satisfy the control objective that demanded operation around a singular point. Difficulties were encountered for the local linear controller as it required very large move suppression weights to achieve stability and chattering was observed. This was attributed to the use of a linear model for the predicting the effect of future control changes. The linear prediction model rendered the controller unable of detecting future crossings of the singular point from future control changes.

The structure of the error trajectory controller objective is not applicable for the CSTR control problem, which is closer to an economic optimizer than a regulatory controller. Singular point crossing for the error trajectory controller is demonstrated using the first example system with a setpoint change in y_2 from 2 to -2. Performance of three error trajectory formulations is shown in Figure 37, Figure 38 and Figure 39. The three formulations include sampled implementation of the continuous controller, the exact discrete error trajectory controller and the optimal error trajectory controller. These forms are representative of all the error trajectory controllers described. Adhering to the general rule that performance is correlated to the controller design effort and computational complexity, the exact discrete error trajectory controller performed better than the discrete implementation of the continuous error trajectory controller and the optimal error trajectory controller performance was better than the exact discrete error trajectory controller. Evident from the simulations the discrete implementation of the continuous error trajectory objective and the exact discrete error trajectory controller do not perform as intended, further indicating the inappropriateness of satisfaction of the error trajectory objective at discrete intervals. The performance of the optimal error trajectory is as intended. Of note the optimal error trajectory controller with no input move suppression, which essentially simplifies to the integrated error trajectory controller does not perform well. In this case input weighting is needed to move the system through the ill conditioned singular region.

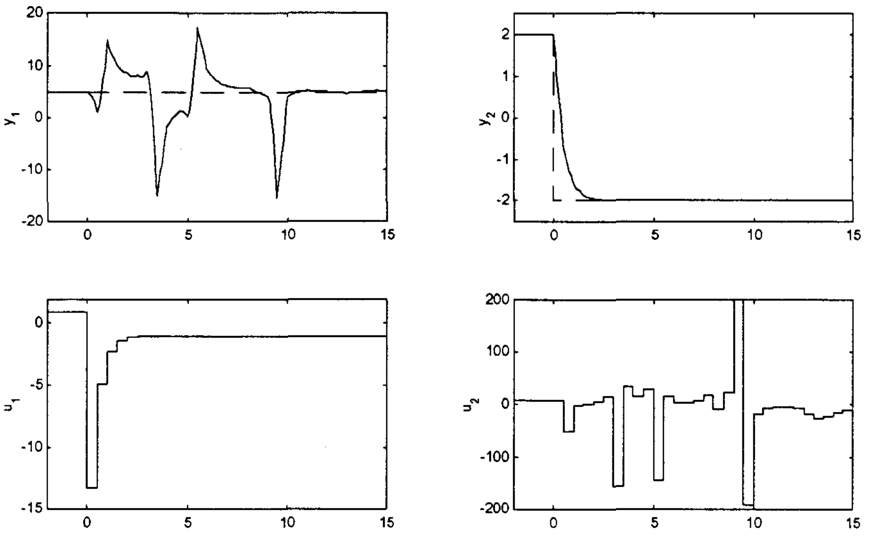

Figure 37, Closed loop response for a setpoint change in y_2 from the example system (3-15) that forces the system to cross a singular point using the discrete implementation of the continuous error trajectory controller (3-19) with a sample time, $T_s = 0.5$.

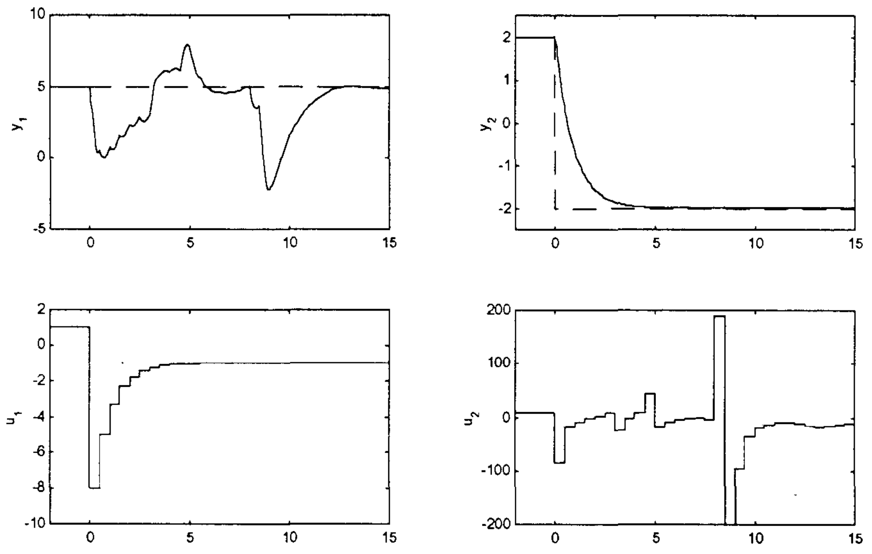

Figure 38, Closed loop response for a setpoint change in y_2 from the example system (3-15) that forces the system to cross a singular point using the discrete exact error trajectory controller (3-20) with a sample time, $T_s = 0.5$.

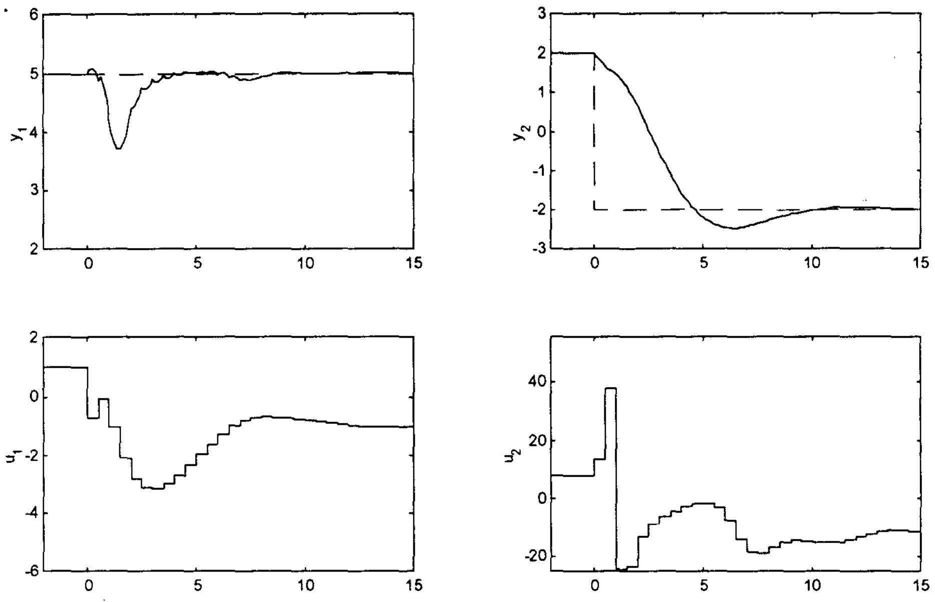

Figure 39, Closed loop response for a setpoint change in y_2 from the example system (3-15) **that forces the system to cross a singular point using the optimal error trajectory controller** (3-31) **with a sample time, $T_s = 0.5$.**

Application of the ITPC controller for the same setpoint change implemented on the error trajectory controllers is shown in Figure 40. The maximum deviation of y_1 from its setpoint is smaller for the optimal error trajectory controller yet the ITPC controller tends to settle the system faster and does not overshoot the y_2 setpoint. As in the operating point setpoint changes there is not a clear cut winner between the optimal error trajectory controller and ITPC. Simulation of the general nonlinear predictive control structure from section 2.2 is shown in Figure 41. The performance is similar to ITPC. The slight difference in performance, observed as the system crosses a singular point, is attributed to the zero order hold structure of the continuous intermediate state, x_1. The singular point of the system is at $x_1 = 0$. As x_1 approaches zero the gain between u_2 and y_1 also approaches zero. Consequently prediction of the system response around the singular point is highly dependent on knowledge of x_1. The zero order hold structure of the intermediate state prediction simplifies the continuous dynamics of x_1. At all other regions of operation the simplification results in relatively small prediction errors but around the singular point the error is magnified. This decrease in the predictive ability of the ITPC method around the singular point is a characteristic of the example system and is not necessarily true in a general sense.

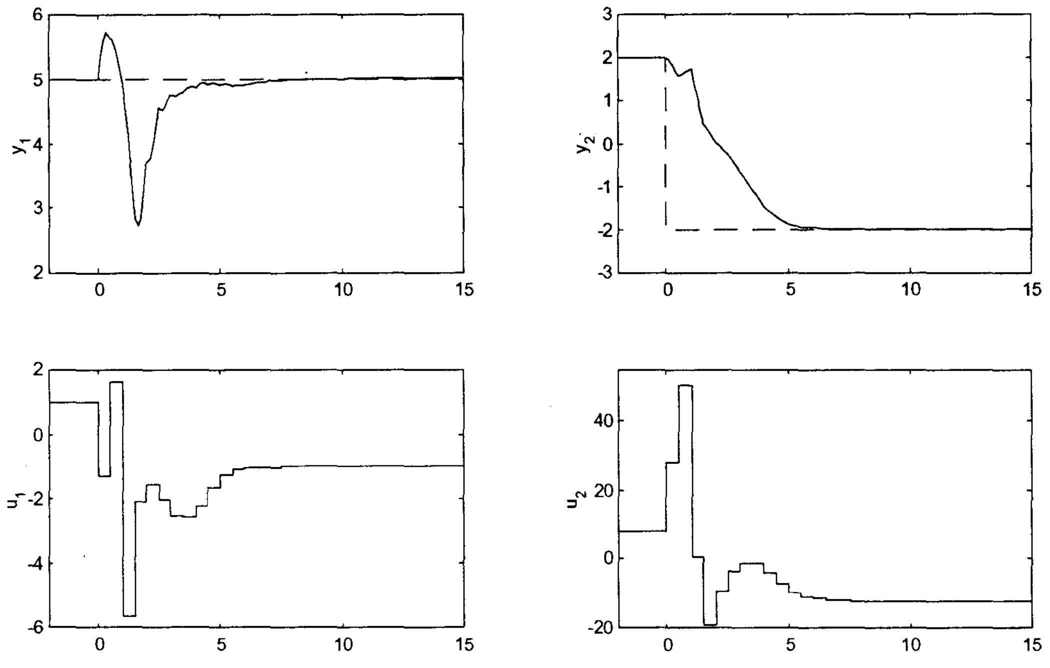

Figure 40, Closed loop response for a setpoint change in y_2 from the example system (3-15) that forces the system to cross a singular point using the ITPC controller from section 4.2 with a sample time, $T_s = 0.5$.

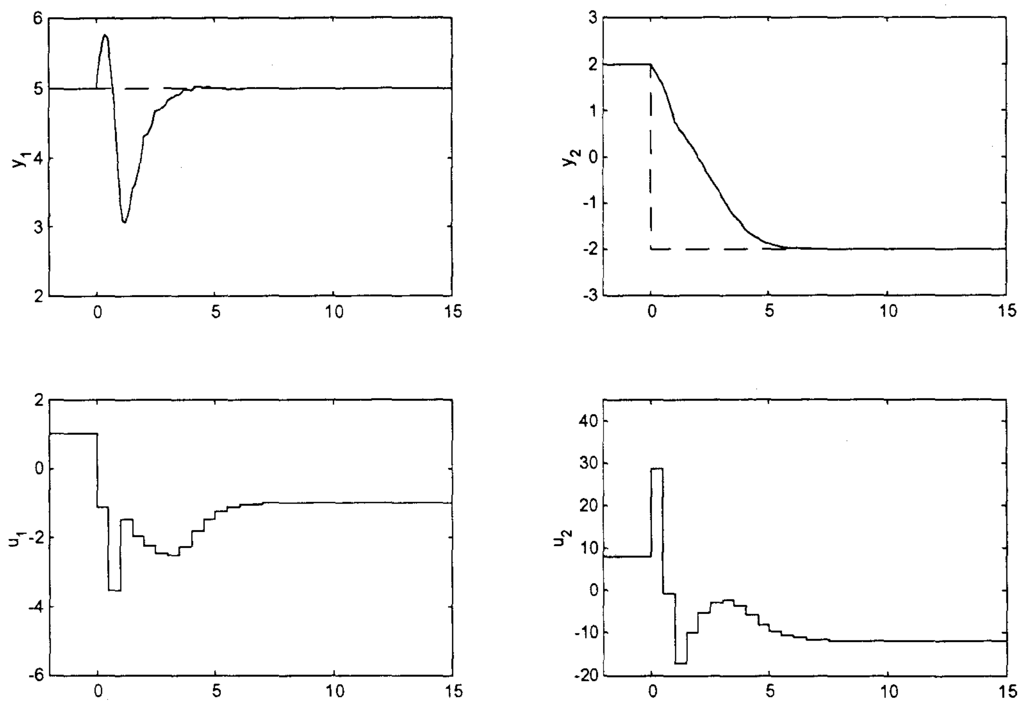

Figure 41, Closed loop response for a setpoint change in y_2 from the example system (3-15) **that forces the system to cross a singular point using the nonlinear predictive controller from section 2.2 with a sample time, T_s = 0.5.**

The local linear predictive controller uses a linear model for prediction of the effect of future control input changes. The technique demonstrated its ability to maintain the example CSTR at the optimal operating point, which is also a singular point. Application of the local linearization controller on the first example system for a crossing of the singular point is not as successful, as indicated in Figure 42. The controller is not only incapable of maintaining y_1 at its constant setpoint, it is also unable to execute the servo change in y_2. This results from the local linear controller's inability to appropriately describe the system over the singular point. Rather than the linearization technique described in section 2.3, more sophisticated model development techniques could be used to derive the linear system model but that is not of interest here. It is sufficient to note that local linearization control techniques contain a simplified prediction model that may not be suitable for control of systems over singular points of operation.

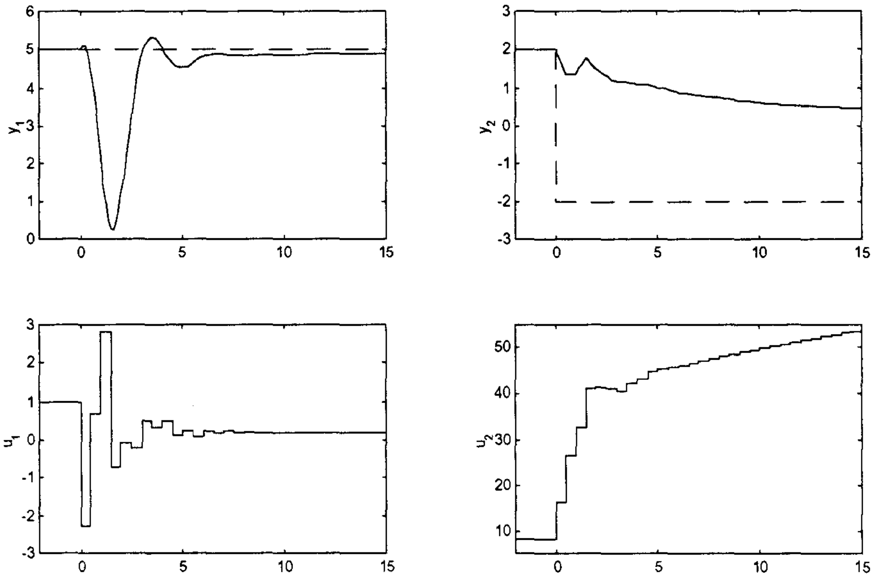

Figure 42, Closed loop response for a setpoint change in y_2 from the example system (3-15) that forces the system to cross a singular point using the local linearizing predictive controller from section 2.3 with a sample time, $T_s = 0.5$.

Finally a linear predictive QDMC controller is used to attempt the setpoint change. It is included just to make the obvious point that linear control technologies are not appropriate for control of system in which the process gain(s) change sign. Figure 43 displays the closed loop response of the system for the linear controller calibrated around the point $u_1 = 1$ and $u_2 = 1$. With the linear controller the system becomes unstable after crossing the singular point.

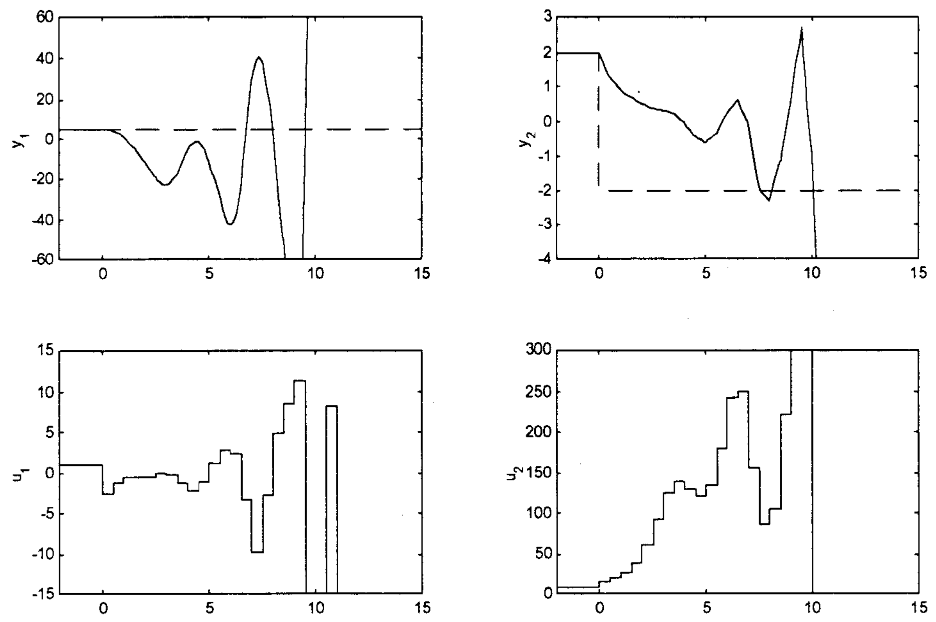

Figure 43, Closed loop response for a setpoint change in y_2 from the example system (3-15) that forces the system to cross a singular point using the linear QDMC controller from section 4.1 with a sample time, $T_s = 0.5$.

5.3 Performance Summary

An input transformation predictive controller and many discrete error trajectory control formulations are presented within. Two example systems are used to describe the design procedures and test the performance of each controller. Performance of the controllers is compared to linear and nonlinear model based control technologies accepted within the control community. The areas of investigation with respect to performance included, the ability to move a process from one operating point to another (servo control), the ability to operate around and detect singular operating points, objective function flexibility, computational burden and design effort. Throughout the design and testing of the control structures some results consistently held true. It was found that the error trajectory objective is not well suited or easily interpreted in the discrete domain. Satisfaction of the error trajectory objective at discrete intervals does not generally result in desired performance. This was found to be true for both, discrete application of the continuous error trajectory controller and the sampled error trajectory objectives from section 3.2.

Many sampled error trajectory controllers were developed that attempted to retain the benefits of the continuous equivalent yet are designed for discrete application. The initial intent of these discrete control designs are the retention of the light computational complexity of the control law, which typically consists of algebraic equations. These sampled error trajectory formulations found limited success, offering improved performance from the discrete application of the continuous controller, overall however their performance was not acceptable. Abandoning the attempt at deriving a computationally efficient discrete error trajectory controller, an integrated structure was introduced that best emulated the continuous objective in the discrete domain. The integrating error trajectory objective is cast as an optimization problem, requiring the iterative integration of the system model over the controller sampling interval. The performance of the integrating controller was adequate but did not fully exploit the opportunities afforded by the optimization objective structure. Borrowing some techniques from predictive control, constraints and input move suppression are included in the objective to produce an optimal error trajectory controller. Input move suppression is used to remove or reduce ill conditioning in the plant model and imparts stabilizing qualities in predictive controllers. The applicability of these predictive control techniques are appropriate for the optimal error trajectory controller as it can be viewed as predictive controller with a limited prediction horizon of one sampling interval. An additional benefit provided by the use of an optimization objective is the relevance for nonsquare systems. The continuous error trajectory controller is only applicable to square systems as the control law produces N_Y equations and N_u unknowns.

Continuing with the attempt at designing a computationally efficient nonlinear control technology the ITPC nonlinear predictive controller is developed. Using a unique variable transformation technique the nonlinear system is broken into separate steady state and dynamic descriptions. The technique allows nonlinear prediction of system response through elementary matrix algebra. It was shown that the ITPC prediction method is capable of retaining not only the steady state nonlinear map but also the system's nonlinear dynamic properties. This is accomplished through the introduction of an intermediate state prediction layer.

Comparatively, the performance of the ITPC controller was as good or better than all the control technologies tested on the two example problems. The flexibility of its objective provides for implementation on a wide range of applications, including regulation and optimization. In comparison the error trajectory objective is primarily relevant for the regulation problem. In addition the error trajectory method is not easily applied to systems that can not be described in a control affine structure. In the unconstrained case, where the transformed inputs uniquely define a set of physical system inputs, an explicit ITPC controller may be derived. The explicit controller resembles the unconstrained DMC algorithm with an additional transformed input inversion step, to retrieve the physical system inputs. The explicit ITPC controller does not sacrifice predictive ability but does modify the input move suppression weighting. The transformed input changes are penalized, not the physical inputs. This requires a well conditioned input inverse function.

The general nonlinear predictive controller described in section 2.2 and the ITPC method are very similar, only differing in their predictive algorithm. The general controller estimates future system response by numerically integrating the full system model over the prediction horizon. This procedure is computationally expensive relative to the ITPC prediction method. A limitation of the ITPC prediction method is the zero order hold description of continuous states included in the ITPC transformed inputs. This modeling constraint introduces a source of error. This error however did not considerably change controller performance for the example systems tested within, as the response of the general nonlinear predictive controller and ITPC were very similar. A popular extension of the general nonlinear predictive controller is the local linearization technique described in section 2.3. Application of the local linear controller on the first and CSTR example systems was performed with limited success. In both of the examples the controller is required to maintain the system around a singular point including an inversion of the process gains. The linear models used for prediction of future control changes can not detect the future gain changes, limiting the controller's predictive ability, thus hindering performance.

The applicability of the error trajectory and ITPC control methods are related to many factors including, system model, computing and control hardware and the control objective. If the controller sample time is insignificant with respect to system dynamics the error trajectory method is excellent. If however this same system exhibits nonminimum phase dynamics or the objective includes economic or optimization variables the error trajectory method may not be appropriate. Application of the error trajectory method on sampled systems with dynamics that are fast with respect to the controller execution frequency was generally found to be ineffective. The discrete formulations that were effective including the integrated and optimal error trajectory control are as computationally expensive at predictive techniques and if anything did not perform as well as the predictive controllers. The ITPC controller is structured identical to the general nonlinear predictive controller. The prediction algorithm however consists of elementary linear algebraic operations as opposed to the integration of the full system model. ITPC therefore is an alternative that may be used to decrease computational complexity and in special cases an explicit solution may exist.

6.0 Conclusions

i) The error trajectory control method is not easily interpreted in the discrete domain. Satisfaction of the continuous error trajectory objective at discrete intervals does not in general result in appropriate or intended performance. This poor performance is attributed to the inappropriateness of error trajectory objective in the discrete domain.

ii) Many discrete forms of the error trajectory method are introduced. The first set of discrete controllers are those that satisfy the error trajectory objective at the next sampling interval given the predicted system trajectory to the next controller sampling interval. The second set includes those that minimize the integrated distance from the error manifold over the sampling interval. Performance of the first set of discrete error trajectory controllers was found to be poor if the controller sampling interval was significant with respect to the system dynamics. Performance of the integrating error trajectory controllers was acceptable and best emulated the error trajectory objective in the discrete domain. Additionally the integrating error trajectory controllers are applicable to nonsquare systems unlike the traditional error trajectory method.

iii) An extended form of the integrated error trajectory controller referred to as optimal error trajectory control includes enhancements from predictive control structures such as input move suppression, output weighting and system constraints. This optimal method was found to have improved stability especially around singular operating points and less erratic input moves than the basic integrated method.

iv) The predictive algorithm for the input transformation predictive control (ITPC) technique described within allows for the nonlinear prediction of system behaviour through elementary matrix algebraic operations. The ITPC prediction method retains both steady state and dynamic nonlinear properties of the system. Steady state nonlinearity is represented through input transformation and dynamic nonlinearity is captured through the introduction of an intermediate state prediction layer.

v) Performance of the ITPC controller was observed to be as good or better than all the control technologies tested within when considering a broad range of characteristics including, the ability to move a process from one operating point to another (servo control), the ability to operate around and detect singular operating points, objective function flexibility, computational burden and design effort. The control technologies included are the various error trajectory controllers described and linear, local linear and nonlinear predictive control structures.

7.0 Future Research

Throughout the development of the topics covered within many issues were not explained in detail due to the limited scope of this report. Furthermore the development of the control structures presented give rise to interesting topics worthy of further investigation. This section summarizes the areas of study that were not covered in full and suggests topics for future research.

The most significant omission in the included studies is the treatment of noise and measurement or estimation of system states. Early on it was stated that proper examination of noise and unknown disturbance rejection requires the use and analysis of filtering techniques such as the extended Kalman filter (EKF) therefore all states were perfectly observed and no noise or stochastic states were included in the example systems. The exclusion of filtering effects on control performance allowed a more direct comparison of the control techniques presented. The perfect observation of the system however is unrealistic for application on real physical systems. Analysis of the effect of state estimation through filtering on controller performance with respect to measurement noise may produce some additional important conclusions. For example if the estimation of state values is poor or slow the difference in performance between selected control technologies may become insignificant as the main limitation to controller performance becomes the filter itself. An extension to the analysis of filtering effects is the propagation of noise through the ITPC prediction method by the inclusion of the intermediate state prediction layer. In a related topic, the random walk assumption of future disturbances used within could be expanded to incorporate stochastic noise models.

The analysis of the controllers investigated within primarily focused on issues relating to performance. The control community is interested not only in performance but also in many cases stability. It is mentioned within that stability results for linear predictive controllers can be applied to the explicit ITPC controller. In fact it may be possible to extend results from linear constrained predictive controllers to the general ITPC controller assuming the feasible region of the physical system inputs can be expressed in terms of the ITPC transformed inputs.

If appropriate representations can be found for the transformed input feasible regions given a constrained input space, another potentially significant extension to the presented ITPC method exists. This extension involves transforming the objective to a constrained version of the explicit form of the ITPC objective. The advantage of this transformation is it reduces the nonlinear programming problem into a much more convenient quadratic program. A secondary requirement of this technique is a suitable input inverse function.

Many authors have expressed the benefits of infinite horizon control over predictive controllers that utilize finite prediction horizons. The separation of steady state and dynamic properties of a system used in the ITPC prediction method may provide a convenient structure that enables representation for infinite horizons. This is especially relevant for the explicit ITPC structure. Extension of the ITPC method for infinite horizons may impart additional stability properties and is offered as an area of future study.

The stability of error trajectory controllers was lightly discussed through explanation of its limitations, such as inability to control systems containing nonminimum phase dynamics. In literature stability results for discrete error trajectory controllers is weak. An area that may be of interest for future study is the stability of the sampled and integrating discrete error trajectory control structures presented within, especially given the portability of these methods to predictive control analysis, as illustrated.

Identification of a system model is required for the development of the controllers described within yet details on system identification were not given. The approach taken to identify a model is a complex task and varies between different fields of study. Factors such as measurement noise and the required resolution influence the identification procedure. Many resources are available in literature and the topic is an active area of research. While system identification is a relatively mature area of study the required amount of resolution in the model used for control to satisfy performance specifications is not necessarily clear. In particular the inclusion of nonlinearity or certain types of nonlinearity in the system model can add unnecessary complexity. For instance Stack and Doyle (1997) suggest that some classes of highly nonlinear systems are optimally controlled by linear controllers, while different systems with much smaller measured nonlinearities demand the used of nonlinear control. Foss and Johansen (1997) discuss the interplay between identification of a predictor and the convexity of the associated optimization-based control problem. These two articles illustrate the potential benefit of selective inclusion and/or exclusion of certain nonlinear properties. Continued research in understanding the relationship between the actual system, the model used for control and the resulting closed loop controller performance will provide useful information for the identification problem and is important for simplifying the resulting control problem.

Qualitatively it is suggested that the ITPC prediction method is computationally more efficient than the general nonlinear predictive control structure, which requires iterative integration of the system model. Quantitatively the difference in efficiency is not explained and was not investigated. Anecdotal evidence was observed from the relative computer time used for different simulation. For example, a given simulation with the ITPC control structure may take approximately one minute while the general nonlinear controller would take an hour. This result depends highly on the integration algorithm and other programming subtleties and it should be said that neither prediction method was optimized for performance. Analysis on quantitative differences in computational efficiency would aid in understanding the actual advantages of using one prediction method over another.

8.0 References

Arapostathis A., B. Jakubczyk, H.-G. Lee, S. I. Marcus and E. D. Sontag, The effect of sampling on linear equivalence and feedback linearization, *Sys. & Cont. Let.*, Vol. 13, pp. 373-381, (1989).

Bartusiak R. D., C. Georgakis and M. J. Reilly, Nonlinear feedforward/feedback control structures designed by reference system synthesis, *Chem. Engng. Sci.*, Vol. 44, pp. 1837-1851, (1989).

Bequette W. B., Nonlinear Control of Chemical Processes: A Review, *Ind. Eng. Chem. Res.* Vol. 30, pp. 1391-1413, (1991).

Boggs P. T. and J. W. Tolle, A Strategy for Global Convergence in a Sequential Quadratic Programming Algorithm, *SIAM J. Numer. Anal.*, Vol. 26, pp. 600-623, (1989).

Coleman T., M. A. Branch and A. Grace, Optimization Toolbox, For Use with MATLAB, Users Guide, Version 2.

Cutler D. M. and B. L. Ramaker, Dynamic Matrix Control – A Computer Control Algorithm, *Proc. Oint Auto. Control Conf.*, Paper WP5-B,San Francisco, (1980).

Dennis J. E., J. Heinkenschloss and L. N. Vicente, Trust-Region Interior-Point SQP Algorithms for a Class of Nonlinear Programming Problems, *SIAM J. Control Optim.*, Vol. 36, pp. 1750-1794, (1998).

Doucet A., *On sequential simulation-based methods for Bayesian filtering*. Technical Report CUED/FINFENG /TR.310, Department of Engineering, University of Cambridge, 1998.

Doyle F. J. III and L. S. Balasubramhanya, Nonlinear Model-based Control of a Batch Reactive Distillation Column, *J. Process Control*, Vol. 10, pp. 209-280, (2000).

Economou C. G., A. M. Morari and B. O. Palsson, Internal Model Control. 5. Extention to Nonlinear Systems. *Ind. Eng. Chem. Process Des. Dev.* Vol. 25, pp. 403-411, (1986).

Fernandez B. R. and K. J. Hedrick, Control of multivariable non-linear systems by the sliding mode method. *Int. J. Control*, Vol. 46, pp. 1019-1040, (1987).

Ferreira A. M. and S. K. Agrawal, On Discrete-Time Trajectory Planning for State Space Exact Linearizable Systems, 2000 AMERICAN CONTROL CONFERENCE ACC00-ASME1046.

Fletcher R. and M. J. D. Powell, A rapidly convergent descent method for minimization, *Comp. J.*, Vol. 6, pp. 163-168, (1963).

Fogler S. H., *Elements of Chemical Reaction Engineering, Second Edition*, Prentice Hall PTR, 1992.

Foss B. A. and T. A. Johansen, Identification and convexity in optimizing control, *In Proc. SYSID'97, Fukuoka, Japan*, pp. 691--6, 1997.

Foss B. A., B. Lohmann and W. Marquardt, A Field Study of the Industrial Modeling Process. In *Proc. ADCHEM'97, Banff, Canada*, pages 581--7, 1997. Keynote lecture.

Gagnon L. and J. F. MacGregor, State Estimation For Continuous Emulsion Polymerization, *The Can. J. of Chem. Engng*, Vol. 69, pp. 648-656, (1991).

Garcia C. E., Quadratic/Dynamic Matrix Control of Nonlinear Processes: An Application to a Batch Reaction Process. Presented at the 1984 AIChI Annual Meeting, San Francisco CA, (1984).

Garcia C. E. and M. Morari, Internal model control. 1. A unifying review and some new results, *Ind. Eng. Chem. Proc. Des. Dev.*, Vol. 21, pp. 308-323, (1982).

Garcia C. E. and M. Morari, Internal Model Control. 3. Multivariable Control Law Computation and Tuning Guidelines, *Ind. Eng. Chem. Proc. Des. Dev.*, Vol. 24, pp. 484-494, (1985).

Garcia C. E. and A. M. Morshedi, Quadratic programming solution of dynamic matrix control (QDMC), *Chem. Eng. Commun.*, Vol. 46, pp. 73-87, (1986).

Gattu G. and E. Zafiriou, Observer Based Nonlinear Quadratic Dynamic Matrix Control for State Space and Input/Output Models, *The Can. J. of Chem. Engng*, Vol. 73, pp. 883-895, (1995).

Geladi P., Notes on the History and Nature of Parial Least Squares (PLS) Modelling, *Journal of Chemometrics*, Vol. 2, pp. 231-246 (1988).

Geladi P. and B. R. Kowalski, Partial Least-Squares Regression: A Tutorial, *Analytica Chemica Acta*, Vol. 185, pp. 1-17, (1986).

Grizzle J. W. and P. V. Kokotovic, Feedback Linearization of Samples-Data Systems, *IEEE Trans. on Auto. Cont.*, Vol. 33, pp. 857-859, (1988).

Guay M., P. J. McLellan and D. W. Bacon, Measurement of Nonlinearity in Chemical Process Control Systems: The Steady State Map, *The Can. J. of Chem. Engng*, Vol. 73, pp. 868-882, (1995).

Isidori A., *Nonlinear Control Systems*, Prentice Hall, (1991).

Jakubczyk B. and E. D. Sontag, Controllability of Nonlinear Discrete Time Systems: a Lie-Algebraic Approach, *SIAM J. Cont. and Opt.*, Vol. 28, (1990).

Jang S. B., B. Joseph and H. Mukai, Control of constrained multivariable nonlinear process using a two-phase approach, *Ind. Eng. Chem. Res.*, Vol. 26, pp. 2106-2114, (1987).

Kravaris C. and C. Chung, Nonlinear State Feedback Synthesis by Global Input/Output Linearization, *AIChE J.*, Vol. 33, pp. 592-603, (1987).

Kravaris C. and J. C. Kantor, Geometric Methods for Nonlinear Process Control. 1. Background. *Ind. Eng. Chem. Res.*, Vol. 29, pp. 2295-2310, (1990a).

Kravaris C. and J. C. Kantor, Geometric Methods for Nonlinear Process Control. 2. Controller Synthesis. *Ind. Eng. Chem. Res.*, Vol. 29, pp. 2310-2324, (1990b).

Kravaris C. and M. Soroush, Synthesis of Multivariable Nonlinear Controllers by Input/Output Linearization, *AIChE J.*, Vol. 36, pp. 249-264, (1990).

Kruger U., Q. Chen, D. J. Sandoz and R. C. McFarlane, Extended PLS Approach for Enhanced Condition Monitoring of Industrial Processes, *AIChI Journal,* Vol. 47, pp. 2076-2091, (2001).

Lee P. L. and G. R. Sullivan, Generic Model Control (GMC), *Comput. Chem. Eng.,* Vol. 12, pp. 573-580, (1988).

Liao L.-Z. and C. A. Shoemaker, Advantages of Differential Dynamic Programming over Newton's method for discrete-time optimal control problems, Tech. Report, CTC92TR97, 1992.

Li W. C. and L. T. Biegler, Multistep, Newton type control strategies for constrained, nonlinear processes. *Chem. Eng. Res. Des.*, Vol. 67, pp. 562-577, (1989).

McLellan P. J., T. J. Harris and D. W. Bacon, Error trajectory descriptions of nonlinear controller designs, *Chem. Eng. Sci.*, Vol. 45, pp. 3017-3034, (1990).

Monaco S. and D. Normand-Cyrot, A Unified Representation for Nonlinear Discrete-Time and Sampled Dynamics, *Journal of Mathematical Systems, Estimation and Control*, Vol. 7, pp. 477-503, (1997).

Morari, M., Model Predictive Control: Multivariable Control Technique of Choice in the 1990s?, *California Institute of Technology – Technical Reports*, Serial Code: aD103, CIT/CDS 93-024, (1993).

Parrish J. R. and C. B. Brosilow, Nonlinear Inferential Control. *AIChE J.*, Vol. 34, pp. 633-644, (1988).

Qin S. J. and T. A. Badgwell, An Overview of Industrial Model Predictive Control Technology. In Jeffrey C. Kantor, Carlos E. Garcia, and Brice Carnahan, editors, *fifth International Conference on Chemical Process Control*, pp. 232-256. AIChE and CACHE, (1997).

Qin S. J. and T. A. Badgwell, An overview of nonlinear model predictive control applications. In *Nonlinear Model Predictive Control*, edited by F. Allgower and A. Zheng, Birkhauser, Switzerland, (2000).

Qin S. J., S. Valle and M. J. Piovoso, On unifying multiblock analysis with application to decentralized process monitoring, *Journal of Chemometrics,* Vol. 15, pp. 715-742, (2001).

Seborg D. E., T. F. Edgar and D. A. Mellichamp, *Process Dynamics and Control*, Wiley, (1989).

Shanno D. F., Conditioning of Quasi-Newton Methods for Function Minimization, *Mathematics of Computation*, Vol. 24, pp. 647-656, (1970).

Sistu P. B., R. S. Gopinath and B. W. Bequette, Computational issues in nonlinear predictive control, *Computers chem. Engng.*, Vol 17, pp 361-366, (1993).

Stack A. J. and F. J. Doyle III, The Optimal Control Structure: An Approach to Measuring Control-Relevant Nonlinearity, *Comp. Chem. Eng.,* Vol. 21, pp. 998-1009, (1997).

Zafiriou E., On the Effect of Tuning Parameters and Constraints on the Robustness of Model Predictive Controllers, in Chemical Process Control – CPC IV, Y. Arkun and W. H. Ray (eds), American Institute of Chemical Engineers, 363-393, (1991).