# INCORPORATION OF THE GENERALIZED TSK MODELS IN MODEL PREDICTIVE CONTROL

By

HAOXIAN CHEN

Bachelor of Engineering in Automation

East China University of Science and Technology

Shanghai, China

2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2011

INCORPORATION OF THE GENERALIZED TSK

MODELS IN MODEL PREDICTIVE CONTROL

Thesis Approved:

Dr. R. Russell Rhinehart

Thesis Adviser

Dr. Joshua D. Ramsey

Dr. Heather Fahlenkamp

Dr. Sheryl A. Tucker

Dean of the Graduate College

## ACKNOWLEDGEMENTS

This has been a journey. Much grace has been received, and many thanks need to be sent.

Foremost I would like to express my sincere appreciation to my research advisor, Dr. R. Russell Rhinehart whose patience, kindness, guidance, insight and encouragement has always been a shining lamp during my studies. His love and enthusiasm for research greatly inspired me to explore and discover, to “Do, not study”. He challenges me when I am not making my best efforts. He is also a great example for me both as a teacher and a researcher. Lastly, I am very grateful to him for the financial support while pursuing my education at OSU.

I thank my committee members, Dr. Joshua D. Ramsey and Dr. Heather Fahlenkamp for their insightful comments and suggestions as well as warming support and encouragement.

I want to thank my former labmate Dr. Ming Su for his coaching and inspiration. I am also thankful to Mr. Anand Govindarajan, Ms. Upasana Sridhar and Ms. Ting Huang for their generous support and continuous input on my research. Without my fellow labmates’ insightful discussion and sharing, the compilation of this thesis would have been extremely complicated and difficult.

A special thanks to my friend and mentor Dr. Steven Moore. His patient guidance and wise mentorship are ready all the time.

I wish to extend my thanks to the Amoco Foundation for its financial support.

Finally, I would like to thank Ms. Li Zhou for her endless understanding and support at those hard times. Most importantly, I wish to thank my parents, Qiaosheng

Chen and Zhifang Pu for their unconditional love and support in every way possible throughout my study in the U.S. and beyond. To them I dedicate this thesis.

Thank God, who has made all things possible, and made me the person that I am today.

TABLE OF CONTENTS

LIST OF TABLES

## LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

## 1.1 Overview

Due to the complexity of chemical processes, and quality and environmental requirements in process operations, advanced process control strategies have been widely used in the industry to fulfill control needs. Model predictive control (MPC), also referred to as moving horizon control (MHC), has become the most effective and attractive advanced control strategy in process industries [1-3]. The term MPC does not designate any kind of specific control strategy but rather a family of control approaches based on the same philosophy. In general, it makes explicit use of a model of the process to obtain a sequence of control signals by optimizing predictions of the process. Therefore, the model is the essential element of an MPC controller.

Linear models were used to predict the process dynamic behavior in the initial industrial MPC applications, and remain most common today[3]. MPC approaches using linear models are called linear model predictive control (LMPC). LMPC was rapidly developed and well accepted in both academia and industry over the past three decades. Some successful commercialization of LMPC, e.g. Dynamic Matrix Control (DMC) and Model Predictive Heuristic Control (MPHC) have enjoyed great popularity in the industry, especially in the area of process control[2]. However, most batch and continuous processes in chemical and petro-chemical industries are nonlinear. Furthermore, efficiency demands and manufacturing flexibility in today's plants often drive process over a wide region, often very close to the operation boundaries. Considering these facts, linear MPC strategies may not always provide satisfactory performance for many industrial applications. Therefore, nonlinear model predictive control (NMPC) is drawing more attention with respect to both theoretical and application aspects[3].

A sizable number of studies on new NMPC algorithms and related industrial application were reported in the past 15 years[2]. In general, the NMPC used in these works has a nonlinear predictive model and is a direct and intuitive extension of LMPC. Unlike LMPC which is based on linear dynamic models, NMPC makes use of nonlinear models to address the issue of process nonlinearity and frequently changing operating region. With a better understanding of process nonlinearity, nonlinear models are expected to improve the control performance of MPC. However, approaches to develop adequate nonlinear models from plant testing data are usually complicated. The complexity of nonlinear models also adds computational burden to model prediction and dynamic optimization of NMPC.

There are three main types of models that are used to represent general nonlinear process in the area of nonlinear model predictive control: first principles models, empirical models and grey box approaches. Among these three, the empirical modeling approach is the most popular modeling method for NMPC controller development and applications according to the survey of the recent studies[3].

The Takagi-Sugeno-Kang (TSK)[4] fuzzy model, as a type of empirical model, has been used to represent complex systems in many recent NMPC publications. In this work, a new generalized TSK modeling approach, which is referred to as GTSK (generalized TSK) modeling[5], is used in the nonlinear model generation, and as the modeling approach for the predictive controller.

The GTSK modeling approach was developed based on the TSK fuzzy model representation. It proposed a more efficient TSK model with the generalized rule antecedent structure. By reducing antecedent dimension and introducing a more flexible antecedent structure, the number of rules was significantly reduced in this new TSK structure model. The innovation of GTSK model will alleviate the computational burden of an NMPC, and be qualified for on-line applications.

A novel global optimization method, the Leapfrogging technique[6], is also used to further improve the NMPC's computational efficiency. Another innovation in this work is a new test input signal design in the model generation part. Instead of using a skyline function to generate the test signal, the "sawtooth" pattern is used as the input to generate the GTSK model.

In this work, these three innovations are demonstrated to work effectively by simulation. The experimental system is a nonlinear process simulator, in which the NMPC algorithm was embedded. The virtual process in this simulator is fourth-order-plus-dead-time (FOPDT) process with a nonlinear gain and the environmental effect (noise and disturbance). It is subject to both soft and hard constraints – soft on both the controlled and the auxiliary variable, and hard on both the limits and rate of change of the manipulated variable. The NMPC performance is evaluated via several simulation experiments, which involved constraint handling, interactions and process nonlinearity.

This study features simulation demonstration. Extending the use of GTSK model in MPC to the multi-input-multi-output system, or implementing the MPC with GTSK model on the real systems, like the heat exchanger or distillation column, is considered to be done to demonstrate its effectiveness on the real process and prepare it for future applications.

## 1.2 Literature Survey

### 1.2.1 Model Predictive Control

The term Model Predictive Control (MPC) describes a class of control algorithms that regulate the future process behavior by explicitly using a predictive process model[7]. Based on past measurements and given future control signals, the MPC algorithm can predict the future dynamic behavior of the process through the model. At each time interval, MPC calculates an optimal sequence of future control signals to drive the process to a set point over a finite horizon. The first control signal of the sequence is then applied on the process at each step[3].

In general, the model predictive problem is formulated as solving on-line a finite horizon optimal control problem subject to system dynamics and constraints[1]. A MPC controller is usually comprised of three parts, the predictive process model, the constrained optimizer, and the model adjustment[8].

### 1.2.2 Modeling Approaches

The nonlinear predictive model is most critical part in NMPC. Many efforts were made to explore suitable modeling approaches for nonlinear system representation and application in NMPC. In general, three main types of models are used to represent general nonlinear process in the area of nonlinear model predictive control. They are known as first principles models, empirical models and semi-empirical (grey box) approaches[8].

The first principle model comes directly from balance equations, i.e., material balance, energy balance, momentum balance, together with the hydraulic and thermodynamic information of the process[3, 8]. The first principle model contains the information of process characteristics, and provides deep understanding of the process mechanics. However, constructing a first principle model is usually complicated and costly, sometimes even infeasible. The complexity of the first principle model could also cause computational difficulties in incorporating it into a NMPC[8].

In contrast to a first principle model, an empirical model treats the process as a black box and fits itself to the process data[9]. The modeling method assumes that the characteristics of the process are embedded in the given process data. Comparing to a first principle model, an empirical model usually does not provide information on the mechanics of the process.

Between the first principle model and the empirical model is the grey box model. A grey box model is developed by combining the first principle and empirical approaches. For instance, some parameters in a first principle model are unavailable but can be estimated by an empirical approach. Or, in an empirical model, some parameters can be determined by the knowledge of the process characteristics, such as the structure of the model, or the order of the process[8].

Among these three, empirical modeling is now widely used for NMPC application. Its simplicity and data-oriented feature benefit both model development and NMPC computation.

Neural Network (NNs)[10] is one of the most popular empirical modeling approaches[11]. Neural Network, also called artificial Neural Network, have an inherent ability to approximate any nonlinear function to an arbitrary degree of accuracy[12]. Coupled with appropriate training techniques, Neural Network has been very successful in many NMPC applications and commercial products[3].

Fuzzy model is another type of empirical model that has been widely used. The recent development of fuzzy system attracts many attempts of incorporating fuzzy modeling techniques intro MPC from both researchers and practitioners[13]. Takagi-Sugeno (TS) fuzzy model, which was first introduced by Takagi and Sugeno in 1985[14], has been applied on function approximation, stability analysis, and controller synthesis over the last twenty years or so[13]. TS model defines a set of fuzzy rules to describe a nonlinear system. A number of local linear models, defined by the rules, are smoothly blended by fuzzy membership functions to represent the global system. In 1988, Sugeno and Kang proposed a new method to improve the structure identification of a TS type model[4], which is then referred as TSK model. TS or TSK models are recognized as universal approximators[15], which are able to describe any nonlinear behavior with a sufficiently flexible structure.

### 1.2.3 NMPC Applications

Neural Network model and fuzzy model are two of the most popular empirical models used in NMPC applications. The recent development of NMPC using these two types of models was reviewed in the following section.

#### 1.2.3.1 Neural Network (NN) Model based NMPC

Georgieva and Azevedo (2011)[16] practiced a model predictive control based on recurrent neural network models. Two types of regression NN models which were identified to be suitable for the model predictive control were proposed. A sugar crystallization process case study was conducted to test the NN-based MPC and its performance.

Al Seyab and Cao (2007)[17] developed an efficient algorithm to train general differential recurrent neural network (DRNN), which could be directly used as the modeling approach for nonlinear model predictive control. This novel training algorithm is based on the efficient Levenberg-Marquardt method, which is combined with the automatic differentiation method. In this work, the trained NN can give an accurate approximation at different sampling time without the re-training. A two-CTSR process is used as a case study to demonstrate the benefit of using this algorithm and the improved control performance.

A novel MPC algorithm using a grouped-neural network (GNN) model was presented by Ou and Rhinehart (2003)[8, 9, 18]. GNN modeling is introduced as an approach for nonlinear long-range prediction by less computational effort. The NN model comprises of a group of sub-models, which are independent and run in parallel. Instead of predicting over a certain-range horizon, each sub-model provides prediction of one process output at one selected future point, reducing the computational burden. The implementation of proposed GNN MPC on a nonlinear, multivariable, constrained pilot-scale distillation unit is demonstrated with the controller performance test.

### 1.2.3.2 Fuzzy Model based NMPC

Eliasi, Davliu and Menhaj (2006)[19] developed an adaptive TSK fuzzy model based predictive controller to control the water level of nuclear steam generators. A recursive estimation algorithm was employed to tune the parameters of the TSK model at each time step. The control performance for tracking the step and ramp reference trajectories and against the stream flow rate change was demonstrated. The proposed NMPC was also compared to the PI controller and showed better performance.

A rule-adaptive fuzzy NMPC using TS type model was designed for a multivariable heating system by Roy, Mann and Hawlader (2005)[20]. The goal of this work is to design a MPC algorithm to control temperatures at three locations in the soil sample using three heat sources at the outer surface of the soil cell. The soil-heating system is modeled using a general-purpose ABAQUS Finite Element program. Under the TS type fuzzy model structure, an adaptive mechanism was used to handle the time-variant behavior of the process. The control performance was compared with a classical non-adaptive fuzzy MPC.

Huang, Lou, Gong and Edgar (2000)[21] introduced a fuzzy model predictive control approach using TS modeling methodology. The nonlinear process system is described by a fuzzy convolution model that consists of a number of linear fuzzy models. In the controller design, a two-layered iterative optimization process was employed to minimize prediction errors and control energy. This two-layer design avoids extensive on-line nonlinear optimization and permits the design of a controller based on linear control theory. Nevertheless, the hierarchical control design leads to more modeling and optimization computation, as well as the complexity of the controller structure.

Mahfouf, Linkfens and Abbod (2000)[22] proposed a TSK model based Generalized Predictive Control (GPC) type MPC. The proposed fuzzy modeling approach is based on a Controlled Auto-Regressive Integrated Moving Average (CARIMA) model structure. An adaptive control scheme was integrated with the proposed control algorithm. Controller's performance and application on the binary distillation column and the Continuous Stirred Tank Reactor (CSTR) system were tested.

### 1.2.4 Optimization Methods for NMPC

In a MPC controller, a set of optimal MV values which drive the process to the desired set point without violating constraints need to be computed at each sampling time. This dynamic optimization must be solved on-line, which has always been a challenge for researchers and practitioners.

In most industrial applications of NMPC, the optimization problem is described as linear program (LP), quadratic program (QP) or nonlinear program (NLP), depending on the type of model and the performance expectation[23]. Newton type methods, such as sequential Quadratic Programming (SQP) and Interior Point Methods, are often used to solve nonlinear problems[24, 25].

A multi-step Newton-type algorithm developed by De Oliveira and Biegler (1995, 1994)[26], named QPKWIK, is used by Aspen’s Target MPC product. This method has the advantage that intermediate solutions, although not optimal, are guaranteed feasible[2]. Diehl, Bock, Nagy and Findeisen (2002)[24] proposed a direct multiple shooting method with a real-time embedding strategy. The application of a deterministic global solution technique on NMPC is reported by Long, Polisetty and Gatzke (2005)[27]. The variable space is reduced using interval analysis techniques to achieve faster convergence. Ghaemi, Sun and Kolmanovsky (2009)[28] introduced Integrated Perturbation Analysis and Sequential Quadratic Programming (InPA-SQP) approach, for the MPC implementation. It synergistically combines the solutions derived using perturbation analysis and SQP to solve the optimization problem with initial state perturbation and input/state constraints.

### 1.2.5 Input Training Signal for Model Generation

Successful empirical model development requires selecting a sufficiently good input training signal. In the area of system identification, the binary signal, the frequency sweep and the multisine are the most commonly used signals[29].

For chemical process control, conventionally, model used in MPC applications are identified through a series of step tests. As summarized in[2, 7], Pseudo-Random Binary Sequence (PRBS) tests are also used in most industrial MPC technologies. In some recent works, Filtered Gaussian White Noise with random amplitude and variation between two periods of the stimulus is used as input training singles. It is referred as “skyline function” signal in GTSK approach, which uses it in the development and testing of models[5].

The sawtooth function used in this work is a signal that jumps or drops to the halfway of a random level, and then ramps to that level at the end of a period of random length.

## 1.3 Summary

The GTSK model has advantages over a TSK model and other fuzzy type models because of fewer model equations and parameters, which provides a computational advantage in both model generation and model use[5]. Leapfrogging is a multi-player optimizer with a global aspect. It has advantages over linear and single trial solution optimizers of finding the global solution in nonlinear applications. It also has advantages over other multi-particle optimizers in computational simplicity and speed of convergence[6]. Sawtooth provides a wider coverage over the whole range of the input training signal. Applying these innovations on NMPC is expected to alleviate the computational burden for on-line applications and improve model accuracy, which should enhance the overall control performance.

# CHAPTER II

# GTSK Model Based MPC

## 2.1 GTSK modeling Approach[5]*

A summary of GTSK modeling approach and its innovation over TSK method is given in Section 2.1.1. The details on developing a GTSK model are presented in the following sections.

### 2.1.1 Overview

In contrast to first-principles modeling, GTSK modeling is an empirical (black-box) modeling technique, which fits itself to the input-output data obtained from the process. As a type of TSK model, it is a subset of novel fuzzy logic-based modeling methodologies, where a nonlinear process system is divided into a number of linear or nearly linear subsystems.

To reduce the complexity of a TSK model, two innovations were introduced[5] in the generalized TSK modeling approach. In a GTSK model, only nonlinear variables are included in the rule antecedent to reduce its dimension. Additionally, an extra degree of freedom is introduced to cover an antecedent space more efficiently.

*Reference citations in headings indicated that substantial portions are duplicated or modified from that reference

### 2.1.1.1 Dimension Reduction in the Antecedent

In a GTSK model, only nonlinear variables are included in the rule antecedent to reduce the antecedent dimension. This is the first innovation of the GSTK approach. Consider a single-input-single-output (SISO) dynamic process, with dynamic orders *ny, nu*, pure time delay *d*, and an additive disturbance *e(t).* The system is represented by

$$y(t) = f\begin{pmatrix} y(t-1),\cdots,y(t-ny), \\ u(t-d),\cdots,u(t-nu-d) \end{pmatrix} + e(t) \tag{2.1}$$

where *y(t)* and *u(t)* are process responses and process input at time *t*. *d* is the pure time delay, while *f* is a nonlinear function.

A rule could be described by a TSK model as below

$$\begin{aligned}
&\textbf{IF } \left(\mathrm{y}(t-1) \text{is } A_1^r \textbf{ AND } \cdots \textbf{ AND } \mathrm{u}(t-nu-d) \text{ is } A_{ny+nu+1}^r\right) \\
&\textbf{THEN} \quad \mathbf{A}^r\left(z^{-1}\right)\mathrm{y}(t) = k^r + \mathbf{B}^r\left(z^{-1}\right)\mathrm{u}(t-d) \\
&\mathbf{A}^r\left(z^{-1}\right) = 1 + a_1^r z^{-1} + \cdots + a_{ny}^r z^{-ny} \\
&\mathbf{B}^r\left(z^{-1}\right) = b_0^r + b_1^r z^{-1} + \cdots + b_{nu}^r z^{-nu}
\end{aligned} \tag{2.2}$$

where, $A_1^r$ is the fuzzy subset for y(t-1) in the rule and z is the backshift operator. The expression $y(t-1) \text{is } A_1^r \textbf{ AND } \cdots \textbf{ AND } u(t-nu-d) \text{ is } A_{ny+nu+1}^r$ is the antecedent of the rule, and the variables *y(t-1), …, y(t-ny), u(t-d),…,u(t-nu-d)* are antecedent variables. The consequent of the rule is a local linear model $\mathbf{A}^r\left(z^{-1}\right)y(t) = k^r + \mathbf{B}^r\left(z^{-1}\right)u(t-d)$.

However, in a TSK model, all the regressors in the rule consequent are also in the rule antecedent, which leads the antecedent dimension to be same as the problem dimension and causes the complexity of a TSK model. In a GTSK model, only the variables appearing nonlinearly were included in the antecedent. This is the first of two innovations introduced by the GTSK modeling. The simplified rule is defined by

$$\begin{aligned}
&\textbf{IF}\ \left(\begin{array}{l} \mathrm{y}(t-1)\,\mathrm{is}\ A_1^r\ \textbf{AND}\ \cdots\ \textbf{AND}\ \mathrm{y}(t-ay)\,\mathrm{is}\ A_{ay}^r\ \textbf{AND} \\ \mathrm{u}(t-d)\ \mathrm{is}\ A_{ay+1}^r\ \textbf{AND}\ \cdots\ \textbf{AND}\ \mathrm{u}(t-bu-d)\ \mathrm{is}\ A_{ay+bu+1}^r \end{array}\right) \\
&\textbf{THEN}\quad \mathbf{A}^r\left(z^{-1}\right)\mathrm{y}(t)=k^r+\mathbf{B}^r\left(z^{-1}\right)\mathrm{u}(t-d) \\
&\mathbf{A}^r\left(z^{-1}\right)=1+a_1^r z^{-1}+\cdots+a_{ny}^r z^{-ny} \\
&\mathbf{B}^r\left(z^{-1}\right)=b_0^r+b_1^r z^{-1}+\cdots+b_{nu}^r z^{-nu}
\end{aligned} \tag{2.3}$$

where the antecedent dimension is reduced to *ay+bu+1* (*ay <ny* and *bu< nu* ). In Equation (2.3)*y(t-1) … y(t-ay), u(t-d) …. u(t-bu-d)* are then antecedent variables. They are collected in an antecedent vector **c**(t). Regressors in the consequent are collected in consequent vector **x**(t).

#### 2.1.1.1 Generalized Antecedent Structure

The second innovation proposed by the GTSK modeling approach is a generalized antecedent structure. This new structure substitutes the combinatorial antecedent structure in the TSK rule in Equation (2.4) by a more flexible one. One more degree of freedom is introduced to improve the covering efficiency of each rule.

Given a two dimensional antecedent with equal number of fuzzy sets for each antecedent variable, a typical combinatorial antecedent space partition is illustrated in Figure 2.1. 9 rules result from the combinations of 3 fuzzy sets for each antecedent variable ($c_i$, i=1, 2)

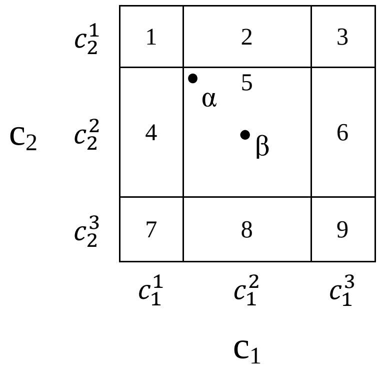


Figure 2.1 Two-Dimension Antecedent

where $c_1$ , $c_2$ are antecedent variables.$c_i^1, c_i^2, c_i^3$ are fuzzy sets of each antecedent variable $c_i$.1, 2, …., 9 is the number of rule. Each rule has a local linear model $\mathbf{A}^r\left(z^{-1}\right)y(t)=k^r+\mathbf{B}^r\left(z^{-1}\right)u(t-d)$, as stated in Equation (2.3).

If points α and β are both in region 5, but α is nearly as close to region 1, 2, 4. Then the rule applied on point β would be only rule 5, while the model value calculation for point α would be also influenced by rules 1, 2, 4. The belongingness of a point to each rule is evaluated by a membership function.

If Gaussian membership functions are applied and the product operator is used for the **AND** conjunction in Equation (2.3), each rule of the antecedent could be evaluated by the truth of antecedent (*TA*)

$$TA = e^{-\left(\frac{c_1-o_1}{\sigma_1}\right)^2-\left(\frac{c_2-o_2}{\sigma_2}\right)^2} \tag{2.4}$$

where *TA* is an ellipsoid centering at ($o_1$,$o_2$) with width of $\sigma_1$ by $\sigma_2$. A possible contour plot of *TA* is shown below

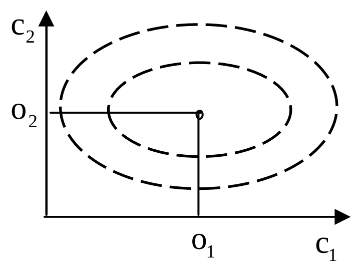


Figure 2.2 The ellipsoid contour of TA

In Figure 2.2, the highest value of *TA* =1 is reached at the centroid. The further out is the contour, the smaller the *TA*. The value of *TA* can be interpreted as the belongingness of a data point to a local region.

Consequently, the two dimensional antecedent structure shown in Figure 2.1could be represented by horizontal and vertical ellipsoids, as shown in Figure 2.1 (a).

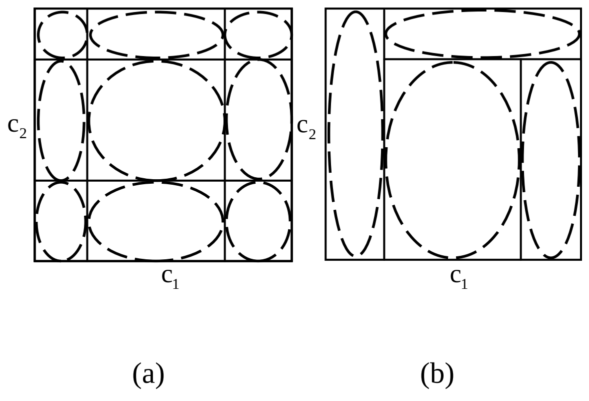


Figure 2.3 Antecedent space partition and representation

A more compact fuzzy model can be constructed by merging regions that exhibit similar local behavior. Figure 2.3 (b) shows a possible partition after merging some regions. However, the partition method which is aligned with the regressor axes in Figure 2.3 becomes inefficient as shown in Figure 2.4, where neither horizontal nor vertical ellipsoids provide an efficient representation of the underlying local region represented by either the rotated “space” of correlated variables or irregular polygons.

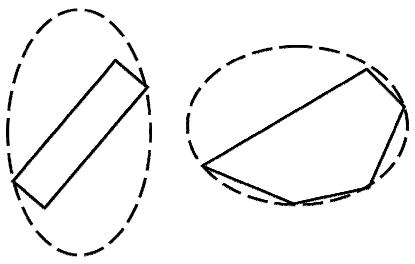


Figure 2.4 A rotated local region covered by a horizontal or vertical ellipsoid

To solve this problem, the GTSK approach chooses to rotate the ellipsoid to cover the local region, which is shown in Figure 2.5.

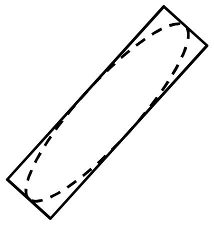


Figure 2.5 A rotated local region covered by a rotated ellipsoid

The rotation is mathematically addressed by one more degree of freedom. the parameters $\sigma$ in Equation (2.4) are replaced by a symmetric positive semi-definite matrix **P**, which is shape matrix in this work, and redefines the truth of antecedent by

$$TA = e^{-(\mathbf{c}-\mathbf{o})^T \mathbf{P}(\mathbf{c}-\mathbf{o})} \tag{2.5}$$

where **o** is the vector representing the centroid with dimension of *nc*, and the dimension for the shape matrix P is *nc* by *nc*.

### 2.1.1.1 GTSK Model Representation

In general, the nonlinear model in Equation (2.1) could be described in the following linear time-varying format

$$\begin{aligned} y(t) = k(t) + a_1(t)y(t-1) + \cdots + a_{ny}(t)y(t-ny) + \\ b_0(t)u(t-d) + \cdots + b_{nu}(t)u(t-nu-d) + e(t) \end{aligned} \tag{2.6}$$

A compact form to represent Equation (2.6) is

$$y(t) = \mathbf{x}^T(t)\boldsymbol{\theta}(t) + e(t) \tag{2.7}$$

with

$$\mathbf{x}(t) = \left[1, y(t-1), \ldots, y(t-ny), u(t-d), \ldots, u(t-nu-d)\right]^T$$
$$\boldsymbol{\theta}(t) = \left[k(t), a_1(t), \ldots, a_{ny}(t), b_0(t), \ldots, b_{nu}(t)\right]^T$$

where **x**(t) is the regressor vector and θ(t) is the parameter vector. Coefficients k(t), a(t) and b(t) are time-varying variables.

Based on Equation (2.3), a GTSK model is defined as below using the generalized antecedent structure.

$$\begin{array}{l} \mathbf{IF}(\mathbf{c}(\mathrm{t})\ \textit{is in}\ \mathrm{R}^1(\mathbf{o}^1, \mathbf{P}^1))\ \ \mathrm{THEN}\ \ \hat{y}^1(\mathrm{t}) = \boldsymbol{\theta}^1\mathbf{x}(\mathrm{t}) \\ \vdots \\ \mathbf{IF}(\mathbf{c}(\mathrm{t})\ \textit{is in}\ \mathrm{R}^M(\mathbf{o}^M, \mathbf{P}^M))\ \ \mathrm{THEN}\ \ \hat{y}^M(\mathrm{t}) = \boldsymbol{\theta}^M\mathbf{x}(\mathrm{t}) \end{array} \tag{2.8}$$

where $\hat{y}^i$ is output from the local model in rule i

The final computation of the GTSK model in Equation (2.8) is defined by

$$\hat{y}(t) = \sum_{i=1}^{M} w^i(t)\,\hat{y}^i(t) \tag{2.9}$$

where $w^i(t)$ are the weights of the local models. In this work, $w^i(t)$ is defined as the normalized truth of the antecedent (TA)

$$w^i(t) = \frac{TA^i(t)}{\sum_{i=1}^{M} TA^i(t)} \tag{2.10}$$

where TA can be calculated by Equation (2.5)

### 2.1.2 Order Determination and Antecedent Variable Selection

The first step in modeling is variable selection. This work first determines the orders, *ny* and *nu*, and delay *d* for a nonlinear dynamic system as defined in Equation (2.1). The value of *ny*, *nu* and *d* give the set of consequent variables in Equation (2.8). Antecedent variables are then selected from the consequent variables.

**A. Nonlinearity Representation Methodology**

Once $\boldsymbol{\theta}$(t) in Equation (2.7) is determined, the orders, *ny* and *nu*, and delay *d*are then defined accordingly. A exponential weighting method[30], represented byEquation (2.11), is used to recursively estimate $\boldsymbol{\theta}$(t).

$$\begin{aligned}
\hat{y}(t) &= \mathbf{x}^T(t)\hat{\boldsymbol{\theta}}(t-1) \\
\mathbf{K}(t) &= \mathbf{P}(t-1)\mathbf{x}(t)\left(\mathbf{x}^T(t)\mathbf{P}(t-1)\mathbf{x}(t)+\alpha\right)^{-1} \\
\hat{\boldsymbol{\theta}}(t) &= \hat{\boldsymbol{\theta}}(t-1)-\mathbf{K}(t)\left(\hat{y}(t)-y(t)\right) \\
\mathbf{P}(t) &= \frac{1}{\alpha}\left(\mathbf{P}(t-1)-\mathbf{K}(t)\mathbf{x}^T(t)\mathbf{P}(t-1)\right)
\end{aligned} \tag{2.11}$$

where $\hat{\boldsymbol{\theta}}(t-1)$ is the parameter value estimates at *t*-1, $\hat{y}(t)$ is the one-step-ahead prediction of *y*(*t*) using $\hat{\boldsymbol{\theta}}(t-1)$, **K**(*t*) is the gain used and correct $\hat{\boldsymbol{\theta}}(t-1)$ to $\hat{\boldsymbol{\theta}}(t)$based on the prediction error, **P**(*t*) records the covariance of $\hat{\boldsymbol{\theta}}(t)$.

In Equation (2.11), α is the tuning factor, termed as ‘forgetting factor’, which has to be chosen for a balanced performance for nonlinearity adaptation and parameter estimation precision. The GTSK modeling approach uses α =0.95 and finds results are relatively insensitive to its choice.

In this work, the resultant regressor vector **x**(t) and output y(t) are reorganized in a “spatial” order to minimize the change in coefficient values during the recursive estimation. The reordering procedure is termed as Sequential Nearest Neighbor Rearrangement (SNNR).By reducing the parameter variation, the SNNR is able to reduce the mean squared error (MSE) of one-step-ahead prediction.

## B. Order Determination by Regressor Selection

In this work, the order determination starts by selecting regressors with user-given value of possible maximum *ny*, *nu* and *d*. Then, a number of candidate regressors are generated by Equation (2.9) and denoted as $[x_1(t)\ x_2(t)\ x_3(t)\ldots\ x_m(t)x^{\text{random}}(t)]$. Regressors of $x_1(t),\ldots,x_m(t)$ are lagged $y(t)$ and $u(t)$ as shown in the regressor vector **x**(t). Regressor $x_{\text{random}}$ is a regressor comprised of random number that presumably contains no meaningful information on $y(t)$.At first, each of $m$+1 regressors, $x_1(t)$, $x_2(t)$, $x_3(t)\ldots\ x_m(t)x^{\text{random}}(t)$, is tried. The trial starts with time-sequence data $(y(t),[x_i(t)])$, where $y(t)$ is the output and $x_i(t)$ (i=1,…,$m$+1) in bracket is the trial regressor. A SNNR is then conducted on $x_i(t)$ to get rearranged data set $(y_{\text{snnr}}(k), [x_{\text{snnr},i}(k)])$. The exponentially weighted recursive estimation in Equation (2.11) is then applied to the rearranged data. The one-step-ahead prediction error on $y_{\text{snnr}}(k)$ is used to evaluate the prediction quality of each regressor, $x_i(t)$. The evaluation criteria is the following Final Prediction Error (FPE) index[31]

$$FPE = \frac{L+np}{L-np}\sum_{k=L}^{N-L+1}\varepsilon^2(k)$$

$$L = \frac{4}{1-\alpha} \tag{2.12}$$

where $\varepsilon(k)$ is the one-step-ahead prediction residual in Equation(2.11) for SNNR rearranged data, *np* is the number of regressors, *L* is related to the ‘forgetting factor’ α.

After the first round of estimation and evaluation, the regressor with the minimum FPE is selected. For instance, if $x_2$ is selected, there will be other *m* regressors to be tried. For each tried regressor $x_i(t)$ (i≠2), the time-sequence data is $(y(t), [x_2(t), x_i(t)])$, where the bracket contains the already selected regressors, $x_2(t)$ and the trial regressor $x_\mathrm{i}(t)$. A SNNR is then applied on $[x_2(t)$ $x_\mathrm{i}(t)]$ to get rearranged data $(y_\mathrm{snnr}(k)$, $[x_{\mathrm{snnr},2}(k)$ $x_{\mathrm{snnr,i}}(k)])$. Equation (2.11) is conducted then to get the one-step-ahead prediction error on $y_\mathrm{snnr}(k)$. The quality of $[x_2(t)$ $x_\mathrm{i}(t)]$ combination is then evaluated using FPE. The regressor combination with the minimum FPE is kept.

The selection continues until either the minimum FPE for a selected set increased with respect to the previous set, or the $x_\mathrm{random}(t)$ is selected. The injection of a random regressor is used as a stopping criterion[32]. The selection of $x_\mathrm{random}(t)$ signifies that the rest of candidates are less influential on $y$ than a random pattern.

Values of *ny*, *nu* and *d* could be easily defined by a selected set of consecutive regressors, for instance, $[y(t\text{-}1)$, $y(t\text{-}2)$, $u(t\text{-}1)$, $u(t\text{-}2)]$, due to implicit constraint on the model structure. However, absences could exist in selected regressors such as $[y(t\text{-}1)$ $y(t\text{-}4)$ $u(t\text{-}1)$ $u(t\text{-}3)]$, which does not correspond a set of *ny*, *nu* and *d*. For database management simplicity, in GTSK modeling, if the situation with absence occurs, a further comparison is executed on different order values. For the illustrated example, an exhaustive comparison is conducted on possible values of $ny$=1, 2, 3 or 4 combined the possible values of $nu$=0, 1, or 2, with $d = 1$.

After this procedure, consequent variables $(x_1,\ldots,x_{nx})$ are determined as the selected regressors$(y(t\text{-}1) \ldots y(t\text{-}ny), u(t\text{-}d) \ldots. u(t\text{-}nu\text{-}d))$.

**C. Nonlinear Component Detection**

To detect the regressors that are affecting the output nonlinearly, which are then used as antecedent variables, the similar technique for order determination is used. In GTSK modeling, antecedent variables $(c_1,\ldots,c_{nc})$ are defined as a subset of the $(x_1,\ldots,x_{nx})$. There are $2^{nx\text{-}1}$ subsets in $(x_1,\ldots,x_{nx})$ excluding the empty one. Each subset is considered as a candidate $(c_1,\ldots,c_{nc})$, on which the SNNR is again conducted and a corresponding FPE is computed. The subset with minimum FPE is selected as $(c_1,\ldots,c_{nc})$, which are $(y(t\text{-}1) \ldots y(t\text{-}ay), u(t\text{-}d) \ldots. u(t\text{-}bu\text{-}d))$ in Equation(2.3)

### 2.1.3 Estimation of Parameter Values

Once the consequent variables $(x_1,\ldots,x_{nx})$ and antecedent variables $(c_1,\ldots,c_{nc})$ are determined, the next task is to determine the antecedent structure as well as the parameter values in each rule.

#### A. Methodology

Figure 2.6 illustrates a GTSK model with a two-dimension antecedent structure.

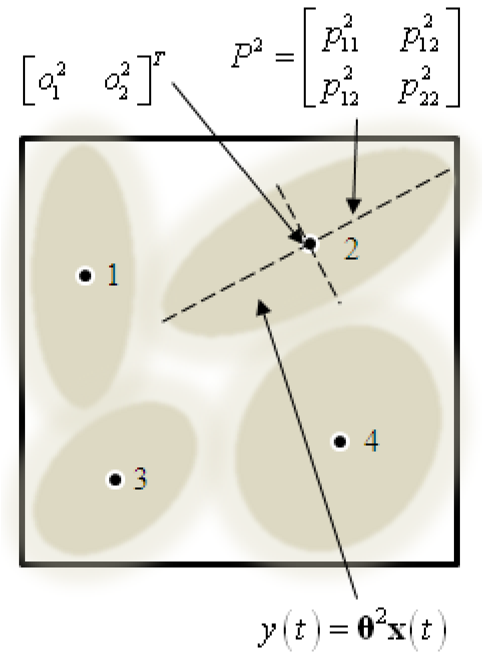

$\begin{bmatrix} a_1^2 & a_2^2 \end{bmatrix}^T$
$P^2 = \begin{bmatrix} p_{11}^2 & p_{12}^2 \\ p_{12}^2 & p_{22}^2 \end{bmatrix}$
1
2
3
4
$y(t) = \boldsymbol{\theta}^2 \mathbf{x}(t)$

Figure 2.6 A GTSK model in a two dimension antecedent space

If underlying rule regions are given, the parameter estimation problem will be easy to solve. In the GTSK approach, rule regions are generated out of an antecedent space by partition. An illustrating example for Figure 2.6 is shown below, where four regions are defined by three linear splitting boundaries (dashed lines).

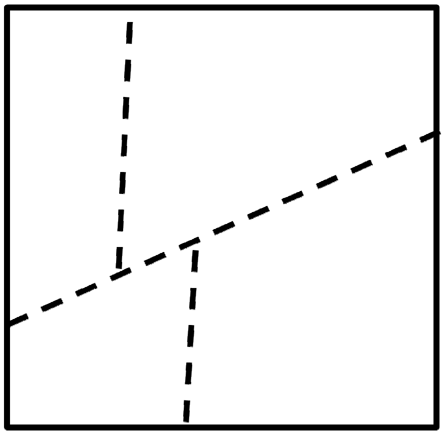


Figure 2.7 Partitioned antecedent space for the GTSK model in Figure 2.6

Once a rule region is determined, an ellipsoid can be defined to cover it, as shown in Figure 2.6. The GTSK approach determines the number and shapes of regions by a recursive partition of the antecedent space. The fundamental step to obtain an antecedent space partition is to solve a splitting and regression problem (SRP).

**B. Solving Splitting and Regression Problem**

An example of one stage in SRP on a two dimensional antecedent space is illustrated in Figure 2.8. The objective is to minimize the modeling error of the partitioned data by the two linear models by placing a linear separation boundary (the bold dashed line) in the antecedent space, which results in two subspaces **A** and **B**. Each subspace has a local linear model. The two linear models shown use all relevant regressors, not just the two ($c_1$ and $c_2$) chosen to express nonlinear behavior. The separation boundary is chosen to be linear, and is a function of the $c_\mathrm{i}$ variables, of which only $c_1$ and $c_2$ are illustrated here.

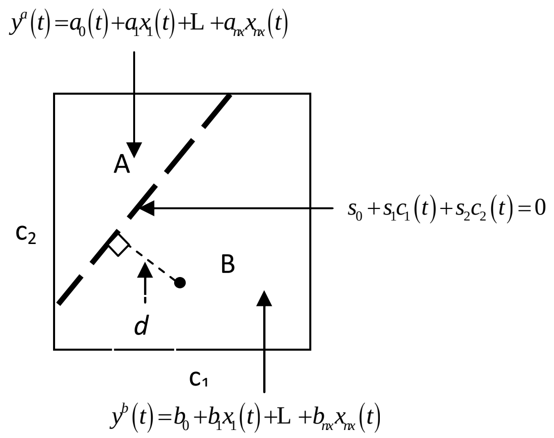

$y^a(t)=a_0(t)+a_1x_1(t)+\mathrm{L}\ +a_{nx}x_{nx}(t)$
A
$s_0+s_1c_1(t)+s_2c_2(t)=0$
c₂
B
d
c₁
$y^b(t)=b_0+b_1x_1(t)+\mathrm{L}\ +b_{nx}x_{nx}(t)$

Figure 2.8 Illustration of a SRP

The belongingness of data sample to subspace **A** is determined by *l*(t) and *φ*(t) as below

$$l(t)=s_0+s_1c_1(t)+s_2c_2(t) \tag{2.13}$$

$$\varphi(t)=\begin{cases}0, & l(t)<0\\ 1, & l(t)\geq 0\end{cases} \tag{2.14}$$

where $s_0$, $s_1$, $s_2$ defines a separation boundary $s_0+s_1c_1(t)+s_2c_2(t)=0$ in Figure 2.8. The value of *l*(*t*) is $\sqrt{s_1^2+s_2^2}$ times of the distance of a point, [$c_1(t)$, $c_2(t)$] to the linear separation boundary , which is $d=\left|s_0+s_1c_1(t)+s_2c_2(t)\right|/\sqrt{s_1^2+s_2^2}$ . However, Equation (2.14) implies that only the sign of *l*(*t*) matters. In Figure 2.8, the points in category A have negative values for *l*(*t*) while B category has positive *l*(*t*).

In Figure 2.8, two local linear models are

$$\begin{aligned}y^a(t)&=a_0+a_1x_1(t)+\mathrm{L}\ +a_{nx}x_{nx}(t)\\ y^b(t)&=b_0+b_1x_1(t)+\mathrm{L}\ +b_{nx}x_{nx}(t)\end{aligned} \tag{2.15}$$

Combing Equation (2.14) with the Equation (2.15), the output is then computed by

$$\hat{y}(t)=(1-\varphi(t))\,y^a(t)+\varphi(t)\,y^b(t) \tag{2.16}$$

The SRP can be then solved by minimizing the following performance index *J*

$$\min_{\mathbf{a},\mathbf{b},\mathbf{s}} J=\sum_{t=1}^{N}\varepsilon^2(t) \tag{2.17}$$

where, $\varepsilon(t) = y(t) - \hat{y}(t)$ is the residual, and parameter values to be estimated include **a** and **b** in Equation (2.15), and **s** in Equation (2.13),

The GTSK approach solves the SRP by a heuristic suboptimal method based on the assumption that there are two local linear models. Given a separation defined by **s**, it results in a split of data [**y C X**] into **A** and **B** regions as [$\mathbf{y_A}\ \mathbf{C_A}\ \mathbf{X_A}$] and [$\mathbf{y_B}\ \mathbf{C_B}\ \mathbf{X_B}$]

$$\begin{aligned}
&\mathbf{y_A} = \mathbf{X_A a} + \mathbf{e_A} \\
&\mathbf{y_B} = \mathbf{X_B b} + \mathbf{e_B} \\
&\text{with} \\
&\mathbf{y_A} = \left[ y^a(1) \quad \mathrm{L} \quad y^a(N_A) \right]^T \\
&\mathbf{y_B} = \left[ y^b(1) \quad \mathrm{L} \quad y^b(N_B) \right]^T \\
&\mathbf{e_A} : N\left(0, \sigma_{\mathbf{A}}^2 \mathbf{I}\right); \qquad \mathbf{e_B} : N\left(0, \sigma_{\mathbf{B}}^2 \mathbf{I}\right)
\end{aligned} \tag{2.18}$$

where, **y** is the vector collecting all *N* sample outputs, **C** is a *N* by *nc* matrix consisting of *N* rows of antecedent variables, and **X** is a *N* by *nx* matrix consisting of *N* rows of consequent variables. $\mathbf{X_A}$ and $\mathbf{X_B}$ are two disjoint subsets of **X**. The corresponding model parameters **a** and **b** are estimated by

$$\begin{aligned}
\hat{\mathbf{a}} &= \left(\mathbf{X}_\mathbf{A}^T \mathbf{X_A}\right)^{-1} \mathbf{X}_\mathbf{A}^T \mathbf{y_A} \\
\hat{\mathbf{b}} &= \left(\mathbf{X}_\mathbf{B}^T \mathbf{X_B}\right)^{-1} \mathbf{X}_B^T \mathbf{y_B}
\end{aligned} \tag{2.19}$$

the residual for model **A** could then be evaluated by

$$\boldsymbol{\varepsilon}_\mathbf{A} = \mathbf{y_A} - \mathbf{X_A}\hat{\mathbf{a}} \tag{2.20}$$

after some algebraic operations, Equation (2.20) is expressed in terms of $\mathbf{e_A}$ by

$$\boldsymbol{\varepsilon}_\mathbf{A} = \left(\mathbf{I} - \mathbf{X_A}\left(\mathbf{X}_\mathbf{A}^T \mathbf{X_A}\right)^{-1} \mathbf{X}_\mathbf{A}^T\right)\mathbf{e_A} \tag{2.21}$$

the quadratic performance criterion is then evaluated and expressed in terms of $\mathbf{X_A}$ and $\sigma_\mathbf{A}$ by

$$\begin{aligned}
J_\mathbf{A} &= E\left[\boldsymbol{\varepsilon}_\mathbf{A}^T \boldsymbol{\varepsilon}_\mathbf{A}\right] \\
&= Tr\left(\mathbf{I} - \mathbf{X_A}\left(\mathbf{X}_\mathbf{A}^T \mathbf{X_A}\right)^{-1} \mathbf{X}_\mathbf{A}^T\right)\sigma_A^2
\end{aligned} \tag{2.22}$$

in the same manner, the performance criterion for model **B** is described by $J_{\mathbf{B}} = Tr\left(\mathbf{I} - \mathbf{X_B}\left(\mathbf{X_B}^T\mathbf{X_B}\right)^{-1}\mathbf{X_B}^T\right)\sigma_{\mathbf{B}}^2$, then the quadratic performance is expressed in terms of $\sigma_A$ and $\sigma_B$ by

$$J = J_{\mathbf{A}} + J_{\mathbf{B}} \tag{2.23}$$

which could be viewed as a weighted combination of $\sigma_{\mathbf{A}}^2$ and $\sigma_{\mathbf{B}}^2$. The weights are determined by the trace function on $\mathbf{X_A}$ and $\mathbf{X_B}$, which are $N_{\mathbf{A}}$-$nx$-1 and $N_{\mathbf{B}}$-$nx$-1 respectively. Since $nx$ is often negligible to $N_{\mathbf{A}}$ and $N_{\mathbf{B}}$, Equation (2.23) is converted to

$$J = N_{\mathbf{A}}\sigma_{\mathbf{A}}^2 + N_{\mathbf{B}}\sigma_{\mathbf{B}}^2 \tag{2.24}$$

where, based on Equation (2.24), $N_{\mathbf{A}}$ and $N_{\mathbf{B}}$ are defined by

$$N_{\mathbf{A}} = \sum_{t=1}^{N}\varphi(t),\ \ N_B = N - \sum_{t=1}^{N}\varphi(t) \tag{2.25}$$

additionally, the unknown $\sigma_{\mathbf{A}}^2$ and $\sigma_{\mathbf{B}}^2$ are to be replaced by their estimates by

$$\begin{aligned}\hat{\sigma}_{\mathbf{A}}^2 &= \sum_{t=1}^{N}\left[\varphi(t)\left(y(t)-\mu_{\mathbf{A}}\right)\right]^2 \Big/ \sum_{t=1}^{N}\varphi(t)\\ \hat{\sigma}_{\mathbf{B}}^2 &= \sum_{t=1}^{N}\left[\left(1-\varphi(t)\right)\left(y(t)-\mu_B\right)\right]^2 \Big/ \sum_{t=1}^{N}\left(1-\varphi(t)\right)\end{aligned} \tag{2.26}$$

where $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ are unknown means of $\mathbf{y_A}$ and $\mathbf{y_B}$ in model **A** and **B**. Substituting Equations (2.25 - 2.26) into Equation (2.23), $J$ is then described by

$$J = \sum_{t=1}^{N}\varphi^2(t)\left(y(t)-\mu_{\mathbf{A}}\right)^2 + \left(1-\varphi(t)\right)^2\left(y(t)-\mu_{\mathbf{B}}\right)^2 \tag{2.27}$$

where, there are $N$+2 decision variables, $N$ belongingness values, and $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$. Since the $\varphi(t)$ s not coupled with any $\varphi(\tau)$ ($t\neq\tau$), it can be solved individually by

$$\varphi(t) = \frac{\left(y(t)-\mu_{\mathbf{B}}\right)^2}{\left(y(t)-\mu_{\mathbf{A}}\right)^2 + \left(y(t)-\mu_{\mathbf{B}}\right)^2} \tag{2.28}$$

Combining Equations (2.28) and (2.27) defines *J* in terms of $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ only by

$$J = \sum_{t=1}^{N} \frac{\left(y(t)-\mu_{\mathbf{A}}\right)^2 \left(y(t)-\mu_{\mathbf{B}}\right)^2}{\left(y(t)-\mu_{\mathbf{A}}\right)^2 + \left(y(t)-\mu_{\mathbf{B}}\right)^2} \tag{2.29}$$

the objective function in Equation (2.29) has only two decision variables $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$. Once *J* is minimized, $\varphi(t)$ is determined by Equation (2.28) and automatically lies between 0 and 1. The resultant $\varphi(t)$ takes any value within 0 and 1 instead of 0 and 1 only. The following Equation (2.30) will convert the $\varphi(t)$ to a two-value indicator (0,1)

$$\varphi(t) = \begin{cases} 0, & \varphi(t) < 0.5 \\ 1, & \varphi(t) \ge 0.5 \end{cases} \tag{2.30}$$

which assigns each data sample to either region **A** or **B**. Notice that the $\varphi(t)$from Equations (2.28) and (2.30) is not confined to a linear separation boundary defined in Equation (2.13). In order to let the indicator values be subject to a linear separation boundary, the following support vector machine (SVM) [33] is then solved to find the linear separation parameters **s** based on $\varphi(t)$ from Equation (2.30)

$$\begin{aligned}
&\text{minimize } \sum_{k=0}^{nc} s_k^2 + r\left( \sum_{i=1}^{N_A} \xi_i^a + \sum_{j=1}^{N_A} \xi_j^b \right) \\
&\text{subject to} \\
&\quad s_0 + s_1 c_1(i) + \cdots + s_{nc} c_{nc}(i) \ge 1 - \xi_i^a, \quad i = 1 \cdots N_A \\
&\quad s_0 + s_1 c_1(j) + \cdots + s_{nc} c_{nc}(j) \le \xi_j^b - 1, \; j = 1 \cdots N_B \\
&\quad \xi_i^a, \xi_i^a \ge 0
\end{aligned} \tag{2.31}$$

The solved **s**is then applied to Equations (2.13) and (2.14) to update $\varphi(t)$, which is now confined a linear separation boundary. The resultant $\varphi(t)$ defines a split, [$\mathbf{y}_A$ $\mathbf{C}_A$ $\mathbf{X}_A$] and [$\mathbf{y}_B$ $\mathbf{C}_B$ $\mathbf{X}_B$]. Then **a** and **b** are estimated by Equation (2.19). It then is able to evaluate residuals $\boldsymbol{\varepsilon}_{\mathbf{A}}$ and $\boldsymbol{\varepsilon}_{\mathbf{B}}$ explicitly by Equation (2.20). The indicator values are then updated by minimizing the following *J* with replacement of ($y(t)$ - $\mu_{\mathbf{A}}$) and ($y(t)$ - $\mu_{\mathbf{B}}$) in Equation (2.27) by $\varepsilon_{\mathbf{A}}(t)$ and $\varepsilon_{\mathbf{B}}(t)$

$$J = \sum_{t=1}^{N} \varphi^2(t)\varepsilon_{\mathbf{A}}^2(t) + \left(1-\varphi(t)\right)^2 \varepsilon_{\mathbf{B}}^2(t) \tag{2.32}$$

where, $\varphi(t)$ is solved by

$$\varphi(t) = \varepsilon_B^2(t) \Big/ \left(\varepsilon_A^2(t) + \varepsilon_B^2(t)\right) \tag{2.33}$$

The new $\varphi$(t)is then converted to 0 and 1 by Equation (2.30) and the SVM is solved again. Subsequently, **a** and **b** are re-estimated.

This successive substitution procedure to solve the SRP is summarized in the following flowchart.

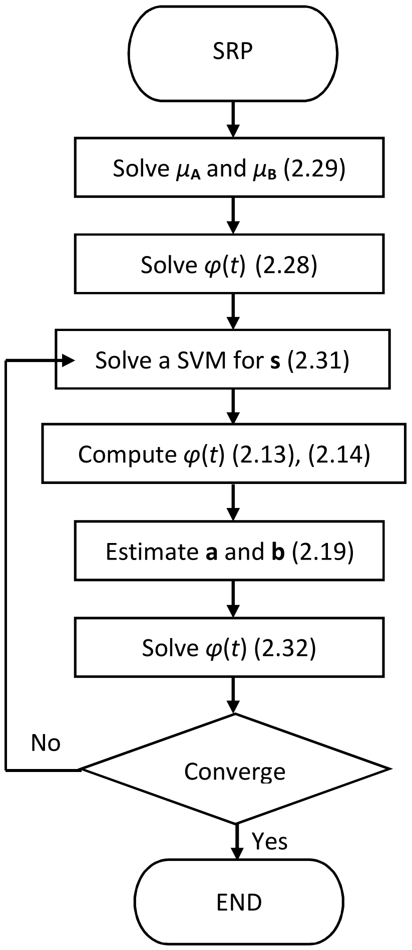

SRP
Solve $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ (2.29)
Solve $\varphi(t)$ (2.28)
Solve a SVM for **s** (2.31)
Compute $\varphi(t)$ (2.13), (2.14)
Estimate **a** and **b** (2.19)
Solve $\varphi(t)$ (2.32)
Converge
No
Yes
END

Figure 2.9 The algorithm solving the SPR

The algorithm stops when the change of $\varphi(t)$in two consecutive steps is very small. The antecedent space is progressively partitioned until no rule region can be further divided.

**C. Rule Antecedent Identification**

Given a partitioned antecedent space, the rule of each local region needs to be defined. This can be interpreted as the estimation of the centroid and shape factors of the best ellipsoids to cover each region. A method which also considers the quality of reach data point is used in the GTSK approach.

The quality is related to the prediction error for each data sample. The solid dots in Fig. 8 represent data points with relatively small residuals from their linear model, Equation (2.18), while the circles represent data points with relatively larger residuals.

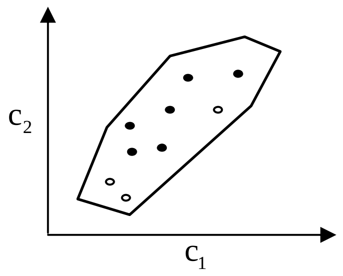


Figure 2.10 A local region in an antecedent space

Only data samples with smaller residuals are used to estimate the antecedent parameters. The importance of each data point is weighted by β, which is defined as

$$\beta_i^r = \exp\left(-N^r \left(\varepsilon_i^r\right)^2 \Big/ \left(\boldsymbol{\varepsilon}^r\right)^T \boldsymbol{\varepsilon}^r\right) \tag{2.34}$$

where $N^r$ is the number of data points in region $r$. The script ($r$, $i$) represents the $i$th data in region $r$. $\beta_i^r$ reaches the highest value at 1 when $\varepsilon_i^r$ is zero.

The centroid $\mathbf{o}^r$ is estimated by

$$\mathbf{o}^r = \sum_{i=1}^{N^r} \beta_i^r \mathbf{c}_i^r \Big/ \sum_{i=1}^{N^r} \beta_i^r \tag{2.35}$$

and the matrix $\mathbf{P}^r$ is defined by its inverse

$$\left(\mathrm{P}^r\right)^{-1} = \sum_{i=1}^{N^r} \beta_i^r \left(\mathbf{c}_i^r - \mathbf{o}^r\right)\left(\mathbf{c}_i^r - \mathbf{o}^r\right)^T \Big/ \sum_{i=1}^{N^r} \beta_i^r \tag{2.36}$$

Then the rules of local regions, TAs can be defined by Equation 2.5.

### 2.1.4 GTSK Modeling Procedure Summary

The above procedure for converting input-output data to a GTSK model is summarized as follows:

Step 1. Determine dynamic orders, *ny*, *nu* and delay *d* by using the SNNR to rearrange data, recursive estimation (Equation (2.11)) to process rearranged data, and the FPE ((Equation (2.12)) to evaluate a particular choice of regressor set.

Step 2. Determine antecedent variables ($c_1,...,c_{nc}$) from consequent variables ($x_1,\ldots,x_{nx}$)

Step 3. Recursively partition the antecedent space by solving a series of SRPs. Note, the parameter values in the consequent model were determined by Equation (2.19) in Step 3.

Step 4. Determine antecedent parameters, centroid and shape matrix for each rule antecedent.

## 2.2 NMPC Methodology

The NMPC proposed in this work is designed to find an optimized sequence of present and future controller outputs (u) to minimize the deviation between a process response prediction and a given reference trajectory for a number of future steps over a time horizon on the order of the process settling time.

The general structure of NMPC or receding horizon control implemented in this work is shown in Figure2.11

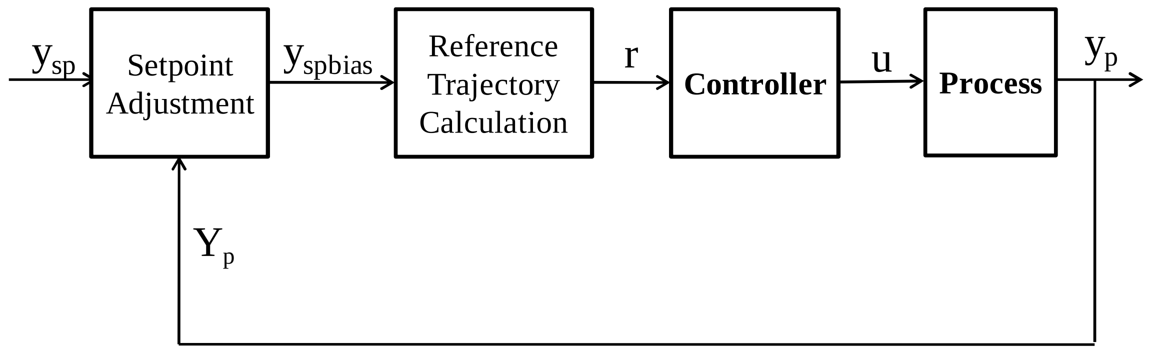


Figure 2.11 General Structure of NMPC

where $y_{sp}$ is the steady-state set point (SP) of controlled variable (CV) $y_p$, $y_{spbias}$ is the biased set point. u is the manipulated variable (MV). The controller block is comprised of the GTSK model and the optimizer.

In this NMPC application, the GTSK model is used to forecast the future process response based on the knowledge of past CVs and MVs. The function of the optimizer is to minimize the distance between the predicted future trajectory of process response and the desired future trajectory, which is also called reference trajectory.

At time t, the current time in the process operation, the process response prediction y(t+i) is calculated for each discrete sampling time for a long horizon into the future (i=1,....N). The length of the prediction horizon, N, is a controller tuning variable defined by user. However, it should be extend sufficiently beyond the dead-time of the process, θ, as well as the inverse process range. The predicted process response is determined by a nonlinear model of the NMPC, which is the GTSK model in this work.

Since a process model mismatch (pmm) exists, a feedback correction is necessary for compensating the pmm and removing the steady state offset. In this work, a biased set point is introduced as the adjustment. It is defined by Equation (2.37), where pmm stands for the difference between the present process output and model prediction.

$$y_{spbias} = y_{sp} - pmm \tag{2.37}$$

The reference trajectory, r(t) applied in this work is a first-order approximation from the current model value, *y(t)* towards a biased set point, $y_{spbias}(t)$. It is defined by

$$\tau_w \frac{d\,r(t)}{dt} + r(t) = y_{spbias}(t) \tag{2.38}$$

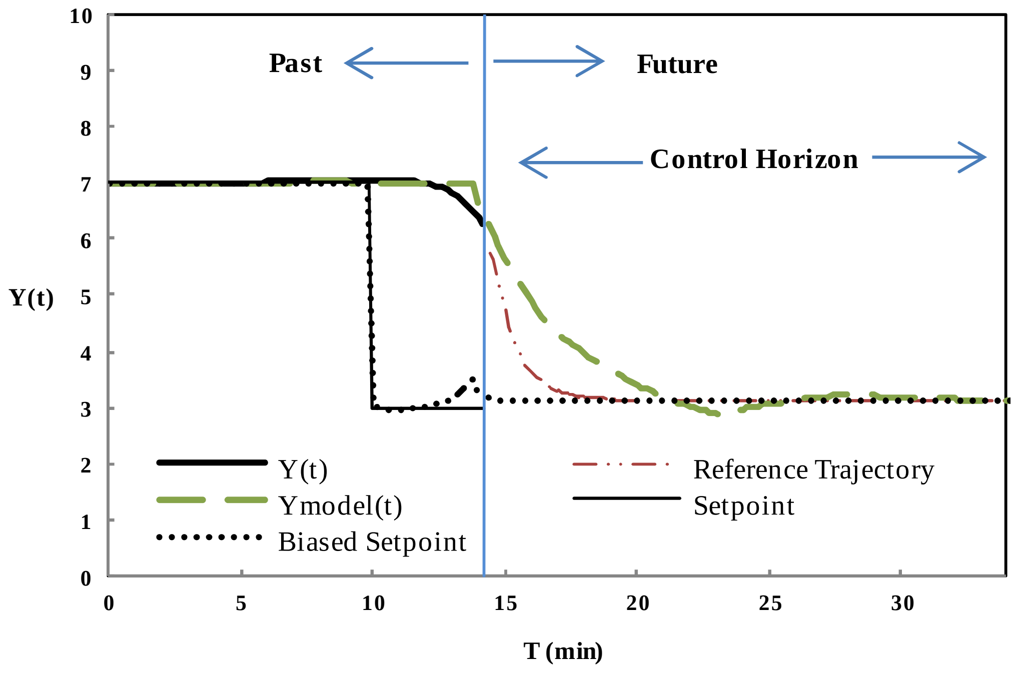
which can be also represented in a discrete time formation as below

$$r_{t+1} = (1 - e^{-\Delta t/\tau_w})\, y_{spbias\ t} + (e^{-\Delta t/\tau_w})\, r_t \tag{2.39}$$

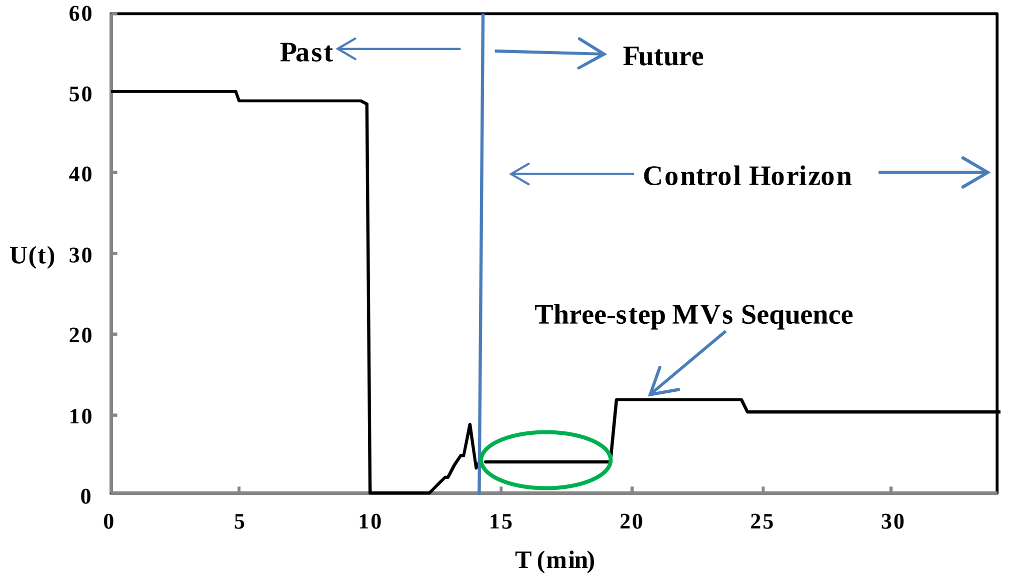
where $\Delta t$ is the sampling time interval, $\tau_w$ is a tuning parameter of the reference trajectory.

In this NMPC control scheme, the controller output scenario (the future sequence of MV values) is “known” within the control horizon to minimize the sum squared deviation (SSD) of process response prediction from reference trajectory.

The controller output scenario in this work is designed to be a sequence of three step-and-hold values. Each of them covers 25%, 25% and 50% of the control horizon, respectively. Nevertheless, only the first step of the MV sequence is implemented as the controller output at instant time, t. Figure 2.12 illustrates an example of the MV sequence and process response prediction in the future. In this simulation, the process is initialized at the steady state of the controlled variable (CV) =7. The controller is in the MAN mode with the MV of 50% at the beginning. It switches to AUTO mode at 5 Minutes. At t=10, the set point is changed from 7 to 3. The MV goes to zero initially after the set point change. Figure 2.12 is a snapshot at t=14 which shows post events and future projections.



(a)



(b)

Figure 2.12 The Process Response and the Prediction (a) and the MV sequence (b)

The nonlinear model embedded in the optimizer is the GTSK model. At the very beginning of the control horizon, the GTSK model is initialized using the past CV and MV values as the regressors. Once a model prediction value is calculated by the GTSK model for the next time step, it is stored as a "past CV" value for next sampling time prediction. The controller output values used as the regressors are delivered from the trial solution of MVs, which is determined by optimizer. The estimated process response for the whole range of the control horizon is calculated by this recurrent prediction approach.

A new optimization technique called Leapfrogging [6] is employed in this work. With a global aspect, the leapfrogging technique can efficiently find the global optima for multi-optima problems. In this work, it searches and determines the MV sequence that minimizes the sum of squared deviations between a process response prediction and a given reference trajectory for the control horizon at each sampling time.

The logic of the NMPC algorithm is illustrated in the following flow chart.

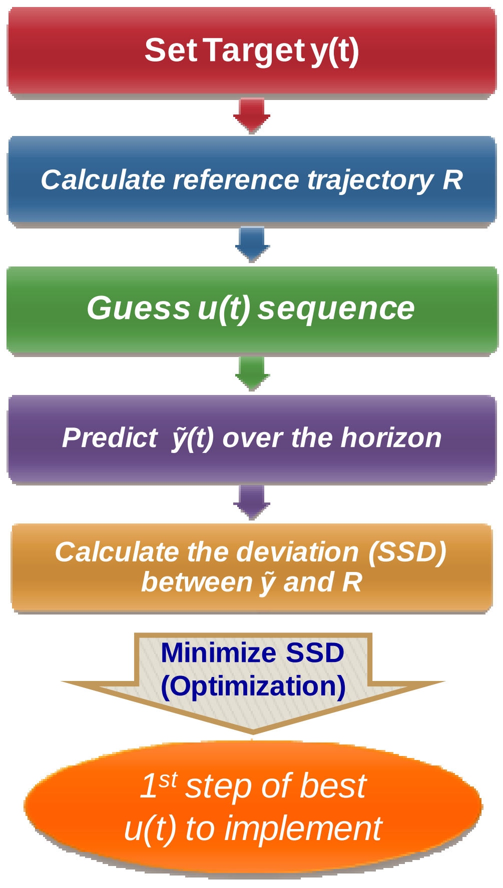

Set Target y(t)
Calculate reference trajectory R
Guess u(t) sequence
Predict ỹ(t) over the horizon
Calculate the deviation (SSD) between ỹ and R
Minimize SSD (Optimization)
1st step of best u(t) to implement

Figure 2.13 The Logic of the NMPC Algorithm

## 2.3 Leapfrogging Optimization Approach in GTSK MPC

In a MPC controller, a set of optimal MV values which drive the process toward the desired set point along the reference trajectory without violating constraints need to be computed at each sampling time.

Recently, a new optimization approach named leapfrogging is proposed by Rhinehart, Su and Sridhar (2010)[6]. With the characteristics of direct search and random starts, it can efficiently find the global optima for multi-optima problems. Considering its simplicity and efficiency for global optimization, which will greatly enhance the online application performance of the NMPC controller, the leapfrogging optimizer is employed in this work.

The leapfrogging technique starts with a random located set of trial solutions (termed players) within the feasible decision variable (DV) space. Infeasible regions may be defined by either DV, OF or auxiliary functions. At each iteration, the player with the worst objective function (OF) value is relocated to a random position within its DV-space reflection on the other side of the player with the best OF value. Figure 2.1 illustrates the leapfrogging mechanism.

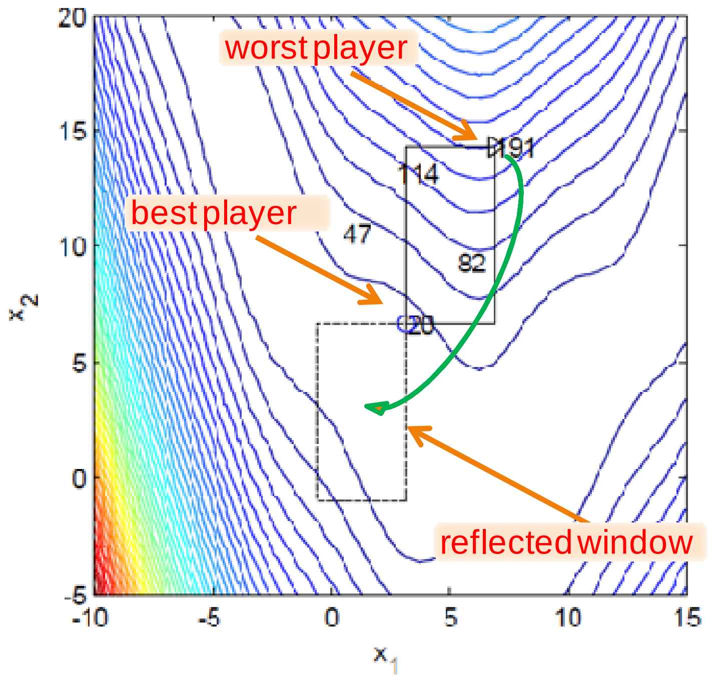


Figure 2.14 Leapfrogging Mechanism

The repulsion of vacated sites is another innovative attribute of this technique. Influenced by the synoptic topographic knowledge, a history of positions of recently vacated sites in DV-space is used to mildly repulse the leap-to position away from the recently vacated sites.

In this work, the controller calls the embedded leapfrogging optimizer to search and determine the MVs sequence that minimizes the sum of squared deviations between a process response prediction and a given reference trajectory for the control horizon at each sampling time.

# CHAPTER III

## EXPERIMENTAL METHOD

### 3.1 Experimental System

In this study, the experimental system is a process simulator, in which the NMPC controller was implemented on a primary process (y) and an auxiliary process (z). The primary process in this simulator is fourth-order-plus-dead-time (FOPDT) process with a nonlinear gain and the environmental effect (noise and disturbance). The process gain ($K_p$) is one over the square root of the manipulated variable *u(t),* which is also the input of the controlled variable *y(t).*

The FOPDT process is represented as

$$\begin{aligned} K_p &= 1/\sqrt{u(t)} \\ \tau_1 \frac{d\,y_1(t)}{dt} + y_1(t) &= K_p\, u(t) \\ \tau_2 \frac{d\,y_2(t)}{dt} + y_2(t) &= y_1(t) \\ \tau_3 \frac{d\,y_3(t)}{dt} + y_3(t) &= y_2(t) + D \\ \tau_4 \frac{d\,y(t)}{dt} + y(t) &= y_3(t - \theta) \end{aligned} \tag{3.1}$$

where *$y_i(t)$* is the process response of each lag, $\tau_i$ is the time constant, *D* is the external disturbance, $\theta$ is the dead-time, *dt* is the sampling interval. The coefficient values are shown in Table 3.1

Table 3.1 Process Coefficients

| **Parameter** | **Value (minute)** |
|---|---|
| $\tau_1$ | 1 |
| $\tau_2$ | 2 |
| $\tau_3$ | 2 |
| $\tau_4$ | 3 |
| $\theta$ | 3 |
| $dt$ | 0.2 |

Figure 3.1 shows the nonlinear dynamic behavior of the process. The input u(t) has a minimum value of 0 and a maximum value of 100. Consequently, the process response y(t) ranges from 0 to 10. In Figure 3.1, a step change of u(t) from 0 to 20 leads to a y(t) change of about 4.5. However, y(t) increases from about 9 to 10 when a same increment was made on u(t) from 80 to 100. Due to the nonlinearity of the process, equal changes in u(t) lead to diminishing changes in y(t).

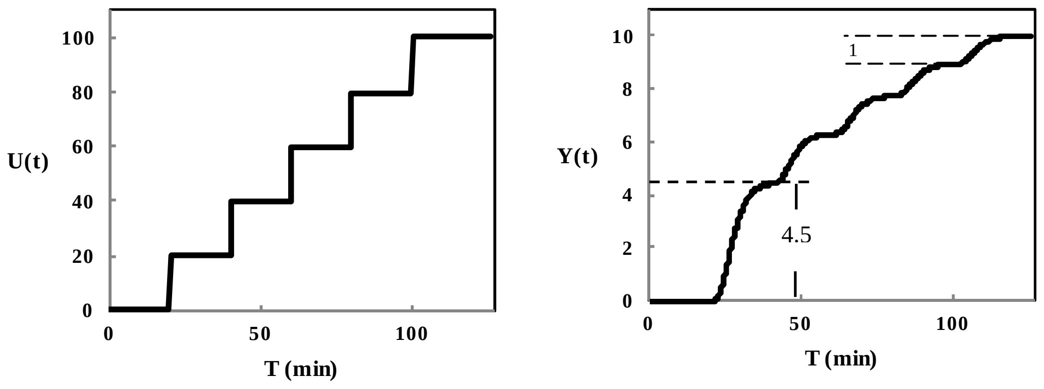


Figure 3.1 CV responses to MV equal-step-changes

The noise used in the simulation is generated by Box-Muller method[34]. It is normally and independently distributed with zero mean and a standard deviation of sigma (σ), which is referred to as NID (0, sigma). The noise equation is

$$NID\,(0,\sigma_{\text{Noise}}) = \sigma_{\text{Noise}}\sqrt{-2\,Ln(r_1)}\;\cdot sin(2\pi r_2) \tag{3.2}$$

where $\sigma_{\text{noise}}$ is 0.2, $r_1$ and $r_2$ are uniformly distributed random number between 0 and 1.

The external disturbance is simulated by an autoregressive-moving-average (ARMA) type model which is first order in response. The model that driven by a NID(o, $\sigma_{\text{Dist}}$ ) noise signal generates autocorrelated time series data as disturbance. It is represented as

$$\tau_{\text{Dist}}\frac{d\,D(t)}{dt}+D(t)= NID\,(0,\sigma_{\text{Dist}}) \tag{3.3}$$

Where *D(t)* is the disturbance, $\tau_{Dist}$ is 5, $\sigma_{\text{Dist}}$ is 0.25.

The auxiliary process variable z(t) is a first-order response to u(t), which is represented as

$$\tau_z\frac{d\,z(t)}{dt}+z(t)=K_z\;u(t) \tag{3.4}$$

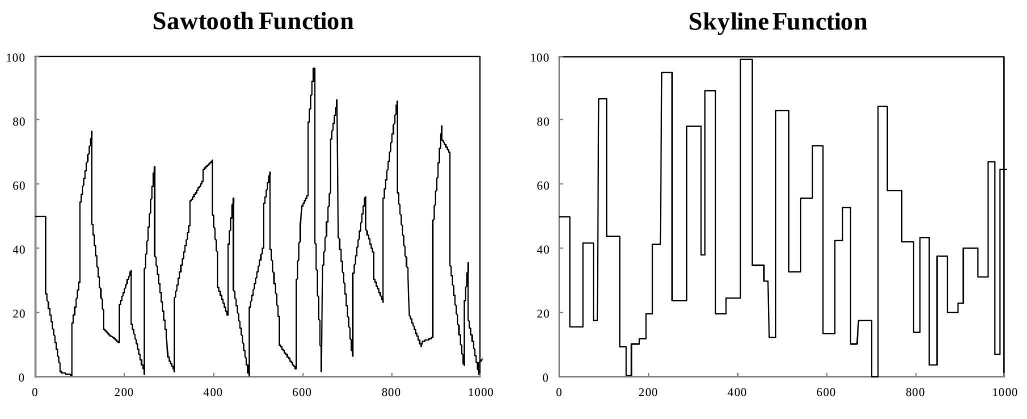
where $\tau_z$ is 2.67, $K_z$ is 0.9.

## 3.2 GTSK Model Development

The GTSK approach converts the time varying input and output data generated by the process simulator to a GTSK model. As presented in Chapter 2, *y(t-1) … y(t-ny), u(t-d) …. u(t-nu-d)* were determined as the consequent variables, and then the antecedent variables were selected from them. The antecedent space was recursively partitioned into several regions. The antecedent parameters, centroid and shape matrix were determined by GTSK modeling technique subsequently.

### 3.2.1 Sawtooth Input Training Signal

In the modeling phase, a "sawtooth function" is chosen to generate the input training signal. As shown in Figure 3.2, it jumps or drops to the halfway of a random level, and then ramps to that level at the end of a period of random length."Skyline function" is another commonly used training signal, which jumps or drops to a random level and holds for a random time interval within certain limits. Compared to "skyline function", the sawtooth function covers wider range of the input data by the random located ramp, and is thought to provide a more complete basis for obtaining empirical process response models.

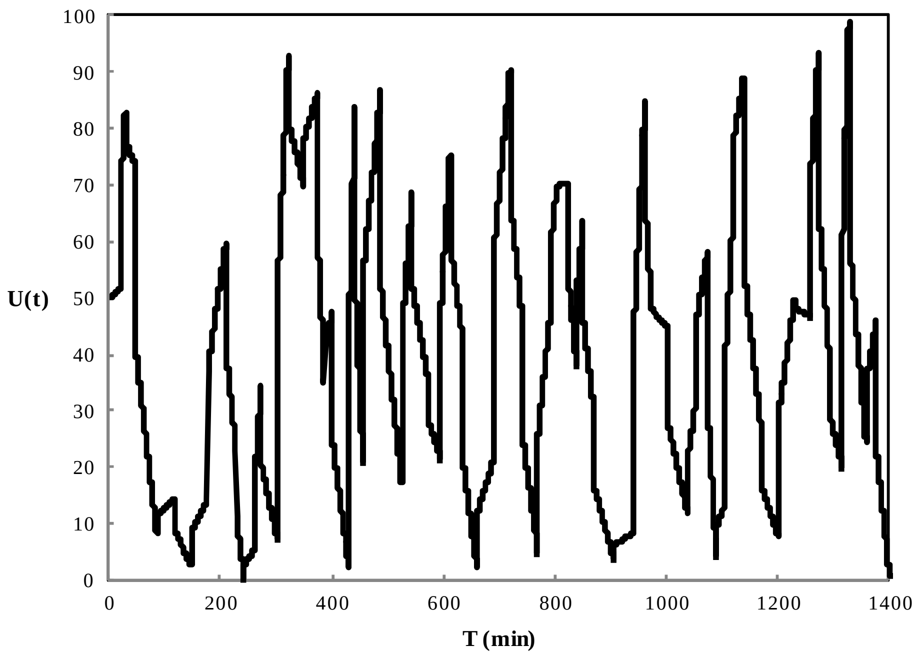


(a) (b)

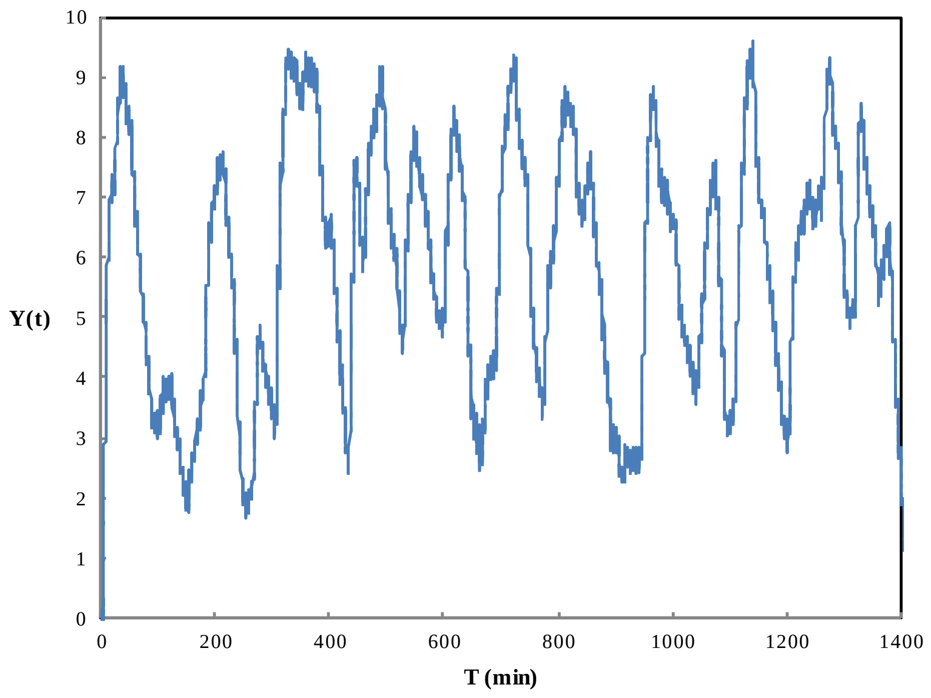
Figure 3.2 Input training signals

### 3.2.2 GTSK Model Generation and Evaluation

The training data used in this work is plotted in Figure 3.3, which shows the input signal u(t) by the sawtooth function and the process response y(t). The range of process response value is 0 to 10, while the range of input signal value is 0 to 100%. The process was initialized at zero.



(a)



(b)

Figure 3.3 The sawtooth-pattern input (a) and process response (b)

The GTSK model in this work has y(t-1), y(t-2), y(t-3), and u(t-20) as regressors, among which u(t-20) is the only antecedent variable. The model, which has 12 local models, is represented by following equations. Table 3.2 shows the values of 84 coefficients in total.

$$\hat{y}^{1}(t) = \theta_1^1 + \theta_2^1\, y(t-1) + \theta_3^1\, y(t-2) + \theta_4^1\, y(t-3) + \theta_5^1\, u(t-20)$$

M

$$\hat{y}^{12}(t) = \theta_1^{12} + \theta_2^{12}\, y(t-1) + \theta_3^{12}\, y(t-2) + \theta_4^{12}\, y(t-3) + \theta_5^{12}\, u(t-20)$$

$$TA^i(t) = EXP\left(\frac{-(u(t-20) - Center^i)^2}{Width^i}\right) \qquad (3.5)$$

$$w^i(t) = \frac{TA^i(t)}{\sum_{i=1}^{12} TA^i(t)}$$

$$\hat{y}(t) = \sum_{i=1}^{12} w^i(t)\, \hat{y}^i(t)$$

Table 3.2 GTSK Model Coefficients

| Model No. | $\theta_1$ | $\theta_2$ | $\theta_3$ | $\theta_4$ | $\theta_5$ | *Center* | *Width** |
|---|---|---|---|---|---|---|---|
| 1 | 0.380729 | 0.374544 | 0.279718 | 0.276694 | 0.002959 | 76.5624 | 0.0197 |
| 2 | 0.198491 | 0.289056 | 0.38512 | 0.259525 | 0.005471 | 56.4061 | 0.0614 |
| 3 | 0.03796 | 0.412064 | 0.185464 | 0.335465 | 0.023535 | 3.8624 | 0.2644 |
| 4 | 0.374987 | 0.30784 | 0.399311 | 0.188244 | 0.00371 | 16.8419 | 0.4129 |
| 5 | 0.318991 | 0.219484 | 0.401633 | 0.30176 | 0.004533 | 47.916 | 1.2644 |
| 6 | 0.190683 | 0.184169 | 0.401888 | 0.337867 | 0.004275 | 8.2903 | 0.9606 |
| 7 | 0.129525 | 0.371462 | 0.27109 | 0.285711 | 0.008859 | 30.4761 | 0.1612 |
| 8 | 0.254841 | 0.348019 | 0.301008 | 0.279506 | 0.003807 | 23.3493 | 0.2459 |
| 9 | 0.234811 | 0.404955 | 0.225322 | 0.310266 | 0.003447 | 44.0831 | 0.3508 |
| 10 | 0.074481 | 0.334408 | 0.327454 | 0.263228 | 0.010187 | 38.0753 | 0.377 |
| 11 | 0.039623 | 0.242396 | 0.305287 | 0.316081 | 0.035061 | 13.3275 | 2.5599 |
| 12 | -0.04174 | 0.300432 | 0.416675 | 0.229603 | 0.018498 | 11.3618 | 3.0833 |

In this work, because some of the width values were so small that there was no model information in between centers, all the width values are adjusted to be 30% to provide a better coverage on the antecedent of each local model.

The GTSK model prediction, $\hat{y}_{GTSK}$, is shown in Figure 3.4; comparing it with the process response in the same plot. The two curves are barely distinguishable. To provide a comparison, a second-order-plus-dead-time (SOPDT) model is generated by the conventional regression approach, which the process-model mismatch is obvious, indicating the benefit of the GTSK model. Figure 3.5 shows the SOPDT model prediction $\hat{y}_{SOPDT}$.

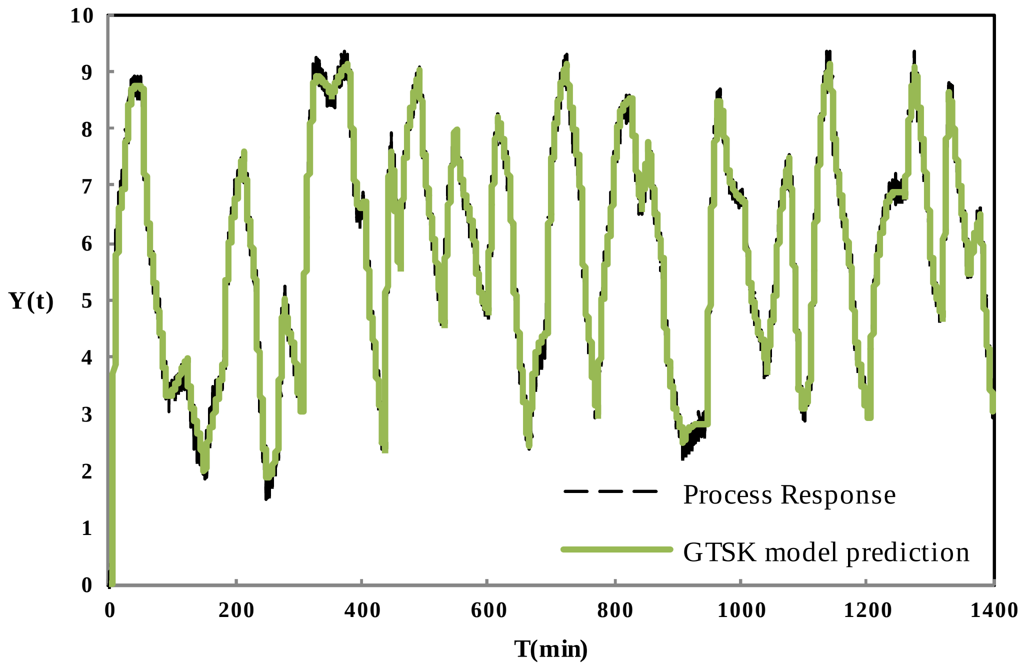


Figure 3.4 GTSK model prediction and process response

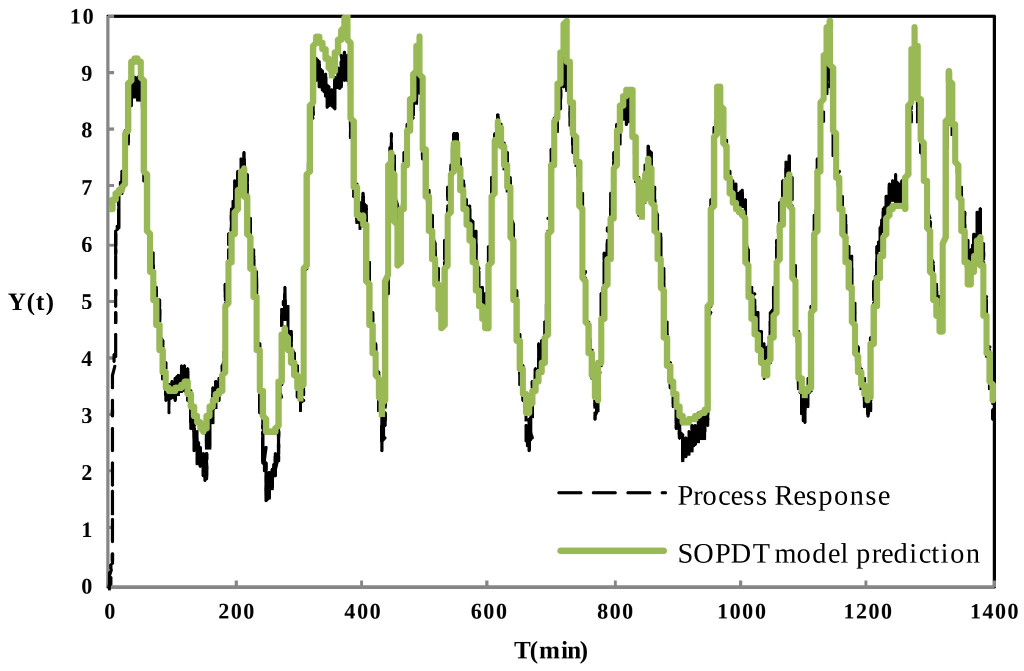


Figure 3.5 SOPDT model prediction and process response

The model performance is evaluated by the sum squared deviation (SSD) from the process response y to the prediction ŷ. The SSD from model to process of two models was compared in Table 3.3

Table 3.3 Comparison of sum-squared deviation (SSD)

| | GTSK model | SOPDT model |
|---|---|---|
| SSD | 159 | 2057 |

According to both the SSD and plot comparison, the GTSK model prediction not only shows a much smaller deviation from the process than the SOPDT model, but also fits the process “very well”. This result indicates that the GTSK model is a well-qualified model for the nonlinear MPC.

# CHAPTER IV

# RESULTS AND DISCUSSION

To evaluate the performance of a GTSK MPC controller, seven sets of dynamic control simulations were run. The performance of disturbance rejection, set point tracking, constraint handling, comprehensive environmental effect (both noise and disturbance) handling and manual to automatic transfer was tested. In all simulations, the process was initialized at the steady state of CV=7 with the MV of 50%. It operates in the MAN mode for 5 minutes before it was transferred to the AUTO mode (except the disturbance rejection test).

## 4.1 Disturbance Rejection

The GTSK-MPC control performance for the disturbance rejection is illustrated in Figure 4.1. Set point is maintained at 7, and noise was not introduced to the process. The disturbance was added to the second-order response of the process. To demonstrate the controller ability to handle the disturbance, the test was carried out under both MAN and AUTO modes of the controller. From 0 to 40 minutes, the controller is in MAN mode with the MV of 50%. After the disturbance was removed, the process was not back to the set point. After the time of 40 minutes, the controller was switched to AUTO mode, and trying to bring the disturbed process back to set point. It is shown that the process was regulated successfully and brought back to the set point at about 80 Minutes. The control performance was measured by the sum squared deviation (SSD) of

CV from set point. The SSD is 21.85 when the controller is in MAN mode. Within the next 40 minutes, when it is in AUTO mode, the SSD is reduced to 5.82.

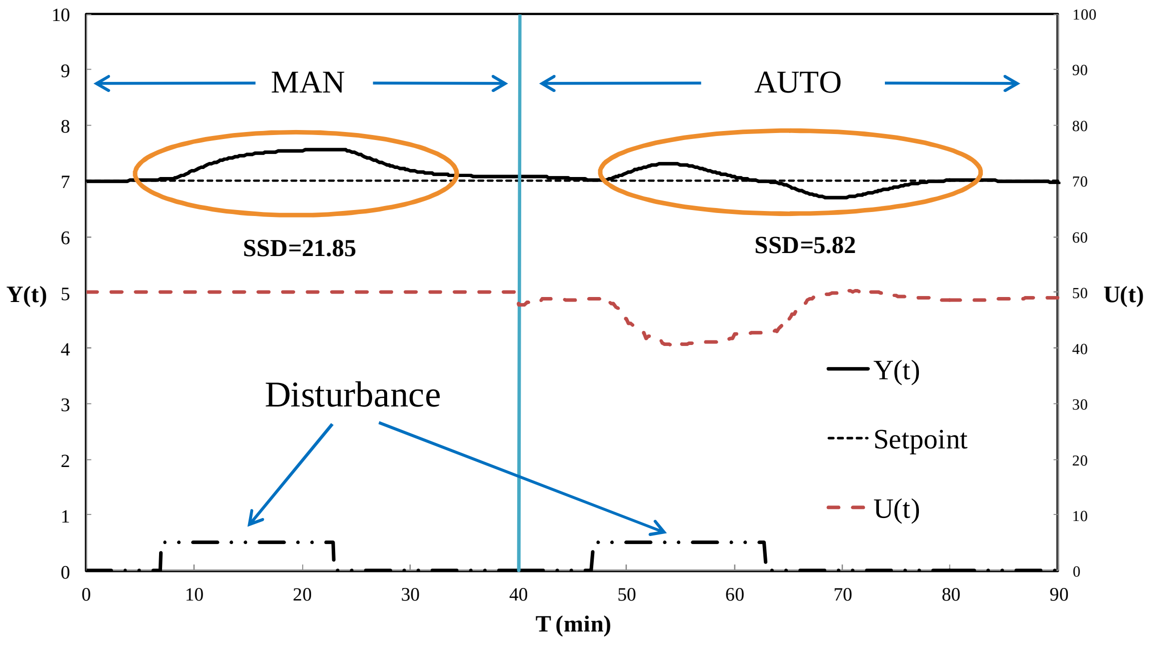


Figure 4.1 Control performance of disturbance rejection

## 4.2 Set point Tracking

Figure 4.2 illustrates the control performance for set point tracking.  In this test, there is no environmental effect added to the process. The controller showed a strong set point tracking ability. It took the process about 30 minutes to settle down when the set point stepped either up or down. The controller also responded quickly and effectively with a moderate aggressiveness. In spite of process nonlinearity ($K_p$ changes by about 2:1 over the range), the process responses to the SP = 3 and SP =9 values are similar, demonstrating control effectiveness.

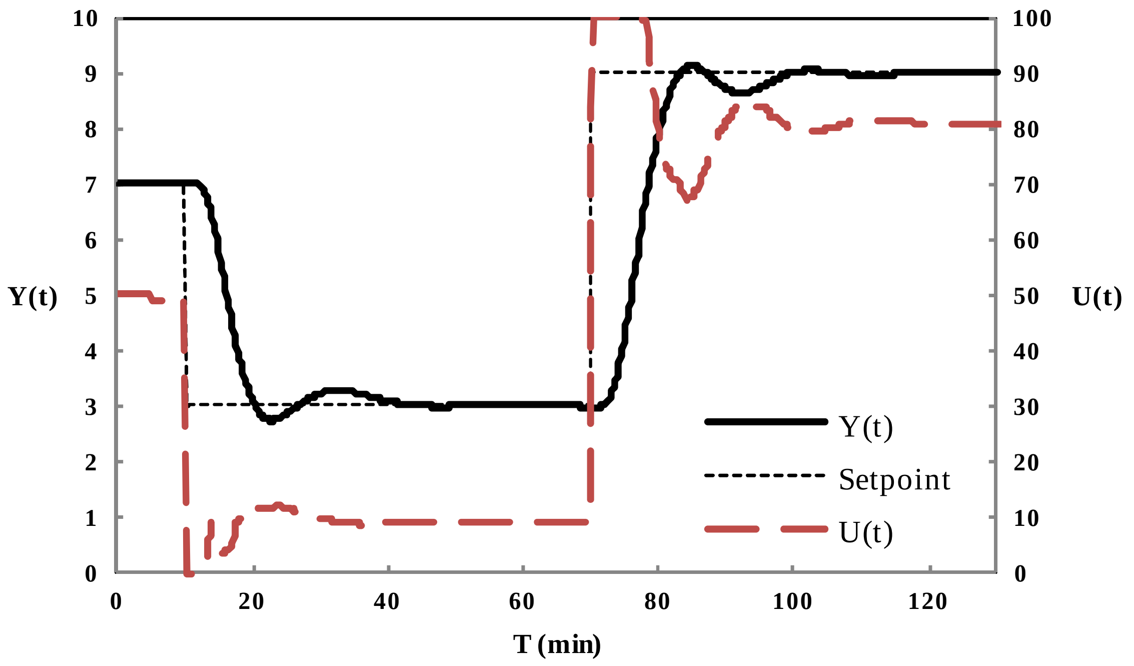


Figure 4.2 Control performance for set point tracking without environmental effects

## 4.3 Constraint Handling

Figures 4.3 to 11 illustrate the impact of the different types of constraints. In each test, two sets of set point step change from 7 to 3 were made to compare the constrained and unconstrained conditions. During the first simulation period from 0 to70 minutes, the process was simulated with the constraints. Then the constraints were removed at 70 minutes. The second parts of the simulation were carried out without the constraint. The four types of constraints were all tested individually.

Figure 4.3 shows the control performance when hard constraints were introduced. In this test, the MV was constrained with an upper limit of 100% and a lower limit of 10%. At about 10 minutes, the set point was changed to 3, and the controller began to push the MV to its lower limit of 10% trying to bring the CV to the set point. The CV reached the steady state value which is greater than the set point at about 40 minutes. As soon as the set point returned to 7, t=70, the

MV changes. There was no windup at the constraint. Once the constraint was removed, the MV could go below 10% so that the CV reached the set point of 3.

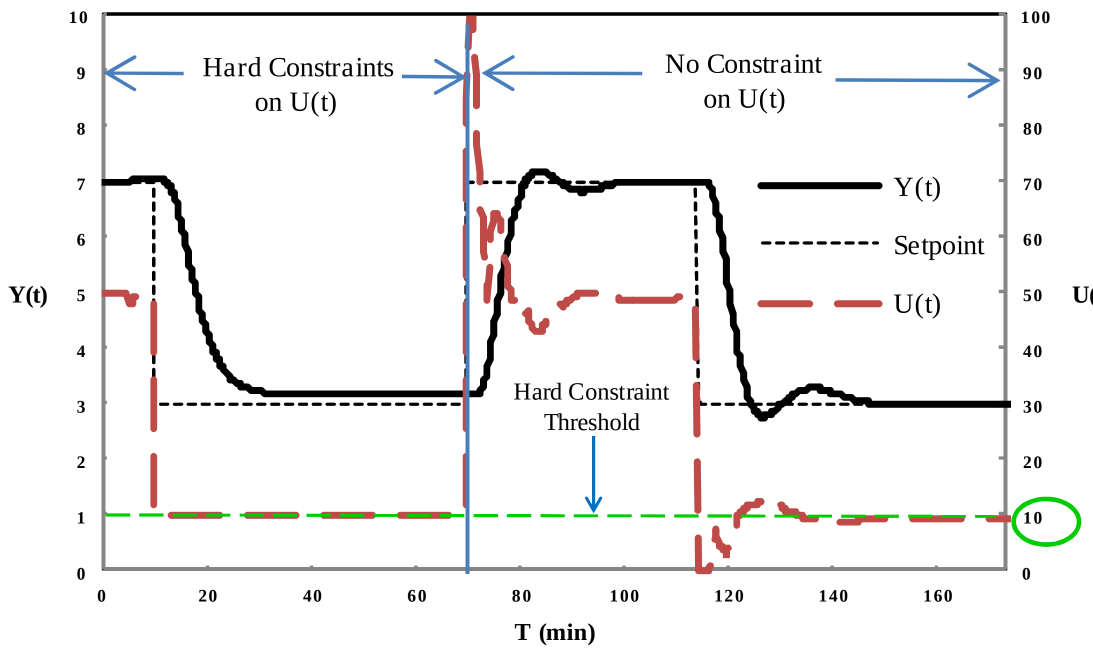


Figure 4.3 Demonstration of the impact of hard constraints on the MV

Figure 4.4 is a demonstration of the impact of hard constraints on the rate-of-change of MV. The limit on the rate-of-change of MV is 1% per sampling interval. Under the constrained condition, the MV gradually moved down after the set point change. In contrast, during the subsequent unconstrained period, the MV jumped straight down immediately when the set point was changed.

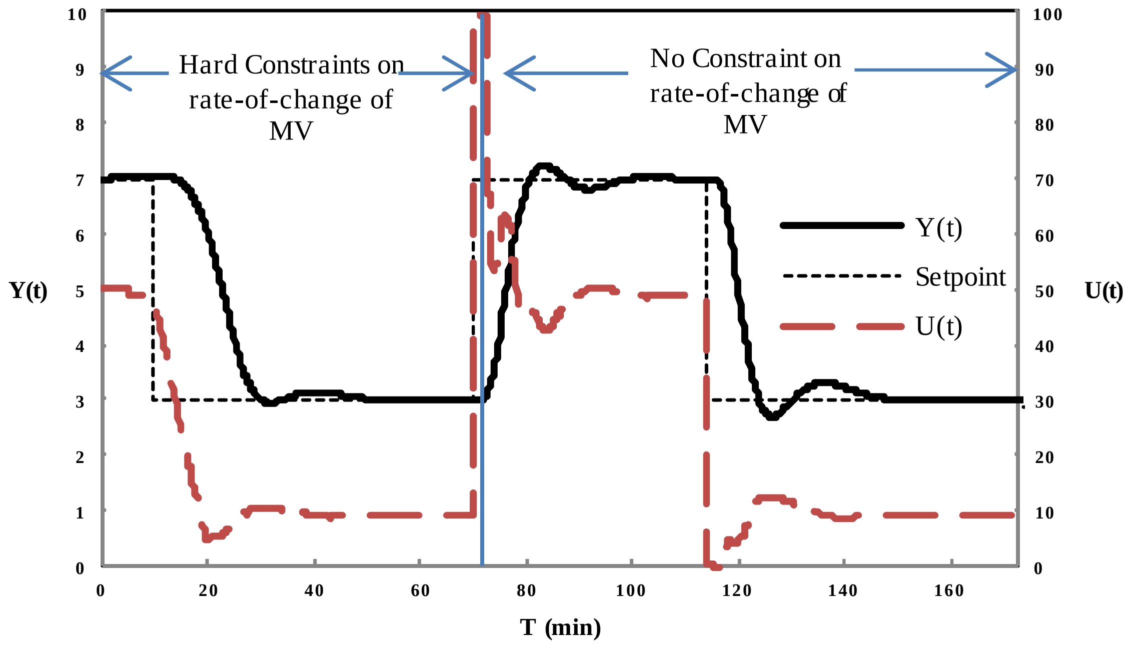


Figure 4.4 Demonstration of the impact of hard constraints on the rate-of-change of MV

Figure 4.5 demonstrates the impact of soft constraints on the value of the CV. The “legal” operation range of the CV was set as 3.2 to 10. If the CV violates the constraints, a penalty is added to SSD so that the optimizer is subject to the soft constraints on the CV. Comparing the different process responses to the same set point step change, Figure 4.5 shows that, under the constrained condition, the controller was not able to drive the CV to the set point 3, which is below the lower limit 3.2. Therefore, the soft constraint successfully influenced the controller’s “decisions”.

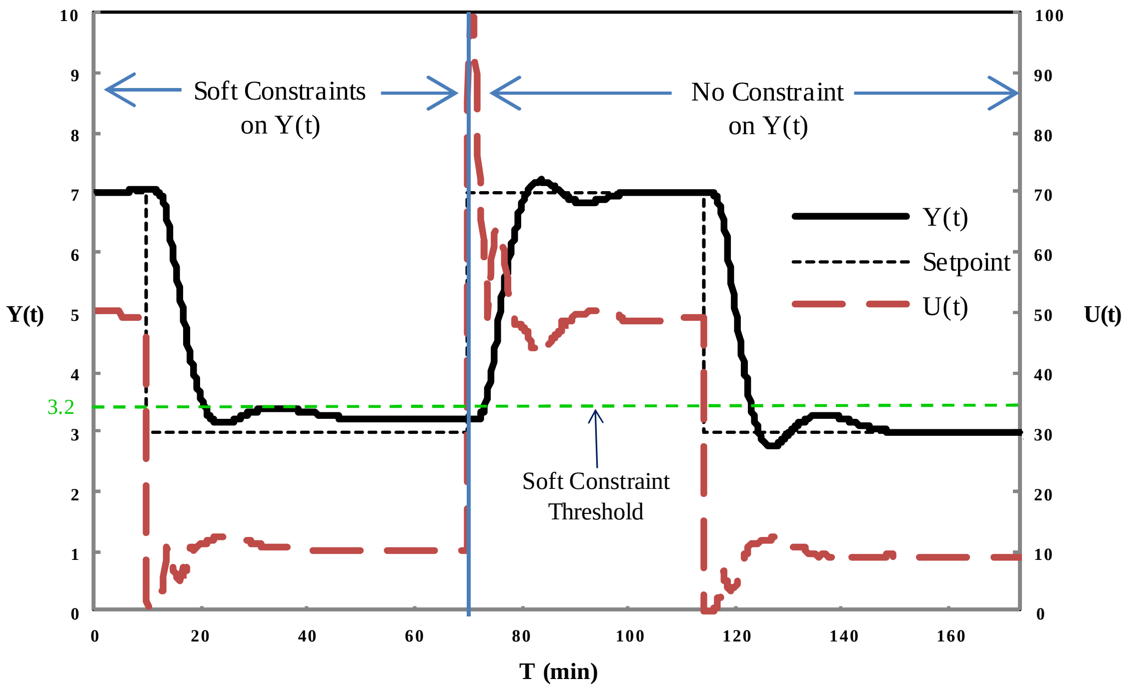


Figure 4.5 Demonstration of the impact of soft constraints on the CV

The auxiliary variable was added to the process in the test shown in Figure 4.6. The lower limit of the auxiliary variable is 15. The optimizer is subject to the soft constraints on the auxiliary variable. It is shown that the controller sacrificed its performance to avoid violating the soft constraints on the auxiliary variable. Due to the optimization objective, the CV did not reach the set point, but achieved a balance between the control performance and the penalty of violating the soft constraints.

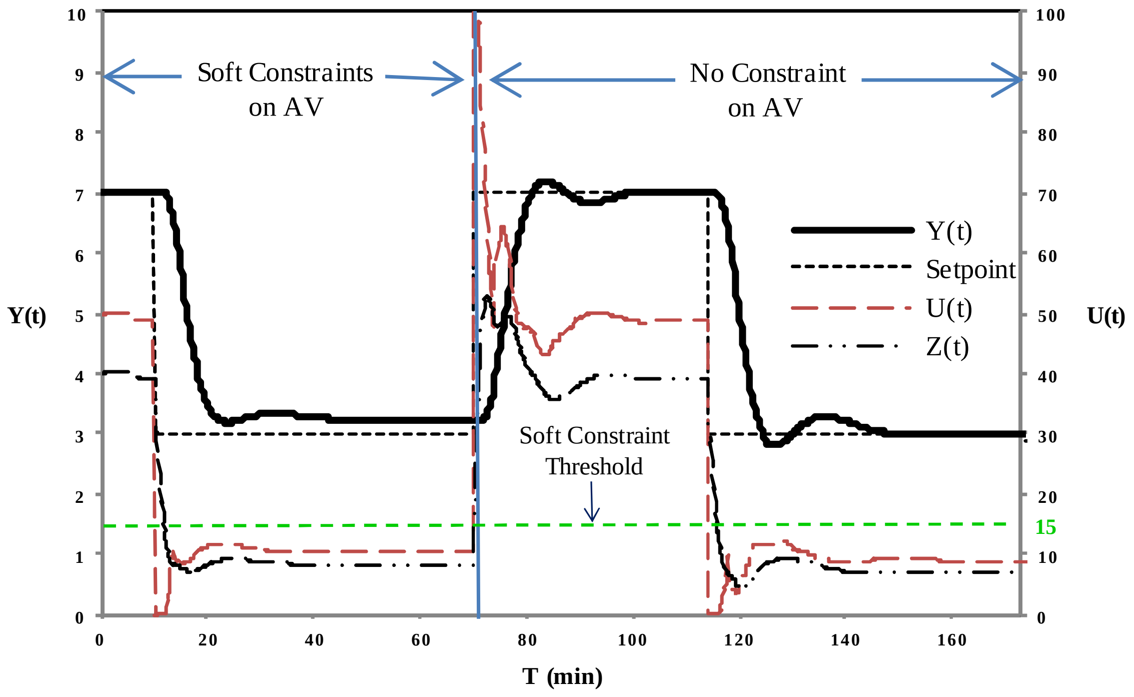


Figure 4.6 Demonstration of the impact of soft constraints on the auxiliary variable

## 4.4 Environmental Effect Handling

When environmental effects are added to the process, the control performance for handling this comprehensive situation is shown in Figure 4.7. The four types of the constraints presented above are also applied in this test. In Figure 4.7, the process tracks the set point change well even under the influence of both noise and disturbance. The disturbance was introduced at 7 minutes. It is shown that there is same amplitude of the noise and wandering of CV at both set point = 3 and set point = 9. However, the amplitude of MV wandering at set point = 3 is about twice larger than that at set point = 9. This is attributed to the nonlinearity of the process.

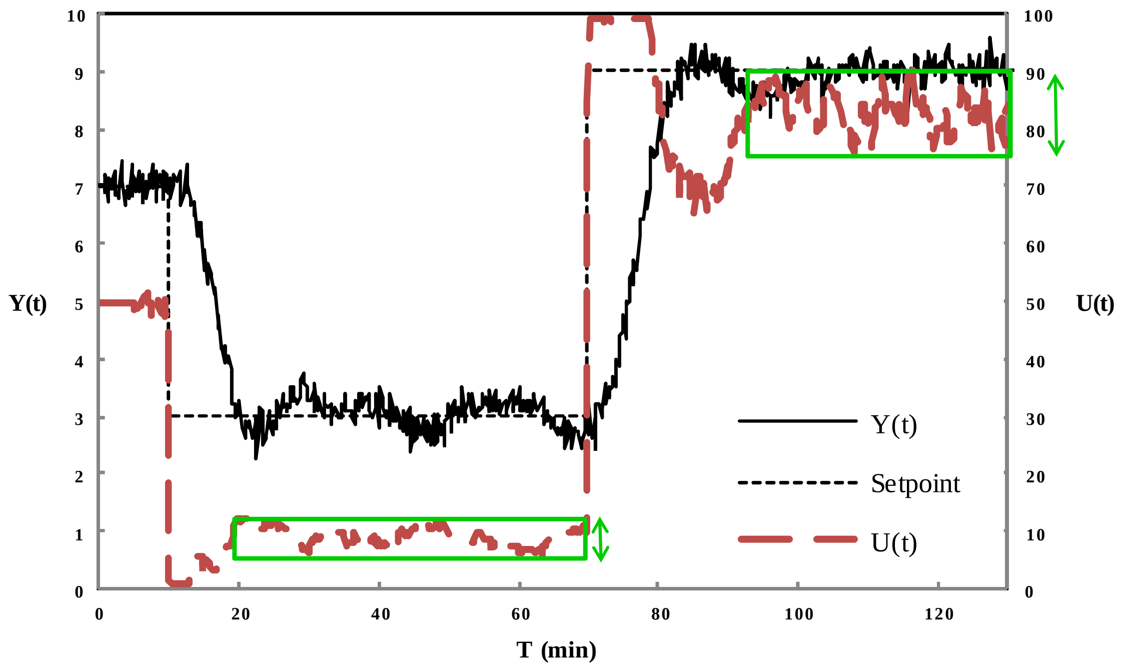


Figure 4.7 Control performance for set point tracking with environmental effects

# CHAPTER V

# CONCLUSION AND RECOMMENDATIONS

## 7.1 Conclusion

1. The use of a GTSK model and Leapfrogging as an optimizer were demonstrated as effective for nonlinear model predictive control.

2. The nonlinear model is firstly developed by using GTSK approach. The prediction accuracy of the GTSK model was illustrated and quantified by a comparison with SOPDT model. The GTSK model was much better.

3. A fourth-order-plus-dead-time (FOPDT) process simulator with nonlinear gain and the environmental effect (noise and disturbance) is used as the experimental system, in which the NMPC control algorithm was embedded. The SIMO process was subject to soft constraints on the controlled and auxiliary variables, and hard constraints on both the limits and rate of change of the manipulated variable.

4. The performance of GTSK MPC controller is evaluated via seven sets of dynamic control simulation. The controller showed desirable performance for disturbance rejection, set point tracking, constraint handling, and comprehensive environmental effect handling.

5. The controller has a bump-less transition from MAN to AUTO mode.

6. The controller does not wind up when the process was constrained.

## 7.2 Recommendations

Recommendations for future work are:

1. Extend the use of GTSK model in MPC to the multi-input-multi-output (MIMO) process simulator with a more realistic and representative chemical engineering unit operation process.

2. Implement the GTSK MPC controller on the real system, like the heat exchanger or distillation column in the Unit Operations Laboratory.

3. Investigate the reason GTSK approach gives such low width of the rule antecedent.

4. Investigate the merits of the sawtooth signal over the skyline signal.

5. Evaluate the use of the GTSK model for control applications to alternate nonlinear modeling approaches. Consider ease of use, computational burden, robustness, understandability, and other technical and human attributes.

## REFERENCES

[1] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," 2002.

[2] J. Qin and T. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice,* vol. 11, pp. 733-764, 2003.

[3] E. Camacho and C. Bordons, "Nonlinear Model Predictive Control: An Introductory Review," ed, 2007, pp. 1-16.

[4] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems,* vol. 28, pp. 15-33, 1988.

[5] M. Su and R. R. Rhinehart, "A generalized TSK model with a novel rule antecedent structure: Structure identification and parameter estimation," *Computers &amp; Chemical Engineering,* vol. 34, pp. 1199-1219, 2010.

[6] M. S. a. U. M. S. R. Russell Rhinehart, "Leapfrogging and Synoptic Leapfrogging – a new optimization approach," *Submitted to: European Journal of Operations Research,* 03-23 2010.

[7] S. J. Qin and T. A. Badgwell, "An overview of nonlinear model predictive control applications," *Nonlinear Model Predictive Control,* pp. 369-392, 2000.

[8] J. Ou, "Grouped neural network model-predictive control and its experimental distillation application," Oklahoma State University Ph.D., Oklahoma State University, United States -- Oklahoma, 2001.

[9] J. Ou and R. R. Rhinehart, "Grouped-neural network modeling for model predictive control," *ISA transactions,* vol. 41, pp. 195-202, 2002.

[10] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural network design*: PWS Pub., 1996.

[11] M. Azlan Hussain, "Review of the applications of neural networks in chemical process control--simulation and online implementation," *Artificial Intelligence in Engineering,* vol. 13, pp. 55-68, 1999.

[12] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, pp. 359-366, 1989.

[13] G. Feng, "A survey on analysis and design of model-based fuzzy control systems," *Fuzzy*

*Systems, IEEE Transactions on,* vol. 14, pp. 676-697, 2006.

[14] T. Takagi and M. Sugeno, "Fuzzy identification of system and its applications to modelling and control," *Äj,* vol. 2, p. 5.

[15] B. Kosko, "Fuzzy systems as universal approximators," *IEEE transactions on computers,* pp. 1329-1333, 1994.

[16] P. Georgieva and S. Feyo de Azevedo, "Neural networks for model predictive control," 2011, pp. 111-118.

[17] R. Al Seyab and Y. Cao, "Differential recurrent neural network based predictive control," *Computers & Chemical Engineering,* vol. 32, pp. 1533-1545, 2008.

[18] J. Ou and R. R. Rhinehart, "Grouped neural network model-predictive control," *Control Engineering Practice,* vol. 11, pp. 723-732, 2003.

[19] H. Eliasi, H. Davilu, and M. Menhaj, "Adaptive fuzzy model based predictive control of nuclear steam generators," *Nuclear engineering and design,* vol. 237, pp. 668-676, 2007.

[20] P. K. Roy, G. K. Mann, and B. C. Hawlader, "Fuzzy rule-adaptive model predictive control for a multivariable heating system," 2005, pp. 260-265.

[21] Y. Huang, H. H. Lou, J. Gong, and T. F. Edgar, "Fuzzy model predictive control," *Fuzzy Systems, IEEE Transactions on,* vol. 8, pp. 665-678, 2000.

[22] M. Mahfouf, D. Linkens, and M. Abbod, "Adaptive fuzzy TSK model-based predictive control using a CARIMA model structure," *Chemical Engineering Research and Design,* vol. 78, pp. 590-596, 2000.

[23] M. Morari, "Model predictive control: past, present and future* 1," *Computers & Chemical Engineering,* vol. 23, pp. 667-682, 1999.

[24] M. Diehl, H. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," *Nonlinear Model Predictive Control,* pp. 391-417, 2009.

[25] P. Deuflhard, *Newton methods for nonlinear problems: affine invariance and adaptive algorithms* vol. 35: Springer Verlag, 2004.

[26] N. De Oliveira and L. T. Biegler, "An extension of Newton-type algorithms for nonlinear process control* 1," *Automatica,* vol. 31, pp. 281-286, 1995.

[27] C. Long, P. Polisetty, and E. Gatzke, "Nonlinear model predictive control using deterministic global optimization," *Journal of Process Control,* vol. 16, pp. 635-643, 2006.

[28] R. Ghaemi, J. Sun, and I. V. Kolmanovsky, "An integrated perturbation analysis and sequential quadratic programming approach for model predictive control," *Automatica*, vol. 45, pp. 2412-2418, 2009.

[29] C. R. O'NEILL, "Improved System Identification for Aeroservoelastic Predictions," Citeseer, 2003.

[30] P. Young, *Recursive estimation and time-series analysis*: Springer Berlin, 1984.

[31] L. Ljung, *System Identification: Theory for the User*: Pearson Education, 1998.

[32] A. J. Miller, *Subset selection in regression*: Chapman & Hall/CRC, 2002.

[33] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. New York: Springer, 2001.

[34] G. E. P. Box and M. E. Muller, "A note on the generation of random normal deviates," *The Annals of Mathematical Statistics*, vol. 29, pp. 610-611, 1958.

# APPPENDIX

The methodologies of the nonlinear simulator, the GTSK MPC, and the Leapfrogging optimizer are described in Chapter 2. This section lists the Excel VBA code of the simulation system.

## A.1 Excel VBA code of the simulation system

```
Sub main()
'   The main subroutine calls each function or subroutine, keeping events organized
'   As you add features, do not code them in the Main, but place them in subroutines
'   or functions for the main sub to call

    If Cells(9, 13) = "N" Then Application.ScreenUpdating = False

    Call initialize                    'Input values and initialize states

    For SimTimeCounter = 0 To 650      'Simulation time counter
        Call events                    'Manage events and user-desired changes
        Call process                   'Process responds with a measurement
        SimTime = dt * SimTimeCounter   'Process simulation time interval is complete
        Call control                   'Controller responds with action
        Call evaluate                   'Determine goodness metrics
        Call output                    'Display results
    Next SimTimeCounter

 Application.ScreenUpdating = True

End Sub
''
Sub process()
'   This is the simulator for the real process.  In real life this is the physical process.
'   You would not be able to know any of these equations or variable values.
'   This is not in deviation variables.

    If u > 0 Then
       kp = u ^ (-0.5)     'Process gain is designed so that y_ss=10 when u=100%, and y-ss=0 when
u=0
    Else
       kp = 0
    End If
    If Environment = "ON" Then
       dist = lambdadist * sigmadistdriver * Sqr(-2 * Log(Rnd())) * Sin(twopi * Rnd()) + Clambdadist
* dist
    Else
       dist = 0
    End If
    j = j + 1                                       'Pointer for storage
    If j = 21 Then j = 1                            'Array max is 20th element
    y4 = (dt / taup4) * y3 + (1 - dt / taup4) * y4              'Process lags are calculated
    y3 = (dt / taup3) * (y2 + dist) + (1 - dt / taup3) * y3
    y2 = (dt / taup2) * y1 + (1 - dt / taup2) * y2              'In reverse order, so that ...
    y1 = (dt / taup1) * kp * u + (1 - dt / taup1) * y1             'each uses prior old value.
    yhold(j) = y4 + sigmameasurement * Sqr(-2 * Log(Rnd())) * Sin(twopi * Rnd())   'Add
NID(0,sigma) noise
    k = j - delay                                   'Pointer for delay read
    If k < 1 Then k = 20 + (j - delay)
    yp = yhold(k)
    yprocess(SimTimeCounter) = yp
    zAuxiliaryProcess = 0.075 * 0.9 * u + 0.925 * zAuxiliaryProcess 'Simple representation for the
auxiliary process

End Sub
'
'
Sub SOPDT_Model_P2N()
' SOPDT model  Past to Now
' Note: Model is in deviation variables, starts at zero with a zero slope, and
'   is influenced by the deviation in u from the initial u value
' This increments the model by one time step each control interval.

    If SimTimeCounter = 0 Then
       YmodelP2N = YmBase
    ElseIf SimTimeCounter < 21 And SimTimeCounter > 0 Then
       YmodelP2N = yP2N(SimTimeCounter - 1)                'when counter<36, yP2N=initial
    ElseIf SimTimeCounter > 20 Then
       yt1 = yP2N(SimTimeCounter - 1)                        'deliever values of regressors
       yt2 = yP2N(SimTimeCounter - 2)
       yt3 = yP2N(SimTimeCounter - 3)
       Ut = MV(SimTimeCounter - 20)

       numerator = 0
       denominator = 0

       For iLM = 1 To 12
         theta1 = matrix(iLM, 1)                             'read parameters
         theta2 = matrix(iLM, 2)
         theta3 = matrix(iLM, 3)
         theta4 = matrix(iLM, 4)
         theta5 = matrix(iLM, 5)
          'Truth
         center = matrix(iLM, 6)

         ylm = theta1 + theta2 * yt1 + theta3 * yt2 + theta4 * yt3 + theta5 * Ut 'local model value

         Ta = Exp(-(Ut - center) ^ 2 / 30)       'truth value

         numerator = numerator + ylm * Ta
         denominator = denominator + Ta
       Next iLM
         If denominator > 0 Then
           YmodelP2N = numerator / denominator
         Else
           YmodelP2N = yP2N(SimTimeCounter - 1)
         End If
    End If
    yP2N(SimTimeCounter) = YmodelP2N
    zAuxiliaryP2N = 0.1 * 0.8 * u + 0.9 * zAuxiliaryP2N
End Sub
'
'
Sub SOPDT_Model_N2F_SSD()
' SOPDT model  Now to Future.

'     Initialize model and variables
    Dim iN2F As Integer     'counter for future time increments
    Dim yN2F(200) As Single
```

```
   Dim MVN2F(200) As Single

   NOFE = NOFE + 1                  'Count number of function evaluations - a measure of work
required by the optimizer
   reference = yP2N(SimTimeCounter)        'Convert reference trajectory to deviation variables
   SSD = 0
   zAuxiliaryN2F = zAuxiliaryP2N

   For iN2F = 1 To NControlHorizon
      If SimTimeCounter + iN2F = 1 Then
         ymodelN2F = yP2N(SimTimeCounter)
      ElseIf SimTimeCounter + iN2F < 21 And SimTimeCounter + iN2F > 1 Then
         ymodelN2F = yN2F(iN2F - 1)                         'when counter<36, yP2N=initial
      ElseIf SimTimeCounter + iN2F > 20 Then

         'special
         MV(SimTimeCounter) = u1

         ' define three MVs
         If iN2F < 25 Then
            MVN2F(iN2F) = u1
         ElseIf iN2F < 50 And iN2F > 24 Then
            MVN2F(iN2F) = u2
         ElseIf iN2F < (NControlHorizon + 1) And iN2F > 49 Then
            MVN2F(iN2F) = u3
         End If

         'deliever values of regressors
         'yt1
         If iN2F < 2 Then                             'yt1
            yt1 = yP2N(SimTimeCounter + iN2F - 1)
         Else
            yt1 = yN2F(iN2F - 1)
         End If
         'yt2
         If iN2F < 3 Then                             'yt2
            yt2 = yP2N(SimTimeCounter + iN2F - 2)
         Else
            yt2 = yN2F(iN2F - 2)
         End If
         'yt3
         If iN2F < 4 Then                             'yt3
```

```
            yt3 = yP2N(SimTimeCounter + iN2F - 3)
          Else
            yt3 = yN2F(iN2F - 3)
          End If
          'u35
          If iN2F < 21 Then                                  'u35
            Ut = MV(SimTimeCounter + iN2F - 20)
          Else
            Ut = MVN2F(iN2F - 20)
          End If
          numerator = 0
          denominator = 0

          For iLM = 1 To 12
            theta1 = matrix(iLM, 1)                          'read parameters
            theta2 = matrix(iLM, 2)
            theta3 = matrix(iLM, 3)
            theta4 = matrix(iLM, 4)
            theta5 = matrix(iLM, 5)
            'Truth
            center = matrix(iLM, 6)
'             left = matrix(iLM, 7)
'             right = matrix(iLM, 8)

            ylm = theta1 + theta2 * yt1 + theta3 * yt2 + theta4 * yt3 + theta5 * Ut 'local model
value
            Ta = Exp(-(Ut - center) ^ 2 / 30)                              'truth value

            numerator = numerator + ylm * Ta
            denominator = denominator + Ta
          Next iLM

          If denominator > 0 Then
            ymodelN2F = numerator / denominator
          Else
            ymodelN2F = yN2F(iN2F - 1)
          End If
        End If

        yN2F(iN2F) = ymodelN2F

        reference = (dt / tauw) * yspBiasDeviation + (1 - dt / tauw) * reference   'deviation variable
```

```
      yProcessEstimate = ymodelN2F + pmmyf                    'Corrects the model value with pmm
to estimate future CV
      zAuxiliaryN2F = 0.1 * 0.8 * Ut + 0.9 * zAuxiliaryN2F
      zProcessEstimate = zAuxiliaryN2F + pmmz                    'Corrects the model value with
pmm to estimate future AuxV

      SSD = SSD + ((reference - ymodelN2F) ^ 2) / (ECVSP ^ 2) +
y_Constraint_Penalty(yProcessEstimate) / (ECVy ^ 2) + z_Constraint_Penalty(zProcessEstimate) /
(ECVz ^ 2)

      If display = "Yes" Then
        Cells(iN2F + SimTimeCounter + 15, 2) = SimTime + dt * iN2F
        Cells(iN2F + SimTimeCounter + 15, 6) = reference ' + YmInitial
        Cells(iN2F + SimTimeCounter + 15, 4) = MVN2F(iN2F) ' + uinitial
        Cells(iN2F + SimTimeCounter + 15, 5) = yN2F(iN2F) ' + YmInitial
        Cells(iN2F + SimTimeCounter + 15, 7) = yspBiasDeviation ' + YmInitial

      End If
   Next iN2F
   If display = "Yes" Then
      Cells(SimTimeCounter + 14, 7) = ""
      Cells(SimTimeCounter + 14, 4) = ""
      Cells(SimTimeCounter + 14, 5) = ""
      Cells(SimTimeCounter + 14, 6) = ""

      Calculate               'VBA command to update the active worksheet (updates graph)
   End If

End Sub
'
'
Function y_Constraint_Penalty(yProcessEstimate)
'  This function assesses a soft penalty if constraints on y are exceeded.
'  Full y range is 0 to 10.
'  In contrast to pushing model to biased setpoint, this compares biased model to limits.
'     yProcessEstimate is the pmm-biased model N2F prediction value.

   y_Constraint_Penalty = 0
   yConstraintMin = 0
   yConstraintMax = 10
   If yProcessEstimate < yConstraintMin Then y_Constraint_Penalty = (yProcessEstimate -
yConstraintMin) ^ 2
```

```
   If yProcessEstimate > yConstraintMax Then y_Constraint_Penalty = (yProcessEstimate -
yConstraintMax) ^ 2

End Function
'
'
Function z_Constraint_Penalty(zProcessEstimate)
'   This function assesses a soft penalty if constraints on Auxiliary Variable z are exceeded.
'   Full x range is 0 to 100.
'   In contrast to pushing model to biased setpoint, this compares biased model of z to limits.
'      zProcessEstimate is the pmm-biased z-model N2F prediction value.

   z_Constraint_Penalty = 0
   zConstraintMin = 0
   zConstraintMax = 100
   If zProcessEstimate < zConstraintMin Then z_Constraint_Penalty = (zProcessEstimate -
zConstraintMin) ^ 2
   If zProcessEstimate > zConstraintMax Then z_Constraint_Penalty = (zProcessEstimate -
zConstraintMax) ^ 2

End Function
'
'
Sub control()
'   This is the controller.

   If MODE = "MAN" Then          'In MANual mode
      Call SOPDT_Model_P2N          'Call P2N model to update its states and delay array history
of states
      pmmy = yprocess(SimTimeCounter) - yP2N(SimTimeCounter)              'Calculate pmmy
      pmmyf = pmmylambda * pmmy + (1 - pmmylambda) * pmmyf        'filter pmmy
      pmmz = zAuxiliaryProcess - zAuxiliaryP2N
      yspbias = ysp - pmmyf             'Calculate a biased setpoint for display
      u = u                         'User decides this value
      If u > 100 Then u = 100           'But the human often needs to be regulated
      If u < 0 Then u = 0
      u1 = u
      u2 = u
      u3 = u
   Else                      'In AUTOmatic mode
      Call SOPDT_Model_P2N          'Call P2N model to get response to past u
      pmmy = yprocess(SimTimeCounter) - yP2N(SimTimeCounter)              'Calculate pmmy
      pmmyf = pmmylambda * pmmy + (1 - pmmylambda) * pmmyf        'filter pmmy
      pmmz = zAuxiliaryProcess - zAuxiliaryP2N
      yspbias = ysp - pmmyf              'Bias setpoint for model
      Call Leapfrogging_Optimizer        'Calculate u1, u2, and u3
      u = u1                             'Implement u1
   End If

'   Since optimizer will return a U value within constraints, the 0 to 100% override is not needed
here.

   If u > 100 Then u = 100         'override calculated extremes with what is implementable
   If u < 0 Then u = 0

   MV(SimTimeCounter) = u

End Sub
'
'
Sub initialize()
'   Initializes variables, reads the input data

   Randomize                         'Randomize the random number generator for noise
   twopi = 2 * 3.14159265358979       'Constant in Box-Muller NID noise generation

   dt = 0.2                  'Time interval (all time constants should be about 10 times larger)
   SimTime = 0

   taup1 = Cells(1, 6)        'Input process values (the controller or operator cannot know these)
   If taup1 < 1 Then taup1 = 1
   taup2 = Cells(2, 6)
   If taup2 < 1 Then taup2 = 1
   taup3 = Cells(3, 6)
   If taup3 < 1 Then taup3 = 1
   taup4 = Cells(4, 6)
   If taup4 < 1 Then taup4 = 1

   delay = Cells(5, 6)

   If delay > 20 Then delay = 20
   sigmameasurement = Cells(6, 6)

   taudist = Cells(1, 14)      'Input disturbance characteristics
   sigmadist = Cells(2, 14)
   dist = 0
   Environment = "OFF"
   lambdadist = dt / taudist
   Clambdadist = 1 - lambdadist
   sigmadistdriver = Sqr(2 / lambdadist - 1) * sigmadist

   If Cells(1, 2) <> "GS" Then km = Cells(1, 2)          'Input operator-chosen controller (includes
model) variable values
   taum1 = Cells(2, 2)
   If taum1 < 1 Then taum1 = 1
   taum2 = Cells(2, 3)
   If taum2 < 1 Then taum2 = 1
   thetam = Cells(3, 2)
   uBase = Cells(4, 2)        'Since linear SOPDT model is in deviation variables, need initial u
   YmBase = Cells(5, 2)       'Since linear SOPDT model is in deviation variables, need y-SS that
goes with initial u
   tauw = Cells(6, 2)
   If tauw < 1 Then tauw = dt
   pmmylambda = Cells(7, 2)

   u = 50                    'Initialize process and controller at an initial steady state
   ysp = 7
   yinitial = 7
   y1 = yinitial
   y2 = yinitial
   y3 = yinitial
   y4 = yinitial
   yp = yinitial
   For j = 1 To 20
      yhold(j) = yinitial
   Next j
   pmmyf = 0
   j = 0
   dist = 0
   zAuxiliaryProcess = 0.9 * u
   zAuxiliaryP2N = 0.8 * u
   zAuxiliaryN2F = zAuxiliaryP2N

   MODE = "MAN"          'Initialize controller in manual

   For i1 = 1 To 12
```

```
        For j1 = 1 To 6
            matrix(i1, j1) = Cells(i1 + 1, j1 + 26)
            'Cells(i + 20, j + 19) = matrix(i, j)
        Next j1
    Next i1

    AUTOcount = 0          'Initialize evaluation variables
    ISE = 0
    Travel = 0
    uold = u

    ECVSP = 1          'Equal Concern Value for CV not being at the SP  (CV goes from 0 to 10)
    ECVy = 0.1         'Equal Concern Value for CV violating a constraint
    ECVz = 10          'Equal Concern Value for Aux Variable violating a constraint (AV goes from 0 to
100)
    ROCu = 75          'Rate of Change constraint on u

    NControlHorizon = 100           'Initialize model N2F variables

    ConvergenceThreshold = Cells(4, 17)      'Read RMS distance convergence criterion for
optimizer

    Call Clear_Old_Data            'Remove all past data from plot points

    If Cells(8, 18) = "Y" Then
        Cells(1, 18) = "Lo/Hi PN)"
        Cells(1, 19) = "Player Num"
        Cells(1, 20) = "OF Value"
        Cells(1, 21) = "u1"
        Cells(1, 22) = "u2"
        Cells(1, 23) = "u3"
    End If

End Sub
'
'
Sub output()
'   Places all data on the worksheet for display

    Cells(SimTimeCounter + 10, 2) = SimTime
    Cells(SimTimeCounter + 10, 3) = yp
    Cells(SimTimeCounter + 10, 4) = u
    Cells(SimTimeCounter + 10, 5) = YmodelP2N
    Cells(SimTimeCounter + 10, 6) = ysp
    Cells(SimTimeCounter + 10, 7) = yspbias
    Cells(SimTimeCounter + 10, 9) = zAuxiliaryP2N

    Cells(SimTimeCounter + 14, 2) = ""
    Cells(SimTimeCounter + 14, 8) = ""
    Cells(SimTimeCounter + 15, 8) = 0
    Cells(SimTimeCounter + 16, 8) = 10

    Cells(1, 10) = NISE
    Cells(2, 10) = NTravel
    Cells(1, 17) = MODE
    Cells(2, 17) = Environment
    Cells(5, 17) = Iteration
    Cells(6, 17) = NOFE

End Sub
'
'
Sub evaluate()
'   Calculate performance measures for controlled process

    If MODE = "AUTO" Then
        AUTOcount = AUTOcount + 1       'Count of samplings in AUTO when ISE and Travel are
calculated
        ISE = ISE + (ysp - yp) ^ 2      'Not really ISE because no dt multiplier.  Really SSE
        NISE = ISE / AUTOcount          'Since no dt in ISE, divide ny count, not time.
        Travel = Travel + Abs(u - uold)
        uold = u
        NTravel = Travel / AUTOcount
    End If

End Sub
'
'
Sub events()
'   Trigger events for the system (process and controller) simulation

    If SimTimeCounter = 25 Then
        MODE = "AUTO"
        ysp = 7
```

```
   End If

   If SimTimeCounter = 50 Then ysp = 3
   If SimTimeCounter = 35 Then Environment = "ON"
   If SimTimeCounter = 350 Then ysp = 9

End Sub
'
'
Sub Clear_Old_Data()

   Range("B10:K2010").Select
   Selection.ClearContents

   Range("R1:R5").Select
   Selection.ClearContents

   Range("S1:W43").Select
   With Selection.Interior
      .Pattern = xlNone
      .TintAndShade = 0
      .PatternTintAndShade = 0
   End With
   Selection.ClearContents

   Range("A1").Select

End Sub
'
'  ************************************
Sub Initialize_Leapfrogging_Optimizer()

   DVDimension = 3
   NumTeammates = 20
   MovementL2H = 1              'size of window on other side of low from high

   PlayerNumber = 1
   PlayerPosition(1, PlayerNumber) = u1    'set player #1 as the previous best solution, which may
still be the best
   PlayerPosition(2, PlayerNumber) = u2
   PlayerPosition(3, PlayerNumber) = u3
   Call Assign
```

```
    Call SOPDT_Model_N2F_SSD    'determine the SSD result
    PlayerOFValue(PlayerNumber) = SSD

    For PlayerNumber = 2 To NumTeammates    'initialize player values - must be within feasible
region
        Constraint = "Unassessed"
        Do Until Constraint = "PASS"
            u1 = u + 2 * (Rnd() - 0.5) * ROCu                  'randomize u1 for the player
            PlayerPosition(1, PlayerNumber) = u1
            u2 = u1 + 2 * (Rnd() - 0.5) * ROCu                 'randomize u2 for the player
            PlayerPosition(2, PlayerNumber) = u2
            u3 = u2 + 2 * (Rnd() - 0.5) * ROCu                 'randomize u3 for the player
            PlayerPosition(3, PlayerNumber) = u3
            Call Assign
            Call u_ConstraintTest
        Loop
        Call SOPDT_Model_N2F_SSD    'determine the SSD result
        PlayerOFValue(PlayerNumber) = SSD
    Next PlayerNumber

    Call Find_High
    Call Find_Low

    If Cells(8, 18) = "Y" Then Call Show_Players

End Sub
'
'   *************************************
Sub Find_High()
'   Search for player with highest OF value

    LFHighpn = 1 + Int(NumTeammates * Rnd())    'Random Assignment for initialization in case
floor is flat
    OFhigh = PlayerOFValue(LFHighpn)
    For PlayerNumber = 1 To NumTeammates              'search through all players
        If PlayerOFValue(PlayerNumber) > OFhigh Then            'Reassign if worst
            LFHighpn = PlayerNumber
            OFhigh = PlayerOFValue(PlayerNumber)
        End If
    Next PlayerNumber
    For DVNumber = 1 To DVDimension
        HighPlayerPosition(DVNumber) = PlayerPosition(DVNumber, LFHighpn)
    Next DVNumber

End Sub
'
'   **************************************
Sub Find_Low()
'   Search for player with lowest OF value

    LFLowpn = 1                         'start with PlayerNumber=1, if floor is flat, this serves as the
base for convergence
    OFlow = PlayerOFValue(LFLowpn)
    For PlayerNumber = 2 To NumTeammates
        If PlayerOFValue(PlayerNumber) < OFlow Then           'Reassign if better
            LFLowpn = PlayerNumber
            OFlow = PlayerOFValue(LFLowpn)
        End If
    Next PlayerNumber
    For DVNumber = 1 To DVDimension
        LowPlayerPosition(DVNumber) = PlayerPosition(DVNumber, LFLowpn)
    Next DVNumber

End Sub
'
'   ***********************************
Sub Leapfrogging_Optimizer()
'   Relocate the player with the worst position to a random position to the other side of the best.
'   If desired reevaluate the best to avoid finding a fortuitous best ever in stochastic functions

    NOFE = 0

    yspBiasDeviation = yspbias ' - YmBase

    Call Initialize_Leapfrogging_Optimizer

    For Iteration = 1 To 500
                        'relocate worst with the leapover the best
    Constraint = "Unassessed"      'but must jump to an unconstrained area
    PlayerNumber = LFHighpn
    Do Until Constraint = "PASS"
        For DVNumber = 1 To DVDimension
```

```
         PlayerLeapDelta(DVNumber) = LowPlayerPosition(DVNumber) -
HighPlayerPosition(DVNumber)       'difference between trial solutions with highest and lowest
OF values
         HighPlayerPosition(DVNumber) = LowPlayerPosition(DVNumber) + MovementL2H * Rnd()
* PlayerLeapDelta(DVNumber)        'high (or infeasible) jumps to random position in window,
repelled by recent vacated spots
      Next DVNumber
      For DVNumber = 1 To DVDimension
         PlayerPosition(DVNumber, LFHighpn) = HighPlayerPosition(DVNumber)       'reassign
position of former high individual to its new feasible location
      Next DVNumber
      Call Assign
      Call u_ConstraintTest
   Loop

   Call Assign
   SOPDT_Model_N2F_SSD
   PlayerOFValue(LFHighpn) = SSD
                           'find the individual with the lowest OF value presently
   If PlayerOFValue(LFHighpn) < OFlow Then  'If needed, reassign player with lowest OF value
      OFlow = PlayerOFValue(LFHighpn)
      For DVNumber = 1 To DVDimension
         LowPlayerPosition(DVNumber) = PlayerPosition(DVNumber, LFHighpn)
      Next DVNumber
      LFLowpn = LFHighpn
   End If
                           'find the individual with the highest OF value presently
   If PlayerOFValue(LFHighpn) > OFhigh Then     'we know which is high
      OFhigh = PlayerOFValue(LFHighpn)
      For DVNumber = 1 To DVDimension
         HighPlayerPosition(DVNumber) = PlayerPosition(DVNumber, LFHighpn)
      Next DVNumber
      If Cells(9, 18) = "Y" Then
         PlayerNumber = LFLowpn
         Call Assign
         display = "Yes"              'Tell SOPDT N2F model to display the optimum results
         Call SOPDT_Model_N2F_SSD
         display = "No"
      End If
   Else
      Call Find_High                'need to search for the new high
   End If
```

```
    If Cells(8, 18) = "Y" Then Call Show_Players

    du1 = HighPlayerPosition(1) - LowPlayerPosition(1)          'Compute Convergence Metric
    du2 = HighPlayerPosition(2) - LowPlayerPosition(2)
    du3 = HighPlayerPosition(3) - LowPlayerPosition(3)
    If Sqr((du1 ^ 2 + du2 ^ 2 + du3 ^ 2) / 3) < ConvergenceThreshold Then Exit For

    Next Iteration

    If Cells(8, 18) = "Y" Then Call Show_Players

        PlayerNumber = LFLowpn
        Call Assign
        display = "Yes"                 'Tell SOPDT N2F model to display the optimum results
        Call SOPDT_Model_N2F_SSD
        display = "No"

End Sub
'
'--------------------------------------------------------
Sub Assign()

    u1 = PlayerPosition(1, PlayerNumber)
    u2 = PlayerPosition(2, PlayerNumber)
    u3 = PlayerPosition(3, PlayerNumber)

End Sub
'
'--------------------------------------------------------
Sub u_ConstraintTest()

    Constraint = "PASS"

    If u1 > 99 Then Constraint = "FAIL"
    If u2 > 99 Then Constraint = "FAIL"
    If u3 > 99 Then Constraint = "FAIL"
    If u1 < 1 Then Constraint = "FAIL"
    If u2 < 1 Then Constraint = "FAIL"
    If u3 < 1 Then Constraint = "FAIL"
    If Abs(u1 - u) > ROCu Then Constraint = "FAIL"
    If Abs(u2 - u1) > ROCu Then Constraint = "FAIL"
    If Abs(u3 - u2) > ROCu Then Constraint = "FAIL"

'    If Constraint = "FAIL" Then
'        Cells(5, 18).Interior.ColorIndex = 3
'    Else
'        Cells(5, 18).Interior.ColorIndex = 4
'    End If

End Sub
'
'   ************************
Sub Show_Players()

    For PlayerNumber = 1 To NumTeammates
        Cells(2 + PlayerNumber, 19) = PlayerNumber
        Cells(2 + PlayerNumber, 20) = PlayerOFValue(PlayerNumber)
        For DVNumber = 1 To DVDimension
            Cells(2 + PlayerNumber, 20 + DVNumber) = PlayerPosition(DVNumber, PlayerNumber)
        Next DVNumber
    Next PlayerNumber

    If LFHighpnold > 0 Then Cells(2 + LFHighpnold, 19).Interior.ColorIndex = 0
    Cells(2 + LFHighpn, 19).Interior.ColorIndex = 3
    LFHighpnold = LFHighpn
    If LFLowpnold > 0 Then Cells(2 + LFLowpnold, 19).Interior.ColorIndex = 0
    Cells(2 + LFLowpn, 19).Interior.ColorIndex = 4
    LFLowpnold = LFLowpn

    Cells(3, 18) = LFLowpn
    Cells(4, 18) = LFHighpn

End Sub
```

VITA

Haoxian Chen

Candidate for the Degree of

Master of Science

Thesis: INCORPORATION OF THE GENERALIZED TSK MODELS IN MODEL PREDICTIVE CONTROL

Major Field: Chemical Engineering

Biographical:

Education:

- Completed the requirements for the Master of Science in Chemical Engineering at Oklahoma State University (OSU), Stillwater, Oklahoma in December, 2011.
- Completed the requirements for the Bachelor of Engineering in Automation at East China University of Science and Technology, Shanghai, China in 2009.

Experience:

- Research Assistant for Dr. R. Russell Rhinehart, OSU, August 2009-December 2011.
- Teaching Assistant to Dr. Lionel M. Raff, OSU, August 2010 - December 2010.
- Teaching Assistant to Dr. Jan Wagner, OSU, January 2011 - May 2011.

Professional Memberships:

- Omega Chi Epsilon, OSU Chapter.
- International Society of Automation (ISA), OSU Chapter.
- American Institute of Chemical Engineers (AIChE), OSU Chapter

Name: Haoxian Chen Date of Degree: December, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: INCORPORATION OF THE GENERALIZED TSK MODELS IN MODEL PREDICTIVE CONTROL

Pages in Study: 74 Candidate for the Degree of Master of Science

Major Field: Chemical Engineering

Scope and Method of Study:

The generalized TSK (GTSK) modeling approach is proved to provide accurate model prediction and to alleviate the computational burden. The scope of this study is to incorporate the GTSK models in the nonlinear model predictive control (NMPC) to improve the overall performance and reliability of NMPC. A novel global optimization method, the Leapfrogging technique, is also used to further improve the NMPC's computational efficiency. Another innovation, the "sawtooth" pattern is used as input signal to generate the GTSK model. The experiments and tests are conducted on a nonlinear process simulation system, in which the NMPC control algorithm was embedded. The virtual process in this simulator is fourth-order-plus-dead-time (FOPDT) process with a nonlinear gain and the environmental effect (noise and disturbance). The controlled process is subject to both soft and hard constraints – soft on both the controlled and the auxiliary variable, and hard on both the limits and rate of change of the manipulated variable. The NMPC performance is evaluated via several simulation experiments, which involved constraint handling, interactions and process nonlinearity.

Findings and Conclusions:

The use of a GTSK model and Leapfrogging as an optimizer were demonstrated as effective for nonlinear model predictive control. The nonlinear model is firstly developed by using GTSK approach. The prediction accuracy of the GTSK model was illustrated and quantified by a comparison with SOPDT model. The GTSK model was much better. The performance of GTSK MPC controller is evaluated via seven sets of dynamic control simulation. The controller showed desirable performance for disturbance rejection, set point tracking, constraint handling, and comprehensive environmental effect handling.

ADVISER'S APPROVAL: Dr. R. Russell Rhinehart