HVAC CONTROL USING SUPPORT VECTOR REGRESSION MODELS

XI XUECHENG

NATIONAL UNIVERSITY OF SINGAPORE

2003

HVAC CONTROL USING SUPPORT VECTOR REGRESSION MODELS

XI XUECHENG

(B.Eng., M.Eng. NUAA)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgement

First and foremost, the author would like to express his sincere gratitude to his supervisors, Professor Poo Aun Neow and Associate Professor Chou Siaw Kiang, for their patient guidance, inspirations and valuable suggestions throughout this project.

The author would also like to thank Mr. Sacadevan Raghavan and Mrs. Hung-Ang Yan Leng, the lab officers of the Air-Conditioning Laboratory in the Mechanical Engineering Department for their invaluable assistance in the experiments. The author is grateful to Mr. Zheng Qiaoqing and Dr. Duan Kaibo and many other friends for their invaluable advices and help during the project.

Finally, the author expresses his heartfelt thanks to his parents, wife and sisters for their love and support.

Table of Contents

Summary

This project focuses on the simultaneous control of temperature and relative humidity of a conditioned space, which is required by some industrial and scientific applications.
HVAC plants are typical nonlinear systems and obtaining accurate models for these systems is a difficult and challenging task. In this project, a new tool—support vector regression (SVR) — is used to model the inverse and forward dynamics of this highly nonlinear system.

Support vector regression is a type of model that is optimized so that prediction error and model complexity are simultaneously minimized. Because of its universal approximation ability, support vector regression can be used to model nonlinear processes, just as neural networks are.

Both the SVR inverse control and SVR model predictive control consist of two stages: The first is the system identification for the HVAC system. For inverse control, SVR inverse models are needed while for model predictive control, SVR forward models are needed. Choosing optimal hyper-parameters for the models is an important step in the identification stage. *k*-fold cross validation is a reliable way to determine the optimal hyper-parameters. The optimal values are firstly searched in coarse grids, and then searched in finer grids. The final models are obtained after training the SVRs using these optimal hyper-parameters. The models obtained this way are found to have good generalization property.

In inverse control, the inverse model is simply cascaded with the controlled system in order that the whole system results in identity mapping between the desired response and the controlled system output. Thus, SVR models act directly as the controllers in such a configuration. It is important to design an appropriate reference for the system to follow. The controller has the ability of set point tracking and disturbance rejection. The controller can work effectively in the start up period which is difficult to be described by a linear model around certain operating points. However, the disadvantage of the SVR inverse controller is that the response time is quite slow.

The basic idea of Model Predictive Control (MPC) is to predict the controlled variables over a future horizon using a prediction model of the process, the control signals are then computed by minimizing an objective function, and only the first control action is finally applied to the process. The procedure is repeated at every sampling instant using the updated information (measurements) of the process. A key advantage of MPC over other control schemes is its ability to deal with constraints in a systematic and straightforward manner. The online MPC problem is solved by iterative dynamic programming (IDP). The items in the performance index are found to have significant impact on the controller performance. The MPC strategy has been proved to be successful experimentally. Experimental results show that both the room temperature and the room relative humidity are accurately controlled to their desired values respectively within the system operating range. The control performances are quite satisfactory in terms of reference tracking ability, steady-state error, amplitude of overshooting and consideration of control constraints.

Nomenclature

b	Constant offset (or threshold)
C	Regularization parameter
g	Composite hyperparameter, $g = 1/(2\sigma^2)$
k	Sampling instant
k_{ij}	Dot product of two feature vectors, $k_{ij} = K(x_i, x_j)$
K	Feature map
M	Number of randomly chosen control candidates
N	Number of y-grid points
N_i	Number of iterations in each pass
r_1	Reference value of room temperature
r_2	Reference value of room relative humidity
r^i	Allowable control region
RRH	Room relative humidity
RT	Room temperature
s^*	Constant sum, $s^* = \lambda_u + \lambda_v = \lambda_u^* + \lambda_v^*$
SRH	Supply air relative temperature
ST	Supply air temperature
u	Input vector
u_1	Supply air fan speed
u_2	Chilled water valve opening
v	Input vector to dynamic model
v_{ij}	Support vector in dynamic model
w	Weigh vector in feature space
x_1	Room temperature
x_2	Room relative humidity
x_i	Vector with feature elements
y	Output vector
y_1	Room temperature

y_2	Room relative humidity
$\bar{y}_1$	Initial value of room temperature at the start of the sampling period
$\bar{y}_2$	Initial value of room relative humidity at the start of the sampling period
y_i	Target value or system output
α_i	Lagrange multiplier or expansion coefficient
γ_i	Penalty coefficient on error between current value and reference value
ε	Parameter of the ε-insensitive loss function
φ_1	Contraction factor after each iteration
φ_2	Restoration factor after each pass
φ_i	Penalty coefficient on change rate of control signal
η	Constant, $\eta = k_{vv} + k_{uu} - 2k_{uv}$
η_i	Terminal cost coefficient or Lagrange multiplier
λ_i	Composite Lagrange multiplier, $\lambda_i = \alpha_i - \alpha_i^*$
θ_i	Penalty coefficient on magnitude of control signal
σ	Width parameter in Gaussian kernel function
ξ_i	Slack variable

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

Temperature and relative humidity (RH) are among the most important thermodynamic parameters in commercial and industrial air-conditioning and in process control. Some industrial and scientific processes require simultaneous and accurate control of temperature and relative humidity. Temperature and relative humidity can influence the rate of chemical and biochemical reactions, the rate of crystallization, the density of chemical solutions, the corrosion of metals and the generation of static electricity and the manufacturing of printed circuit boards (PCB) in clean rooms(ASHRAE Handbook, 1999). In textile and paper processing, the high-speed machinery used requires accurate control of both temperature and relative humidity for proper functioning (Krakow et al., 1995). Some scientific experiments can be performed properly only under some specific controlled environments.

Space temperature is dependent on the sensible heat load and system sensible cooling capacity, while the relative humidity is dependent on the latent heat load and system latent cooling capacity. In order to accomplish the simultaneous control of temperature and relative humidity, HVAC (Heating, Ventilation and Air-Conditioning) systems must be designed to cater to both sensible and latent components of the heat load. The conventional way of the accurate control of temperature and relative humidity of a conditioned space is to cool the air to the required specific humidity and reheat it to the desirable temperature. Obviously, this method is not energy-efficient. According to heat transfer theories, chilled water flow rate and supply airflow rate decide the system cooling capacity jointly. In this project, the simultaneous control of room temperature and relative humidity is carried out by the means of varying both the supply airflow rate and the chilled water flow rate.

The behavior of the controller has direct impact on the performance of a HVAC system. The objective of a control system is to adjust the plant cooling capacity to adapt to the varying thermal load. Because of their simplicity, PID (Proportional-Integral-Derivative) controllers are widely used in industry and have been proven to be valuable and reliable in HVAC applications (Rosandich, 1997). Consisting of many mechanical, hydraulic and electrical components, HVAC plants are typical nonlinear systems. Conventional PID controllers work well for linear plants. When used on a nonlinear plant, the PID controller can perform well around a small region of an operating point. The PID controller does not adapt well to changes in the plant characteristics brought about by shifting of operating points. If a PID controller is intended to work in a wide operating range of a nonlinear plant, it must be tuned very conservatively to provide stable behavior. When a well-tuned PID controller is applied to another system with different model parameters, or when the system parameters change during operation, its performance degrades (Kasahara *et al.,* 1999). In view of the shortcomings of linear controllers, it is necessary to adopt a nonlinear controller for better performance.

With theoretical developments in model-based control strategies and availability of cheap and fast computers, the design and analysis of nonlinear control systems have been received considerable attention from both academia and industry in the past decades. Fuzzy logic (Arima 1995) and neural networks (Khalid *et al.* 1995) have been successfully used to design nonlinear controllers for HVAC plants. In this project, a new method of support vector regression (SVR) will be used to build inverse and forward dynamic models. Two nonlinear controllers, an inverse controller and a nonlinear model predictive controller, are designed based on the inverse and forward dynamic models.

A support vector machine (SVM) is a type of model that is optimized so that prediction error and model complexity are simultaneously minimized (Vapnik, 1995). Support vector machines have been developing very fast in recent years. SVMs not only have a more solid foundation than artificial neural networks, but are able to serve as a replacement for neural networks that perform as well or better, in a wide variety of fields (Scholköpf and Smola, 2002). SVMs work by mapping the input space into a high-dimensional feature space using kernel tricks. Like conventional neural networks (Haykin 1999), SVMs have been used by researchers to solve classification and regression problems. One advantage of the SVM over neural networks is that the SVM formulates classification and regression as a quadratic optimization problem which ensures that there is only one global minimum. The training of neural networks may get trapped at a local minimum. Another advantage is that training of the SVM is generally faster than that for the neural networks. This is a desirable property for online applications.

Because of its universal approximation ability, support vector regression can be used to model nonlinear process, just as neural networks are. Support vector regression has been reported to be used in control area.

In this project, SVR is used to model the inverse and forward dynamics of a HVAC system. Based on these models, an inverse controller and a model predictive controller are designed, respectively. Direct inverse control utilizes an inverse system model. The inverse model is simply cascaded with the controlled system in order that the composed system results in identity mapping between desired response and the controlled system output. Thus, the inverse SVR model acts directly as the controller in such a configuration. Model predictive control (MPC) or receding horizon control (RHC) is a form of control in which the current control action is obtained by solving on-line, at each sampling instant, a finite horizon open-loop optimal control problem, using the current state of the plant as the initial state. The optimization yields an optimal sequence and the first control in this sequence is applied to the plant (Mayne *et al*., 2000). A key advantage of MPC over other control schemes is its ability to deal with constraints in a systematic and straightforward manner. Within the framework of constrained optimization, the constraints that can be handled include not only saturation limits but also sorts of performance and safety constraints on inputs, outputs and state variables of the process.

1.2 Objectives and Scope

This project studies the simultaneous control of the temperature and relative humidity in a conditioned space using support vector regression. The strategy of controlling space temperature and relative humidity by varying the supply airflow rate and the chilled water flow rate was investigated. The system energy efficiency was improved because the fan need not always run at its maximum speed and reheat of the air is not needed.

Support vector regression was utilized to model the inverse and forward dynamics of the HVAC system. The optimal tuning parameters are chosen by *k*-fold cross validation. Based on the inverse dynamic model, a relatively simple SVR inverse controller was designed. Experimental results show that the SVR inverse controller has the ability of simultaneous control of both room temperature and relative humidity. However, the response time of the SVR inverse controller is slow. In view of the shortcomings of the inverse controller, a more sophisticated control strategy of model predictive control, which utilizes the forward dynamic model of HVAC system, was investigated. Experimental results demonstrate that the SVR model predictive controller not only has good reference tracking ability, small steady errors and shorter response time, it also has the ability of considering control limits.

The HVAC system for this project is a chilled water system for a thermal chamber in the Department of Mechanical Engineering, National University of Singapore. Two three-way valves were used to modify the chilled water flow rate and one variable speed fan was installed in the air-handling unit. System data such as the supply air temperature and relative humidity, room temperature and relative humidity, three-way valve position, and supply air fan speed were recorded in the experiments. The designed control system was implemented and applied to the thermal chamber to study its performance. The feasibility of the control strategy was confirmed by experimental results.

1.3 Outline of Thesis

This thesis is divided into seven chapters. Chapter 1 gives the background, objectives and scope, and outline of the thesis. Chapter 2 reviews previous literature on temperature and relative humidity control of HVAC systems. An overview of support vector machines and the training algorithm--sequential minimal optimization-- are given in Chapter 3. The inverse and forward modeling of the HVAC system is done in Chapter 4. Chapter 5 discusses the design of the SVR inverse controller and the experimental results. Chapter 6 describes the design of the model predictive controller and the experimental results. The online optimization algorithm of iterative dynamic programming is also introduced. The parameters that affect the control performance are investigated. The conclusions are given in Chapter 7 as well as some recommendations for future developments.

Chapter 2 Literature Review

2.1 HVAC

Heating, ventilation and air-conditioning (HVAC) is widely used and affecting aspects of our lives. Simultaneous control of temperature and relative humidity will be discussed in this project. Before doing that, it will be helpful to introduce some aspects of HVAC relating to this project.

2.1.1 Variable Air Volume System vs. Constant Volume System

HVAC can be broadly classified into constant air volume (CAV) and variable air volume (VAV) systems. Our HVAC control will be performed on a variable air volume (VAV) system. The popularity of VAV systems has grown rapidly due to their ability to save large amounts of heating, cooling and fan energy when compared to other HVAC systems. The increasing applications of direct digital control (DDC) also intensify the uses of VAV technology. Variable speed drive (VSD) technology has helped the greater use and has especially shown the advantage of VAV systems.

In CAV systems, the volume flow rates of conditioned air are maintained constant to a conditioned space while the supply air temperature is continuously varied to match the thermal load (Shepherd, 1998). The chilled water flow rate is controlled to get the desired supply air temperature. In such systems, fans run at a fixed high speed, designed to meet the peak load. However, for most of the time, systems actually operate at part load. Thus, a large amount of energy is wasted since the energy consumption of the fan is approximately proportional to the cubic of its speed. It is also observed that the use of a constant volume air flow rate without regard to thermal loads leads to high space relative humidity in part-load conditions.

In the alternative approach of VAV systems, conditioned air is supplied at a constant temperature to a conditioned space while the volume flow rates are continuously varied to match the thermal load. Therefore, smaller amounts of conditioned air are used at part loads. A variable speed drive (VSD) for the fan can be used for this purpose. Energy savings are achieved and high relative humilities are not experienced at part-load conditions by using VAV systems.

2.1.2 HVAC Control

The performance of the controller has a direct effect on the performance of HVAC systems. The objective of a HVAC controller is to adjust the plant cooling capacity to adapt to the varying thermal load. PID controllers are by far the most widely used and have been proven to be valuable and reliable in HVAC applications (Rosandich, 1997). However, PID control works most favorably when the system model parameters do not change much. HVAC systems have complex dynamics with nonlinearity, distributed parameters, and multi variables and are subject to external disturbances, such as the weather variation and changing heat loads. The conventional PID controllers work well for linear plants. When used on a nonlinear plant, PID controllers can perform well only in a small region around a certain operating point. PID controllers cannot adapt to changes in the plant characteristics brought about by the shifting of the operating point. When a well-tuned PID controller is applied to another system with different model parameters, or when system parameters change during operation, its performance degrades (Kasahara *et al.*, 1999). So it is necessary to adopt a nonlinear controller. Fuzzy logic (Arima 1995) and neural networks (Khalid *et al.* 1995) have been successfully used in nonlinear controllers for HVAC plants.

In this study, a new method of support vector regression will be used to build inverse and forward dynamic models for a HVAC system. Two nonlinear controllers, i.e. inverse controller and a nonlinear model predictive controller, are designed based on the inverse and forward dynamic models, respectively.

2.2 Nonlinear Control

In the past decades, the control of nonlinear systems has received considerable attention in both academia and industry. The recent interest in the design and analysis of nonlinear control systems is due to several factors. Firstly, linear controllers usually perform poorly when applied to highly nonlinear systems. Secondly, significant progress has been made in the development of model-based controller design strategies for nonlinear systems. Finally, the developments of inexpensive and powerful computers have made on-line implementation of these nonlinear model-based controllers feasible (Henson and Seborg, 1997).

Many common process control problems exhibit nonlinear behavior, in that the relationship between the input and output variables depends on the operating conditions. For examples, if the dynamic behavior of a nonlinear process is approximated by a linear model with a transfer function, the model parameters (e.g. steady state gain, time constant, time delay) depend on the nominal operating condition. If the process is only mildly nonlinear or remains in the vicinity of a nominal steady state, then the effects of the nonlinearities may not be severe. In these situations, conventional feedback control strategies can provide adequate performance. However, many important industrial processes exhibit highly nonlinear behavior. The process may be required to operate over a wide range of conditions due to large process upsets or set-point changes. When conventional PID controllers are used to control such highly nonlinear processes, the controllers must be tuned very conservatively in order to provide stable behavior over the entire range of operating conditions. But conservative controller tuning can result in serious degradation of control system performance.

In view of the shortcomings of linear controllers for highly nonlinear processes, there are considerable incentives for developing more effective control strategies that incorporate knowledge of the nonlinear characteristics of the plant under control. During the past decade, there have been intensive research interests in developing nonlinear control strategies that are appropriate for process control. There are several kinds of nonlinear controller such as, but not limited to, fuzzy control, input/output linearization (i.e. feedback linearization control), and nonlinear model predictive control.

Fuzzy control is a kind of control approach that uses fuzzy set theory. Fuzzy sets were first proposed by Zadeh (1965). Fuzzy control offers a novel mechanism to implement such control laws that are often knowledge-based (rule-based) expressed in linguistic description (Cai, 1997). The drawback for fuzzy control is that it is often very difficult to build up appropriate rule sets for MIMO system, especially when the system has cross-strong couplings between inputs and outputs.

The input/output linearization controller design method provides exact linearization of nonlinear models. Unlike conventional linearization using Taylor series expansion about some operating point, this technique produces a linearized model that is independent of operating points. An analytical expression for the nonlinear control law can then be derived for broad classes of nonlinear systems. The approach is based on concepts from nonlinear system theory. The resulting controller includes the inverse of the dynamic model of the process, providing that such an inverse exists. The general approach has been utilized in several process control design methods such as: generic model control, globally linearizing control, reference system synthesis and a nonlinear version of internal model control.

Within the last decades, model-based control strategies such as model predictive control (MPC) have become the preferred control technique for difficult multivariable control problem (Camacho and Bordons, 1995). Morari and Lee (1999) gave a good overview on the past, present and future of MPC. It has been proven that MPC has desirable stability properties for nonlinear systems (Keerthi and Gilbert, 1988; Mayne and Michalsha, 1990).

Because the current generations of MPC systems are largely based on linear dynamic models such as step response and impulse response models, the resulting linear controllers must be conservatively tuned for highly nonlinear processes. The success of linear model predictive control systems has motivated the extension of this methodology to nonlinear control problems. The general approach is referred to as nonlinear model predictive control. The control problem formulation is analogous to linear model predictive control except that a nonlinear model is used to predict future process behavior. The required control actions are calculated by solving a nonlinear programming problem at each sampling instant.

2.3 Neural Networks in Nonlinear Control

Before we discuss support vector regression (SVR) in control, it is necessary to discuss the application of neural networks in control. A comprehensive survey paper on this topic was given by Hunt *et al.* (1992). Due to their theoretical ability to approximate arbitrary nonlinear mappings, neural networks can be used to build forward or inverse models of a dynamic system. Judged by the control structures, neural direct inverse control and neural internal model control belong to feedback linearizing control while neural model predictive control falls into the category of nonlinear model predictive control. Direct inverse control utilizes an inverse system model. The inverse model is simply cascaded with the controlled system in order that the composed system results in an identity mapping between the desired response and the controlled system output (Hunt *et al.* 1992). Internal model control (IMC) uses both the system forward and the inverse models as elements within the feedback loop. IMC has shown good performance of robustness and stability (Li *et al*., 1995). In nonlinear model predictive control, a neural network model provides prediction of the future plant response over a specified time horizon. Much work has focused on the neural model predictive control (Potocnik and Grabec, 2002; Duarte *et al.*, 2001; Gu and Hu, 2002). The neural networks model is obtained by training the neural network using actual input-output data from the plant under control.

2.4 A New Tool – Support Vector Regression

The Support Vector Machine (SVM) (Schölkopf and Smola, 2002) has been developing very fast in recent years. Like conventional neural networks, SVM has been used by researchers to solve classification and regression problems. One advantage of the SVM over neural networks is that the SVM has only one global minimum. Another advantage is that the training of the SVM is faster than that of neural networks. Because of its universal approximation ability, support vector regression (SVR) can be used to model nonlinear processes, just as neural networks are. The problem formulation of SVR and its training algorithm will be discussed in the next chapter.

Support vector regression has been reported to be used in the control area. Miao and Wang (2002) used a SVR model in nonlinear model predictive control for a SISO system. Suykens *et al.* (2001) used the least squares support vector machines (LS-SVM's) for the optimal control of nonlinear systems. An *N*-stage optimal control problem is incorporated with a least squares support vector machines which is used to map the state space into the action space. Kruif & Vries (2001) has proposed the support vector machine as a learning mechanism in Feed-Forward control.

In this project, the feasibility of applying SVR in control has been explored. A SVR inverse model controller is designed for a nonlinear HVAC system in Chapter 5. Chapter 6 will discuss using the SVR forward model in nonlinear model predictive control for the HVAC system.

Chapter 3 Support Vector Regression

In this chapter, the formulation of support vector regression (SVR) and its training algorithm are discussed in details. We start by giving a brief introduction of the motivations and formulations of a SV approach for regression estimation, followed by a derivation of the associated dual programming problems. Finally the sequential minimal optimization (SMO) algorithm especially the step size derivation will be studied.

3.1 Introduction to Support Vector Machine

A support vector machine (SVM) is a type of model that is optimized so that prediction error and model complexity are simultaneously minimized (Vapnik, 1995). The support vector machine has been developing fast in recent years. SVMs not only have a more solid foundation than artificial neural networks, but are able to serve as a replacement for neural networks and perform as well or better, in a wide variety of fields (Scholköpf and Smola, 2002). The SVM works by mapping the input space into a high-dimensional feature space using kernel tricks. Like conventional neural networks (Haykin 1999), the SVM has been used by researchers to solve classification and regression problems. One advantage of the SVM over neural networks is that the SVM formulates classification and regression as a quadratic optimization problem which ensures that there is only one global minimum whereas the training of neural networks may be “trapped” at a local minimum. Another advantage is that the training of the SVM is faster than that of neural networks. Because of its universal approximation ability, Support vector regression can be used to model nonlinear processes, just as neural networks are.

3.1.1 Basic Ideas

Suppose we are given training data $\{(x_1, y_1), \cdots, (x_l, y_l)\} \subset X \times R$, where X denotes the space of the input patterns---for instance R^m. In ε-SV regression, our goal is to find a function *f(x)* that deviates at most ε deviation from the desired targets y_i for all the training data, and, at the same time, is as flat as possible. In other words, we do not care about errors as long as they are less than ε, but will not accept any deviation larger than this.

We begin by describing the case of linear functions *f(x)*, taking the form of:

$$f(x) = \langle w, x \rangle + b, \text{ with } w \in X, b \in R \qquad (3.1)$$

where $\langle \cdot,\cdot \rangle$ denotes the dot product. Flatness in the case of (3.1) means that one seeks small values for w. One way to ensure this is to minimize the Euclidean norm, i.e. $\|w\|^2$. Formally we can write this problem as a convex optimization problem by requiring:

$$\min \frac{1}{2}\|w\|^2$$

$$\text{subject to} \left|\langle w, x_i \rangle + b - y_i\right| \le \varepsilon \qquad (3.2)$$

The tacit assumption in (3.2) is that such a function f that approximates all pairs (x_i, y_i) with ε precision actually exists or, in other words, that the convex optimization problem is feasible. Sometimes, however, this may not be the case, or we also may want to allow for some errors. Analogously to the "soft margin" loss function in the case of support vector machines for classification, one can introduce slack variables ξ_i, ξ_i^* to cope with otherwise infeasible constraints of the optimization problems (3.2). Hence we arrive at the formulation stated in Eq. (3.3):

$$\min_{w \in \mathrm{H}, \xi^{(*)} \in R^m, b \in R} \quad \tau(w, \xi_i^{(*)}) = \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

$$\text{subject to} \begin{cases} (\langle w, x_i \rangle + b) - y_i \leq \varepsilon + \xi_i \\ y_i - (\langle w, x_i \rangle + b) \leq \varepsilon + \xi_i^* \\ \xi, \xi_i^* \geq 0 \end{cases} \tag{3.3}$$

The constant $C > 0$ determines the trade off between the flatness of f and the amount up to which deviations lager than ε are tolerated. The formulation above corresponds to dealing with a so-called ε-insensitive loss function $|\xi|_\varepsilon$ described by

$$|\xi|_\varepsilon := \begin{cases} 0 & |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \tag{3.4}$$

The ε-insensitive loss function is illustrated in Figure 3.1. Only the points outside the ε tube contribute to the cost function, as the deviations are penalized in a linear fashion. It turns out that the optimization problem (3.3) can be solved more easily in its dual formulation. Moreover, the dual formulation provides the key for extending SVMs to nonlinear functions. Hence we will use the standard dualization method utilizing Lagrange multipliers.

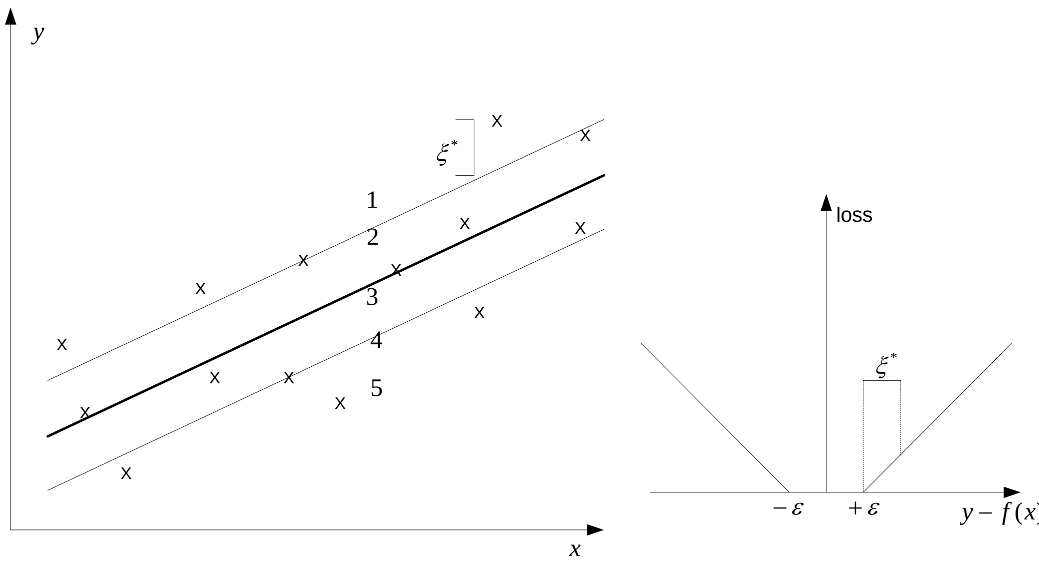

Figure 3.1 ε-insensitive loss function for a linear SV regression

3.1.2 Dual Formulation and Quadratic Programming

The key idea is to construct a Lagrange function from both the objective function (known as the primal objective function) and the corresponding constraints, by introducing dual variables. It can be shown that this function has a saddle point with respect to the primal and dual variables at the optimal solution. The objective is thus to maximize the following Lagrange function with respect to the dual variables.

$$L := \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*) - \sum_{i=1}^{m}(\eta_i\xi_i + \eta_i^*\xi_i^*)$$

$$-\sum_{i=1}^{m}\alpha_i(\varepsilon + \xi_i + y_i - \langle w, x_i\rangle - b) - \sum_{i=1}^{m}\alpha_i^*(\varepsilon + \xi_i^* - y_i + \langle w, x_i\rangle + b) \tag{3.5}$$

where $\{\alpha_i, \alpha_i^*\}$ and $\{\eta_i, \eta_i^*\}$ are the two sets of dual variables such that:

$$\alpha_i, \alpha_i^* \geq 0 \tag{3.6}$$

$$\eta_i, \eta_i^* \geq 0 \tag{3.7}$$

From the saddle point condition, the partial derivatives of L with respect to the primal variables (w, b, ξ_i, ξ_i^*) have to vanish for optimality.

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) = 0 \tag{3.8}$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) x_i = 0 \tag{3.9}$$

$$\frac{\partial L}{\partial \xi} = \frac{C}{m} - \alpha_i - \eta_i = 0 \tag{3.10.a}$$

$$\frac{\partial L}{\partial \xi^*} = \frac{C}{m} - \alpha_i^* - \eta_i^* = 0 \tag{3.10.b}$$

From (3.7) and (3.10), we notice that:

$$\eta_i = \frac{C}{m} - \alpha_i \geq 0 \tag{3.11.a}$$

$$\eta_i^* = \frac{C}{m} - \alpha_i^* \geq 0 \tag{3.11.b}$$

also according to (3.6), the new constraints for α_i, α_i^* are derived as follows:

$$0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{m} \tag{3.12}$$

From (3.9), the two items of $\langle w, x_i \rangle$ and $\|w\|^2$ in (3.5) can be expressed as follows:

$$\langle w, x_i \rangle = \sum_{j=1}^{m} (\alpha_j^* - \alpha_j) \langle x_i, x_j \rangle \tag{3.13}$$

$$\|w\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \langle x_i, x_j \rangle \tag{3.14}$$

Now substituting (3.8), (3.10), (3.13) and (3.14) into (3.5) yields the dual optimization problem.

$$\max_{\alpha^{(*)} \in R^m} \quad -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \langle x_i, x_j \rangle - \varepsilon \sum_{i=1}^{m} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{m} y_i (\alpha_i^* - \alpha_i)$$

$$\text{subject to } \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) = 0 \text{ and } 0 \le \alpha_i, \alpha_i^* \le \frac{C}{m} \tag{3.15}$$

(3.9) can be rewritten as:

$$w = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) x_i \tag{3.16}$$

thus

$$f(x) = \langle w, x_i \rangle + b = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) \langle x_i, x \rangle + b \tag{3.17}$$

This is the familiar *SV expansion*, which states that w can be completely described as a linear combination of a subset of the training patterns x_i. The complete algorithm can be described in terms of dot products between the data. Even when evaluating *f(x)*, we need not compute w explicitly. This will allow the formulation of a nonlinear extension using kernels.

According to KKT conditions (Fletcher, 1987), at the point of solution, the product between dual variables and constraints has to vanish. This gives:

$$\alpha_i (\varepsilon + \xi_i + y_i - \langle w, x_i \rangle - b) = 0 \tag{3.18.a}$$

$$\alpha_i^* (\varepsilon + \xi_i^* - y_i + \langle w, x_i \rangle + b) = 0 \tag{3.18.b}$$

and

$$\eta_i \xi_i = 0 \text{ and } \eta_i^* \xi_i^* = 0 \tag{3.19}$$

Substituting (3.11) into (3.19), we have the following:

$$\left(\frac{C}{m} - \alpha_i\right)\xi_i = 0 \text{ and } \left(\frac{C}{m} - \alpha_i^*\right)\xi_i^* = 0 \tag{3.20}$$

This will allow us to draw some useful conclusion:

Firstly, referring to Figure 3.1 and (3.3), for training points (x_i, y_i) lying below the ε-tube, $\xi_i > 0, \xi_i^* = 0$. From (3.20), for such points $\frac{C}{m} - \alpha_i = 0$. This gives $\alpha_i = \frac{C}{m}$. Also we notice that:

$$\langle w, x_i \rangle + b - y_i = \varepsilon + \xi_i \text{, with } \xi_i > 0 \tag{3.21}$$

This implies that

$$\varepsilon + \xi_i^* - y_i + \langle w, x_i \rangle + b = 2\varepsilon + \xi_i > 0 \tag{3.22}$$

From (3.18.b), we can get $\alpha_i^* = 0$. Similarly, for training points (x_i, y_i) lying above the ε-tube, $\xi_i = 0, \xi_i^* > 0$ (Figure 3.1, Area 1), we can get $\alpha_i = 0$, $\alpha_i^* = \frac{C}{m}$.

Secondly, for points that lie inside the tube (Figure 3.1, Area 3), we have $\xi_i = \xi_i^* = 0$,

$$\varepsilon + \xi_i + y_i - \langle w, x_i \rangle - b > 0 \tag{3.23}$$

$$\varepsilon + \xi_i^* - y_i + \langle w, x_i \rangle + b > 0 \tag{3.24}$$

According to (3.18.a), (3.18.b), we get $\alpha_i = \alpha_i^* = 0$

Thirdly, for those points lying exactly on the lower bound of the tube (Figure 3.1, Area 4), $\xi_i = \xi_i^* = 0$. We notice that:

$$\langle w, x_i \rangle + b - y_i = \varepsilon + \xi_i \text{, with } \xi_i = 0 \tag{3.25}$$

This implies that:

$$\varepsilon + \xi_i^* - y_i + \langle w, x_i \rangle + b = 2\varepsilon + \xi_i + \xi_i^* = 2\varepsilon > 0 \tag{3.26}$$

From (3.18.b), we can get $\alpha_i^* = 0$. From (3.25), we get $\varepsilon + \xi_i + y_i - \langle w, x_i \rangle - b = 0$. From (3.18.a) and (3.12), we can derive that α_i can be any value within the constraint, i.e. $0 \le \alpha_i \le C/m$. Similarly, for points lying exactly on the upper bound of the tube (Figure 3.1, Area 2), we get α_i =0, $0 \le \alpha_i^* \le C/m$. It is worth noting that some of the points lying exactly on the boundary may have zero multipliers, which are not called SVs. Only those boundary points that have non-zero multiplier are called SVs.

From the above discussion, we conclude that $\alpha_i \alpha_i^* = 0$ in all cases. In other words, a set of dual variables α_i and α_i^* cannot be simultaneously nonzero. The summary of the above analysis is shown in Table 3.1

Table 3.1 Lagrange multipliers in different areas

Area	ξ_i	ξ_i^*	α_i	α_i^*
1	0	ξ_i^*>0	0	C/m
2	0	0	0	$0 \le \alpha_i^* \le C/m$
3	0	0	0	0
4	0	0	$0 \le \alpha_i \le C/m$	0
5	ξ_i>0	0	C/m	0

For those points in Area 2 or Area 4 in Figure 3.1, we have $\xi_i = 0$ and $\xi_i^* = 0$, b can be computed as follows:

$$\begin{aligned} b &= y_i - \langle w, x_i \rangle + \varepsilon \quad \text{for} \quad 0 \le \alpha_i \le C/m \\ b &= y_i - \langle w, x_i \rangle - \varepsilon \quad \text{for} \quad 0 \le \alpha_i^* \le C/m \end{aligned} \tag{3.27}$$

Now consider the sparsity of the SV expansion. From (3.13), we can see that the multipliers can be nonzero only for the points $|f(x) - y_i| \ge \varepsilon$. That means for those examples lying inside the tube, the multipliers of α_i, α_i^* vanish. The examples with nonzero multipliers are called the Support Vectors. In other words, we can remove any points inside the tube and still get the same result. After training, only those examples with nonzero of α_i and α_i^* are support vectors and will enter into the model. So the number of support vectors must be less than or equal to the number of training examples.

3.1.3 Nonlinear Regression and Kernel Tricks

The regression estimate which takes the form of (3.12) is actually a linear regression. For a nonlinear case, we can use kernel tricks to map the input vector in the original space to a higher dimensional feature space and obtain the following:

$$f(x) = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) \langle \Phi(x_i), \Phi(x) \rangle + b = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) K(x_i, x) + b \tag{3.28}$$

$\Phi(x_i)$ and $\Phi(x)$ are the mapped vectors of x_i and x in the feature space respectively. K is referred to as the kernel function which is used to compute the dot product of two feature vectors of $\Phi(x_i)$ and $\Phi(x)$ without actually forming those feature space vectors. As long as a given function satisfies Mercer's Theorem (Schölkopf and Smola, 2002), it can be used as a kernel function. Inhomogeneous polynomial kernel, homogeneous polynomial kernel, Gaussian kernel and sigmoid kernel are the commonly used kernel functions. In this project, the most popular Gaussian kernel is used. An interesting aspect of Gaussian kernel is that its corresponding feature space is infinite dimensional. Gaussian kernel has the following form:

$$K(x,\tilde{x}) = \exp(-\|x-\tilde{x}\|^2 / 2\sigma^2) \tag{3.18}$$

When we choose the Gaussian kernel function, there are three tuning parameters, and these are the width parameter σ in the kernel function, ε for ε-insensitive loss function and the penalty constant C.

(a) The width parameter σ is used to control the power of the feature space. When σ is very small, x and $\tilde{x}$ do not "interact" even when they are reasonably close. Small values of σ lead to very powerful feature spaces. On the other hand, when σ is large, x and $\tilde{x}$ have "interaction" even when they are far away from each other (Keerthi, 2002).

(b) ε can control the range of the ε-insensitive loss function. The smaller the value of ε is, greater accuracy is obtained by learning the training examples. However, too small an ε value will force the SVR to remember the noise in the training examples, thus sacrificing the generalization property of SVR. A good choice of ε should be a trade-off between the training accuracy and the generalization property.

(c) C is a constant determining the trade-off with the complexity penalizer $\|w\|^2$ (Scholköpf and Smola, 2002). In short, minimizing (3.3) captures the main idea of statistical learning theory: in order to obtain a good generalization performance, it is necessary to control both training error and model complexity, by explaining the data with a simple model. Large values of C will not tolerate errors while small values will allow too many errors. So an intermediate choice of C needs to be found.

ε, σ and C are also are referred to as hyperparameter values in literature. Choosing optimal hyperparameter values for support vector machine is an important step in SVM design (Duan *et al*., 2003). This is usually done by minimizing either an estimate of the generalization error or some other related performance measure. In this project, *k*-fold cross validation is used to choose the optimal value of σ and C while ε is set to 0.001.

3.2 Sequential Minimal Optimization

SVM can be optimized by decomposing a large quadratic programming (QP) problem into a series of smaller QP subproblems. Optimizing each subproblem minimizes the original QP problem in such a way that once no further progress can be made with all the smaller subproblems, the original QP problem is solved. Many experimental results indicate that decomposition can be much faster than QP. More recently, the sequential minimal optimization algorithm (SMO) was introduced (Platt, 1998) as an extreme example of decomposition. It puts decomposition of the original problem to the extreme by iteratively selecting subsets of size two and optimizing the target function with respect to them. The key point of SMO is that for a subset of size two, the optimization subproblem can be solved analytically without resorting to a quadratic programming (QP) solver. SMO has been shown to be an effective method for training support vector machines.

Smola and Schölkopf (1998) derived regression rules for SMO that use four separate Lagrange multipliers, where one pair of multipliers forms a single composite parameter. Properly handling all special cases for the four Lagrange multipliers is somewhat difficult as two pages of pseudo-codes are required to describe the update rule. Its modified version for regression was given by Shevade *et al.* (2000). Flake and Lawrence (2002) has made one of the simplest and most complete derivations of SMO the regression algorithm and gave some heuristics that can improve the convergence time by over an order of magnitude. While other existing SMO regression algorithms compute the two multipliers of each training example, Flake and Lawrence (2002) combined the two multipliers into one variable so that the implementation becomes more concise and understandable.

SMO actually consists of two parts: (1) a set of heuristics for efficiently choosing pairs of Lagrange multipliers to work with, and (2) the analytical solution to a QP problem of size two. Here only (2) will be introduced; more details of (1) are given in Flake and Lawrence (2002).

3.2.1 Step Size Derivation

We begin by substituting $\lambda_i = \alpha_i - \alpha_i^*$, and $|\lambda_i| = \alpha_i + \alpha_i^*$. Thus, the new unknowns will obey the box constraints $-C \le \lambda_i \le C, \forall i$. We will also use the shorthand $k_{ij} = K(x_i, x_j)$ and always assume that $k_{ij} = k_{ji}$. The model output and objective function can now be written as

$$f(x,\lambda,b) = \sum_{i=1}^{l} \lambda_i K(x_i, x) + b, \tag{3.19}$$

$$W(\lambda) = \varepsilon \sum_{i=1}^{l} |\lambda_i| - \sum_{i=1}^{l} \lambda_i y_i + \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \lambda_i \lambda_j k_{ij} \tag{3.20}$$

with linear constraints $\sum_{i=1}^{l} \lambda_i = 0$. Our goal is to analytically express the minimum of Eq. (3.20) as a function of two parameters. Let these two parameters have indices *u* and *v* so that λ_u and λ_v are the two unknowns. We can rewrite Eq. (3.20) as

$$\begin{aligned} W(\lambda_u, \lambda_v) = {} & \varepsilon|\lambda_u| + \varepsilon|\lambda_v| - \lambda_u y_u - \lambda_v y_v + \frac{1}{2}\lambda_u^2 k_{uu} + \frac{1}{2}\lambda_v^2 k_{vv} \\ & + \lambda_u \lambda_v k_{uv} + \lambda_u z_u^* + \lambda_v z_v^* + W_c \end{aligned} \tag{3.21}$$

where W_c is a term that is strictly constant with respect to λ_u and λ_v ,and z_i^* is defined as:

$$z_i^* = \sum_{j\neq u,v}^{l} \lambda_j^* k_{ij} = f_i^* - \lambda_u^* k_{ui} - \lambda_v^* k_{vi} - b^* \tag{3.22}$$

with $f_i^* = f(x_i, \lambda^*, b)$. Note that superscript * is used to indicate that values are computed with the old parameter values. If we assume that the constraint, $\sum_{i=1}^{l} \lambda_i = 0$, is true prior to any change to λ_u and λ_v, then in order for the constraints to be true after a step in parameter space, the sum of λ_u and λ_v must be held fixed. With this in mind, let $s^* = \lambda_u + \lambda_v = \lambda_u^* + \lambda_v^*$. We can now rewrite Eq. (3.21) as a function of a single Lagrange multiplier by substituting $\lambda_u = s^* - \lambda_v$:

$$\begin{aligned} W(\lambda_v) = {} & \varepsilon|s^* - \lambda_v| + \varepsilon|\lambda_v| - (s^* - \lambda_v)y_u - \lambda_v y_v + \frac{1}{2}(s^* - \lambda_v)k_{uu} + \frac{1}{2}\lambda_v^2 k_{vv} \\ & + (s^* - \lambda_v)\lambda_v k_{uv} + (s^* - \lambda_v)z_u^* + \lambda_v z_v^* + W_c \end{aligned} \tag{3.23}$$

To solve Eq. (3.23), we need to compute its partial derivative with respect to λ_v; however, Eq. (3.23) is not strictly differentiable because of the absolute value function. If we take $d|x|/dx = \text{sgn}(x)$, the resulting derivative is algebraically consistent:

$$\begin{aligned} \frac{\partial W}{\partial \lambda_v} = {} & \varepsilon(\text{sgn}(\lambda_v) - \text{sgn}(s^* - \lambda_v)) + y_u - y_v \\ & + (\lambda_v - s^*)k_{uu} + \lambda_v k_{vv} + (s^* - 2\lambda_v)k_{uv} - z_u^* + z_v^* \end{aligned} \tag{3.24}$$

Now setting Eq. (3.24) to zero yields:

$$\begin{aligned}
&\lambda_v(k_{vv}+k_{uu}-2k_{uv})\\
&= y_v - y_u + \varepsilon(\operatorname{sgn}(\lambda_u)-\operatorname{sgn}(\lambda_v)) + s^*(k_{uu}-k_{uv}) + z_u^* - z_v^*\\
&= y_v - y_u + f_u^* - f_v^* + \varepsilon(\operatorname{sgn}(\lambda_u)-\operatorname{sgn}(\lambda_v)) + \lambda_u^* k_{uu}\\
&\quad -\lambda_u^* k_{uv} + \lambda_v^* k_{uu} - \lambda_v^* k_{uv} - \lambda_u^* k_{uu} - \lambda_v^* k_{uv} - b^* + \lambda_u^* k_{uv} + \lambda_v^* k_{vv} + b^*\\
&= y_v - y_u + f_u^* - f_v^* + \varepsilon(\operatorname{sgn}(\lambda_u)-\operatorname{sgn}(\lambda_v)) + \lambda_v^*(k_{vv}+k_{uu}-2k_{uv})
\end{aligned} \tag{3.25}$$

From Eq. (3.25), we can write a recursive update rule for λ_v in terms of its old value:

$$\lambda_v = \lambda_v^* + \frac{1}{\eta}(y_v - y_u + f_u^* - f_v^* + \varepsilon(\operatorname{sgn}(\lambda_u)-\operatorname{sgn}(\lambda_v))) \tag{3.26}$$

where $\eta = k_{vv}+k_{uu}-2k_{uv}$.

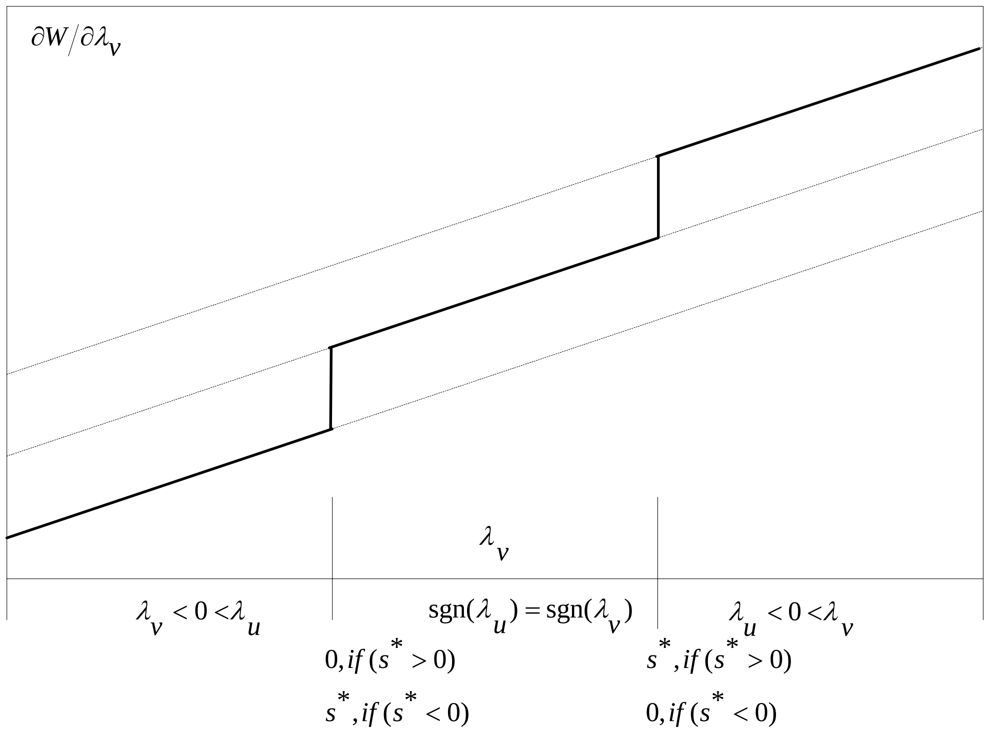
$\partial W/\partial\lambda_v$
λ_v
$\lambda_v < 0 < \lambda_u$
$\operatorname{sgn}(\lambda_u) = \operatorname{sgn}(\lambda_v)$
$\lambda_u < 0 < \lambda_v$
$0, if\,(s^* > 0)$
$s^*, if\,(s^* < 0)$
$s^*, if\,(s^* > 0)$
$0, if\,(s^* < 0)$

Figure 3.2 The derivative as a function of λ_v

3.2.2 Finding Solutions

Figure 3.2 illustrates the behavior of the partial derivative (Eq. 3.24) of the objective function with respect to λ_v. If the kernel function of SVM obeys Mercer's condition (as all common ones do), then it will be guaranteed that $\eta = k_{vv} + k_{uu} - 2k_{uv} \geq 0$ is true (Schölkopf and Smola, 2002). If η is strictly positive, then Eq. (3.24) will always be increasing. Moreover, if s^* is not zero, it will be piecewise linear with two discrete jumps, as illustrated in Figure 3.2. Putting these facts together means that we only have to consider five possible solutions for Eq. (3.24), three possible solutions correspond to using Eq. (3.26) with $(\text{sgn}(\lambda_u) - \text{sgn}(\lambda_v))$ set to -2, 0 and 2. The other two candidates correspond to setting λ_v to one of the transitions in Figure 3.2: $\lambda_v = 0$ or s^*. The update rules for λ_u and λ_v must also ensure that both parameters take values within $\pm C$.

Table 3.2 shows the pseudo-code that implements a single step for SMO with regression. Basically, lines 4 and 5 set the Lagrange multipliers to values appropriate if the two have the same sign. If the two multipliers differ in sign, then line 8 adjusts the multipliers by $2\varepsilon/\eta$ if the adjustment can be made without affecting the sign of either multiplier. If such an adjustment cannot be made, the only solution is for the two multipliers to take the values of s^* and 0. Lines 12 and 13 calculate boundaries that keep both multipliers within $\pm C$. Finally, lines 14 and 15 enforce the constraints. Lines 16 and 17 get the four separate multipliers at a single step size.

A set of heuristics given in a modified version of SMO (Shevade *et al.*, 2000) is followed for efficiently choosing pairs of Lagrange multipliers to work with. When no pairs of multipliers have been found to have violated the optimality criteria, the training of SVR is finished.

Table 3.2 Pseudo-code for analytical step for SMO generalized for regression

```
1.  $s^* = \lambda_u^* + \lambda_v^*$;
2.  $\eta = k_{uu} + k_{vv} - 2k_{uv}$;
3.  $\Delta = 2s/\eta$;
4.  $\lambda_v = \lambda_v^* + \frac{1}{\eta}(y_v - y_u + f_u^* - f_v^*)$;
5.  $\lambda_u = s^* - \lambda_v$;
6.  if($\lambda_u \cdot \lambda_v$)<0{
7.      if($|\lambda_v| \geq \Delta \,\&\&\, |\lambda_u| \geq \Delta$)
8.          $\lambda_v = \lambda_v - \mathrm{sgn}(\lambda_v) \cdot \Delta$;
9.      else
10.         $\lambda_v = \mathrm{step}(|\lambda_v| - |\lambda_u|) \cdot s^*$;
11. }
12. $L = \max(s^* - C, -C)$;
13. $H = \min(C, s^* + C)$;
14. $\lambda_v = \min(\max(\lambda_v, L), H)$;
15. $\lambda_u = s^* - \lambda_v$;
16. $\alpha_u = \max(\lambda_u, 0), \alpha_u^* = \max(-\lambda_u, 0)$
17. $\alpha_v = \max(\lambda_v, 0), \alpha_v^* = \max(-\lambda_v, 0)$
```

Chapter 4 System Identification with Support Vector Regression

In this project, inverse control and model predictive control for the HVAC system will be investigated. Both control strategies are model-based. For inverse control, the inverse dynamic model is needed while for model predictive control, the forward dynamic model is needed. In this chapter, the inverse and forward dynamics of the HVAC system will be modeled by support vector regression.

4.1 HVAC System

A simple diagram of the experimental system is depicted in Figure 4.1. A fan and a motor equipped with a variable speed drive (VSD) are installed in an air handling unit (AHU) to provide a variable airflow rate. The frequency of the input electrical power to VSD varies from 0-37.5Hz, modulated by a frequency converter. Therefore the speed of the fan motor can be varied. The flow rate of the air is assumed to increase with an increase in fan speed. Variable airflow can be made to pass through the cooling coils by varying the fan speed. Two three-way regulating valves are appended to the chilled water pipe. The valves are used in conjunction with an actuator to provide completely close to fully open operation. The actuator’s 0-10 VDC signal corresponds to the valve’s 0-100% opening. Varying the valve opening modifies the chilled water flow rate to the AHU, with 100% opening corresponding to the maximum flow rate.

An electrical resistance heater and a humidifier (The humidifier is simply a container of heated water) are used to simulate the sensible and latent heat load in the conditioned room respectively. The rate of heat emitted from the electrical heater can be varied by resetting the temperature of the thermostat. Two sensors are installed at the end of the supply air duct. These are capable of measuring the supply air temperature and relative humidity. The room temperature and relative humidity are measured by two RTD sensors.

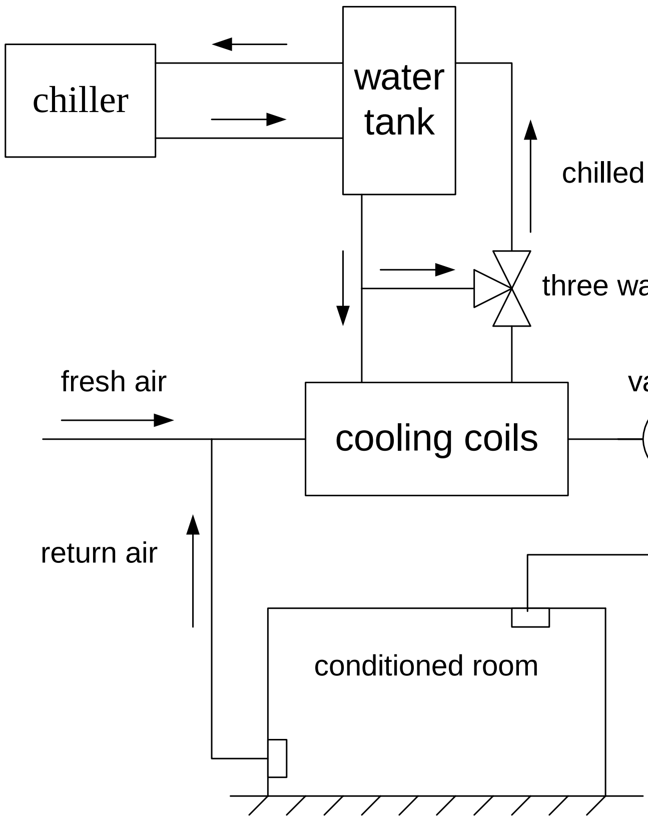

Figure 4.1 Simple diagram of the experimental HVAC system

The start and stop operation of the chiller and dampers are controlled by a METASYS building management system. A Pentium III 800 MHz PC with an ISA bus Servo To Go interface card is used for the experiments. The supply air temperature and relative humidity, and the room temperature and relative humidity are measured and recorded as well as the valve openings and fan speeds. The PC performs the data acquisition and controls the frequency converter and valve actuators.

4.2 System Identification

The heart of model-based control is the process model itself. Models can be classified by various features. The form of the model selected has a large impact on its ability to implement these control strategies. Some choices must be made in the identification process.

Linear or nonlinear: The experimental HVAC plant consists of many mechanical, hydraulic and electrical components. So the overall dynamics of the system is nonlinear. Although some linear models such as state space, transfer function can be used, they are only applicable around certain operating points.

Continuous-time or discrete time: Most of the physical laws that are used by engineers to develop models are presented as differential equations with time as the independent variable. Before the widespread availability of digital computers, differential equation models were the central tool for the study of dynamic systems. With the availability of cheap and fast computers, the study of difference equation, which has been previously relegated to a minor role, assumed a new significance. A nonlinear difference equation can be expressed as $x_{k+1} = f(x_k, u_k)$. Since both inverse control and MPC are implemented via digital computer, difference equations are the models of choice.

First-principles or “black box”: Models that are derived from heat, mass and momentum balances are frequently called “first-principles” models, in contrast to other data fitting schemes. As mentioned above, the HVAC system consists of many mechanical, hydraulic and electrical components, so it is very difficult to built “first-principles” model. An easy and practical way is to model the system based on the data collected within the operating range of the system. A first principle model can presumably predict over a wide range of conditions, even without prior operating experience. On the other hand, the “black box” method has its own disadvantages. The data-based model lacks its predictive value outside the range of operating conditions where data has been collected.

In this project, a new approach--support vector regression will be used to build the dynamic models of the HVAC system. In later chapters, these models will be used to implement two model-based control strategies, i.e. SVR inverse control and SVR model predictive control.

4.2.1 Sampling Interval

One important issue in modeling is the sampling interval. When a model is meant for control purpose, the sampling interval for model development must be the same as that for control application (Ljung, 1999). When choice is available, sampling interval should be taken fairly short compared with the time constant of the system. A sampling interval that is much larger than the significant time constants of the system would yield data with little information about the dynamics. A small sampling interval, on the other hand, would not allow for much noise reduction, and the data might be less informative for that reason. A good choice of sampling interval should be a trade-off between noise reduction and relevance for dynamics. In practice, it is useful to first record a step response from the system, and then select the sampling interval based on these responses.

In this project, the two controlled variables are room temperature and room relative humidity. From the step response of the system (Figure 4.2, Figure 4.4), we find that the dynamics of room relative humidity is faster than that of room temperature. Because simultaneous control of both room temperature and room relative humidity is required, the sampling interval is determined by the faster dynamics, i.e. dynamics of room relative humidity. Isermann (1981) recommended that sampling interval time can be chosen between 1/15 and 1/4 of T_{95} (the time needed by the system to reach 95 % of the final output value).

First, we will check the supply air fan step response. The normalized fan input was set to 0.7 from 0.4 at the 30th sampling instant. The room relative humidity rises from the value of 0.765 at the 30th sample to its final value 0.862 at the 149th sample. So T_{95} value=0.765+0.95*(0.862-0.765)=0.8571, which is reached at the 122th sample. Each sample is 10 seconds. The rise time is therefore equal to (122-30)*10=920 seconds. The choice of sampling interval should be within the range of 61-230 seconds.

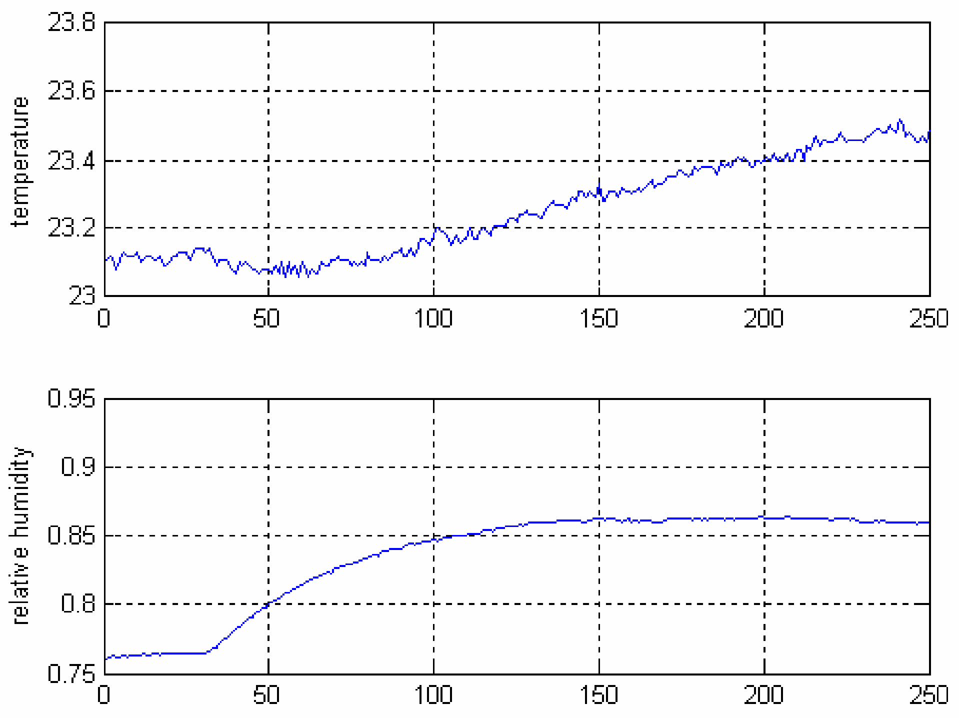

Figure 4.2 Fan step responses of temperature and RH

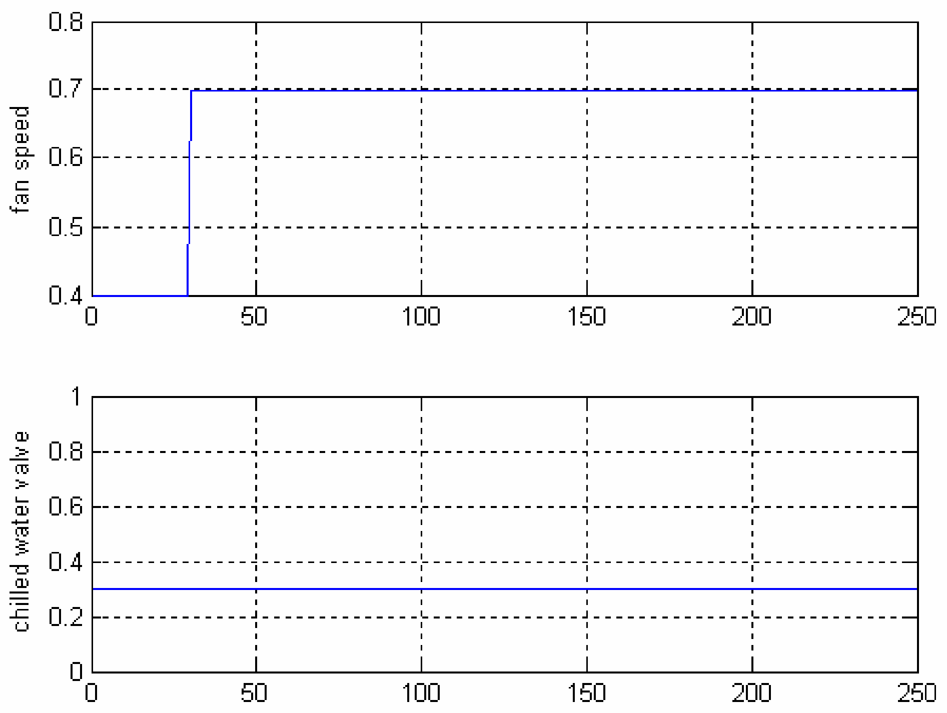

Figure 4.3 Fan step change

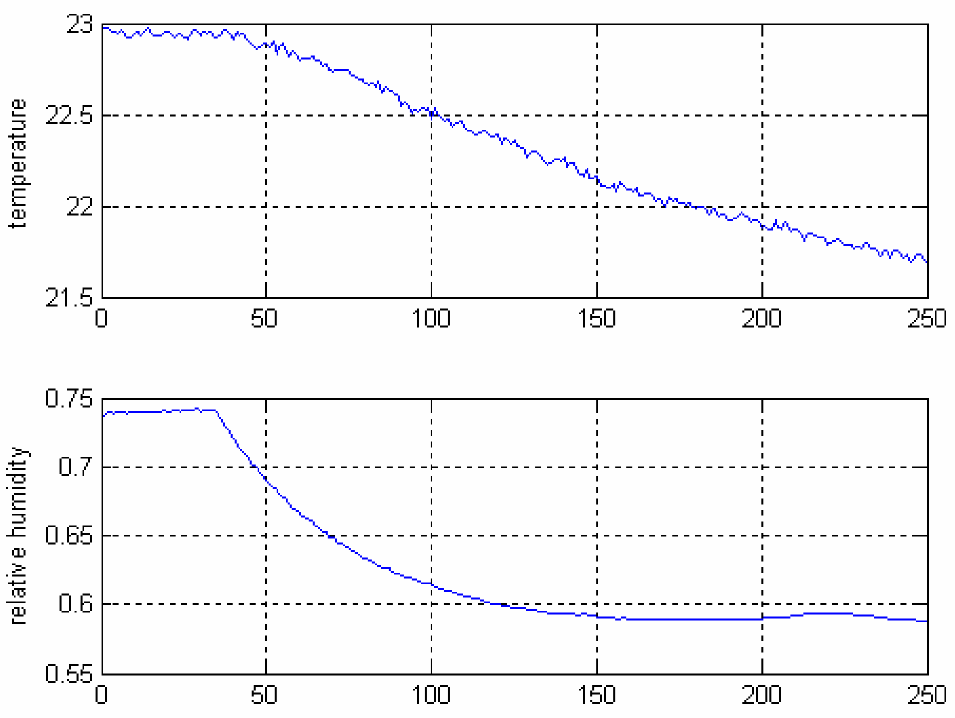

Figure 4.4 Valve step responses of room temperature and RH

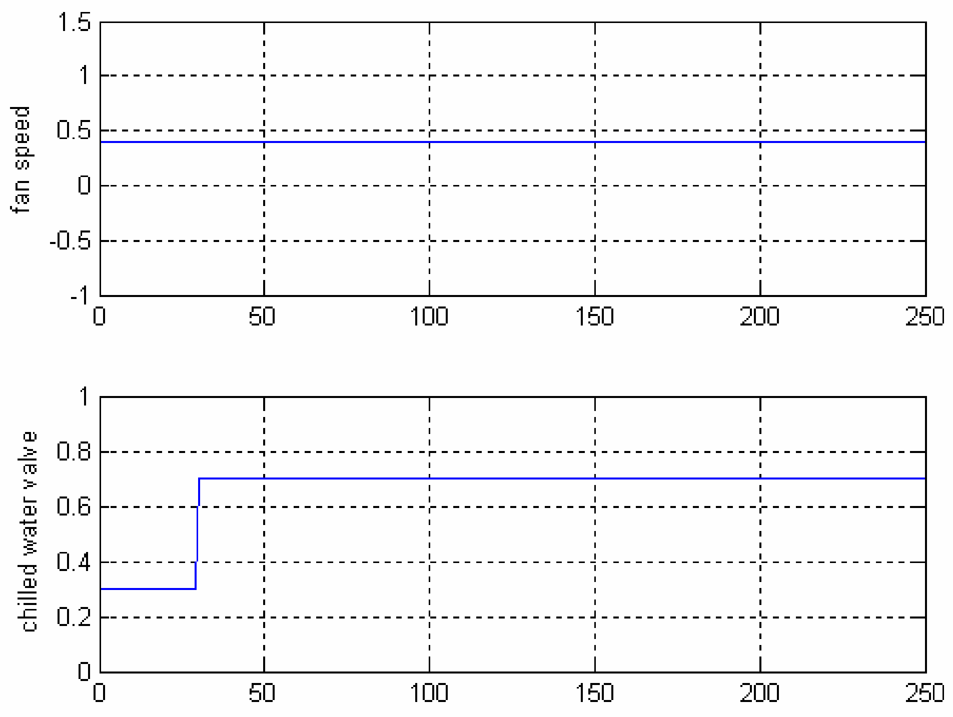

Figure 4.5 Valve step change

Now we will check the chilled water valve step response. The chilled water valve was changed from 0.3 to 0.7 opening at the 30th sample. The room relative humidity changes from its initial value of 0.740 to its final value of 0.589 at the 165th sample. So the T_{95}=0.740-0.95*(0.740-0.589)=0.5965, which is reached at the 127th sample. So T_{95} rise time is (127-30)*10=970 seconds. By the step response of chilled water valve, the choice of sampling interval should be within the range of 65 – 243 seconds. Both step responses give the similar range of choice for sampling interval. Finally, a relatively shorter sampling interval of 60 seconds is chosen throughout this project.

4.2.2 Training Data

From the step responses of both supply air fan and chilled water valve, we can notice that both control input signals will affect the room temperature and relative humidity. So it is a strongly coupled system. For the purpose of identification, the system must be excited fully within its operating range. In this project, uniformly distributed random signals were used to excite the system. The supply air fan was varied in the range of 0-0.75 of the rated speed of 3000 rpm. The chilled water valve was varied in the range of 0-100 % of its full opening.

The data was collected over 4 days. In order to collect as informative data as possible within the system’s operating range, each day the HVAC system was started from a different initial condition. The heater and boiler inside the thermal chamber were used to increase the room temperature and relative humidity, respectively.

4.2.3 NARX Models

NARX denotes Nonlinear ARX (auto-regressive models with exogenous inputs). A NARX model is a subset of the general NARMAX model (MA stands for "Moving Average") in which additional moving average terms are present for modeling the stochastic components of a dynamic process. In a sampled data system where only input and output are available, a dynamic process is usually described by the following equation:

$$y(k) = F_p(y(k-1), y(k-2), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-m+1)) + noise \quad (4.1)$$

where y and u are vectors that represent the system outputs and inputs respectively. Subscript p denotes the plant. The procedure for identifying a model for (4.1) depends on the assumption of the characteristics of the noise. Conventionally, one takes the following NARX model as a representation of the process shown in Eq. (4.1)

$$y(k) = F(y(k-1), y(k-2), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-m+1)) \quad (4.2)$$

In this case, the plant output y is used directly as the input to the model as indicated in the parenthesis of Eq. (4.2).

4.2.4 SVR NARX Modeling

In most of the applications to date, a multilayer feedforward network is employed as a nonlinear NARX in which the network used a number of past plant inputs and outputs to predict the future output. In this project, Support Vector Regression (SVR) will be used as a new tool to build a NARX model for a nonlinear dynamic process.

Compared with the neural network models, the SVR model has its own advantages. The first advantage is that the training of SVR is normally faster than that of neural networks. This will be desirable in a situation where an online model should be developed. The second advantage is that the SVR model is the training result after the original SVR optimization problem reaches its global minimum. This is due to a fact that SVR formulates regression as a quadratic optimization problem which ensures that there is only one global minimum while the training of neural networks may stop at a local minimum. For neural networks, apart from global minima in the weight space, there are also many local minima where the value index is bigger than the value index at the global minima (Haykin, 1999). Most optimization algorithms (such as back-propagation) are based on gradient descent and it is most likely that they will get “trapped” when they reach a local minimum point since the gradient at that point is zero. So mathematically speaking, the SVR model has some nicer properties than the neural network model. The third advantage is that the number of tuning parameters for SVR training is fewer than that for the neural network. When training a neural network model, the first thing that should be decided is how to choose the structure of neural networks: how many layers, how many neurons per layer, for how many epochs the neural networks should be trained. On the other hand, for SVR, if the type of kernel is determined, there are only three tuning parameters. The fourth advantage is the generalization property.

Generalization refers to how well the trained model performs on unseen examples. Even when the network has been trained satisfactorily with respect to the training set, it is still possible that the trained neural network gives very poor accuracy on the unseen examples. This is attributed to the fact that the neural network has been over-trained, i.e. it remembered the noise within the training set. While preventing over-training of neural networks needs a lot of careful considerations, with SVR it is relatively easier to achieve generalization. Given a certain set of tuning parameters, we try to obtain a SVR model that gives a minimal prediction error by a minimal complexity of the model. In the formulation of SVR, C is a constant determining the trade-off with the complexity penalizer $\|w\|^2$. In short, the formulation of SVM captures the main insight of statistical learning theory, i.e. in order to obtain a good generalization, we need to control both training error and model complexity, by explaining the data with a simple model (Schölkopf and Smola, 2002).

As mentioned in Chapter 3, choosing optimal hyper-parameter values for support vector machines is an important step in SVM design. This is usually done by minimizing either an estimate of generalization error or some other related performance measure (Duan *et al.*, 2002). It was found that *k*-fold cross-validation is very reliable to find the optimal hyper-parameters. Cross-validation is a popular technique for estimating generalization error and there are several versions. In *k-fold cross-validation*, the training data is randomly split into *k* mutually exclusive subsets (the folds) of approximately equal size. The SVM decision rule is obtained using *k*-1 subsets and then tested on the subset left out. This procedure is repeated *k* times and in this fashion each subset is used for testing purpose once. Averaging the test error over the *k* trials gives an estimate of the expected generalization error.

4.3 Obtaining Forward Dynamic Model of HVAC System

Suppose we are dealing with Multi-Input-Multi-Output (MIMO) system. After stimulating the system with random input signals within the operating range of the system, sufficient input-output data has been collected which can be used as the training set for SVR models of the system.

The result of training the SVR is in the form of

$$y_i = \sum_{j=1}^{m_i} \alpha_{ij} K(v_{ij}, v) + b_i,\ i = 1, \dots, n \tag{4.1}$$

where n is number of outputs, $y_i, i = 1, \dots, n$ are the outputs. K is the kernel function. $v_{ij}, j = 1, \dots, m_i$ are the support vectors in the model.

If we use the Gaussian Radial Basis Function (RBF) kernels, the expression could be written explicitly as follows:

$$y_i = \sum_{j=1}^{m_i} \alpha_{ij} \exp\left(-\frac{\left\| v - v_{ij} \right\|^2}{2\sigma^2} \right) + b_i,\ i = 1, \dots, n \tag{4.2}$$

$$v = \begin{bmatrix} x \\ u \\ d \end{bmatrix} \tag{4.3}$$

where v is the vector that contains state variables x, input u, and measurable disturbances d. x, u and d are column vectors. For a MIMO system, n individual SVR models are needed for *n* output variables. Each output variable has its own independent SVR model which has a different number of support vectors and different corresponding weights from models for other output variables.

For the specific dynamics of the air-conditioning system used in this project, the output of the plant at any sampling interval can generally be written as

$$\begin{aligned}&RT(k+1) = F_1[RT(k), RRH(k), ST(k), SRH(k), Fan(k),\\ &Valve(k), RT(k-1), RRH(k-1), ST(k-1), SRH(k-1),\\ &Fan(k-1), Valve(k-1), RT(k-2), RRH(k-2), ST(k-2),\\ &SRH(k-2), Fan(k-2), Valve(k-2)]\end{aligned} \tag{4.4}$$

$$\begin{aligned}&RRH(k+1) = F_2[RT(k), RRH(k), ST(k), SRH(k), Fan(k),\\ &Valve(k), RT(k-1), RRH(k-1), ST(k-1), SRH(k-1),\\ &Fan(k-1), Valve(k-1), RT(k-2), RRH(k-2), ST(k-2),\\ &SRH(k-2), Fan(k-2), Valve(k-2)]\end{aligned} \tag{4.5}$$

where k is the sampling instant, RT is the room temperature, RRH is the room relative humidity, ST is the supply air temperature, SRH is the supply air relative humidity, Fan is the supply air fan speed, $Valve$ is the chilled water opening and F_1, F_2 are nonlinear functions in the form of support vector regression.

As we see from Eq. (4.4) and (4.5), the temperature dynamic model F_1 and the relative humidity dynamic model F_2 share the same input vector v which has 18 elements. Each feature element will contribute to a value of kernel function. One thing that should be noted is that different feature elements have different ranges. For the case of input vector of F_1, for example, the first element of room temperature has a range of 17- 30 ^0C while the second element of room relative humidity has a range of 40% to 99%. Every element will affect a value of kernel through the form of square of error. If some elements have larger original absolute values than others, their influence will dominate the final kernel value. It is necessary, therefore, to do some prepossessing to the raw data before feeding them into the SVR model. In this project, all the feature elements and the target values are scaled so that they fall in the range of [-1, 1]. When using these SVR models, the computed target value should be converted back into the same scales that were used for the original targets values.

For convenience, we use a composite hyperparameter $g = 1/(2\sigma^2)$. As discussed before, *k*-fold cross validation was used to determine the optimal values of *C* and *g* for the room temperature and relative humidity dynamic models. Here *k* is chosen as 5. We first search for the optimal values of *C* and *g* in relatively coarse grids, followed by a search in finer grids. Finally we obtained the optimal value of *C* and *g* for these two models. Figure 4.6 is the contour of cross validation squared correlation coefficients for temperature dynamics and Figure 4.7 is for relative humidity dynamics. *C* and g are searched in the range of $C = 2^p$, $p = -5,-3,\cdots,15$ and $g = 2^q$, $q = -15,-13,\cdots,3$. From these contours of raw search, we can find smaller area where a finer search can be carried out.

For room temperature dynamics, a finer search was made in the range of $C = 2^p$, $p = 11,12,\cdots,15$ and $g = 2^q$, $q = -9,-10,\cdots,-13$. Finally the highest value was found to be 0.977311 which is obtained at $C = 2^{12} = 40962$ and $g = 2^{-12} = 2.4414E-4$.

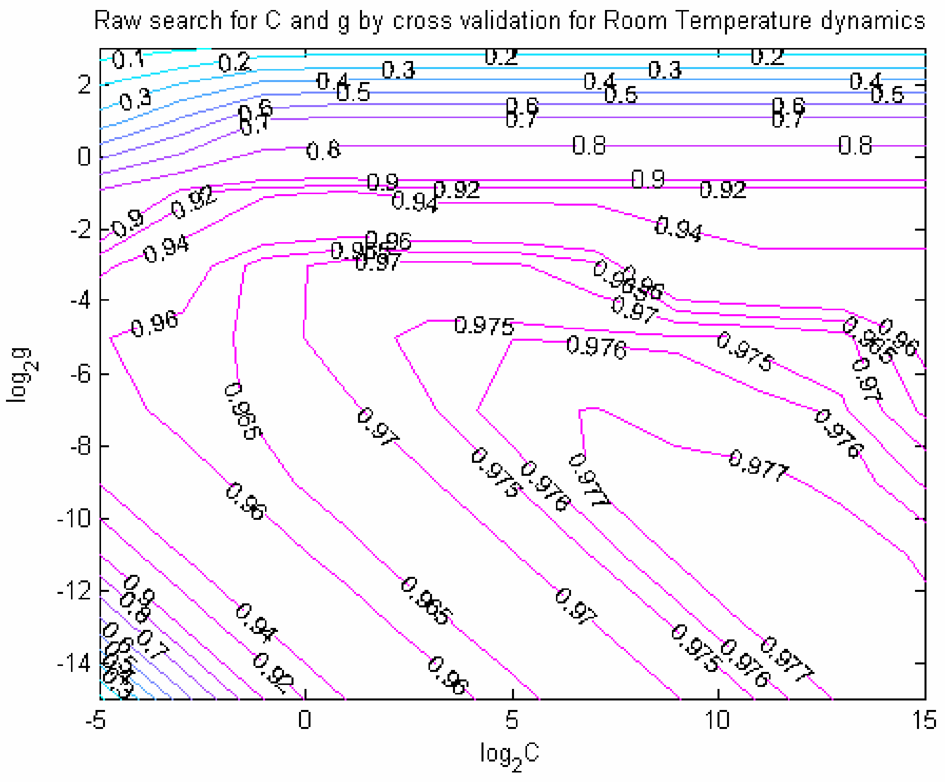

Figure 4.6 Raw search of C and g for temperature dynamics

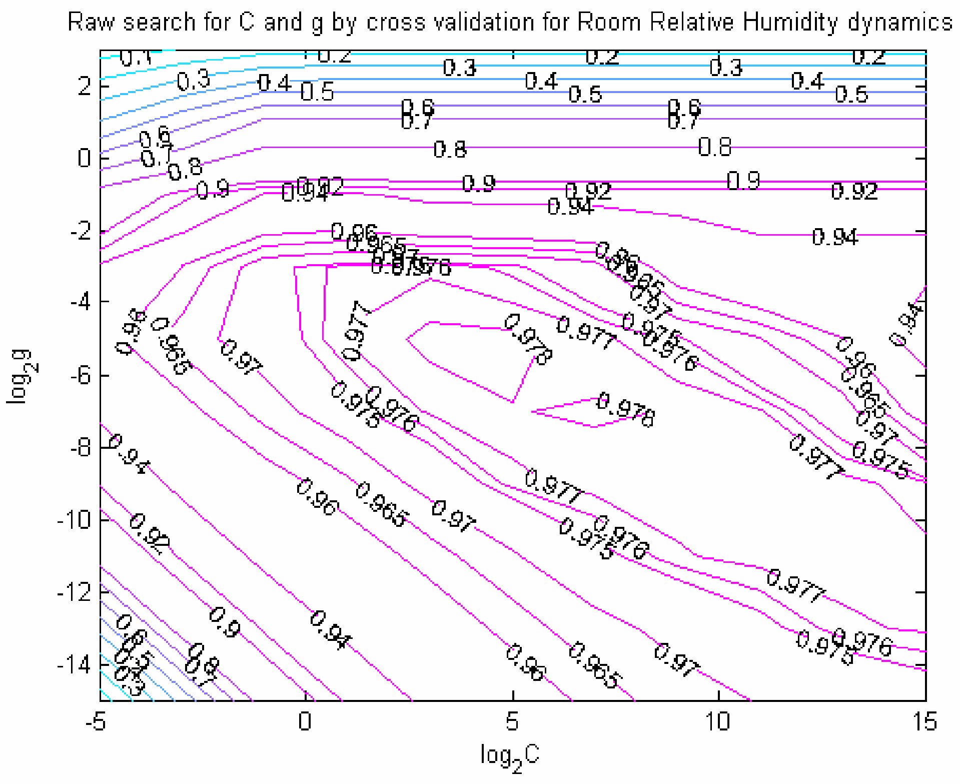

Figure 4.7 Raw search of C and g for RH dynamics

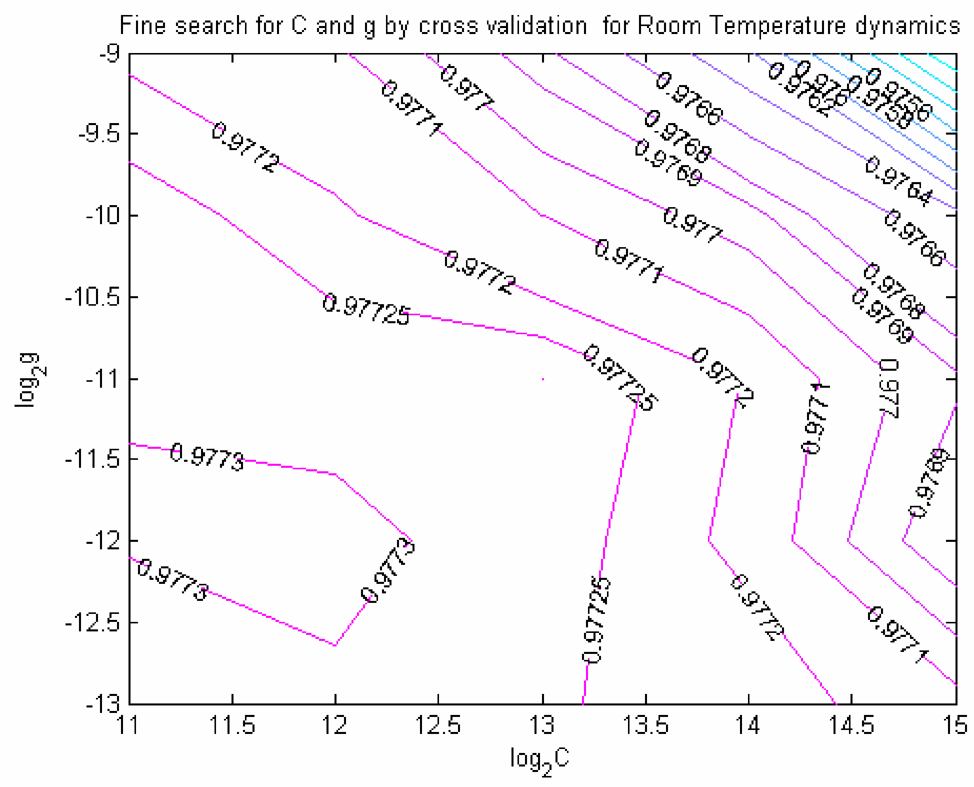

Figure 4.8 Fine search of C and g for temperature dynamics

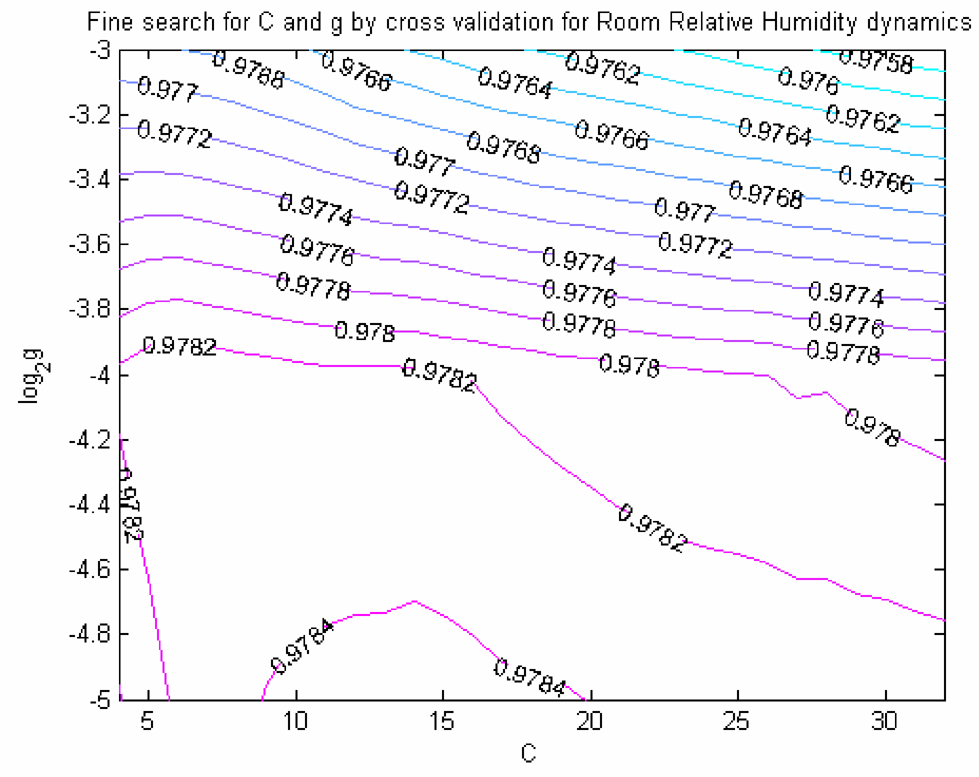

Figure 4.9 Fine search of C and g for RH dynamics

For relative humidity dynamics, a finer search was made in the range of $C = 4,5,\cdots,32.$ and $g = 2^q, q = -5,-4,-3$. The best value was found at C=14 and $g = 2^{-5} = 0.03125$, and the corresponding cross validation squared correlation coefficient is 0.9785.

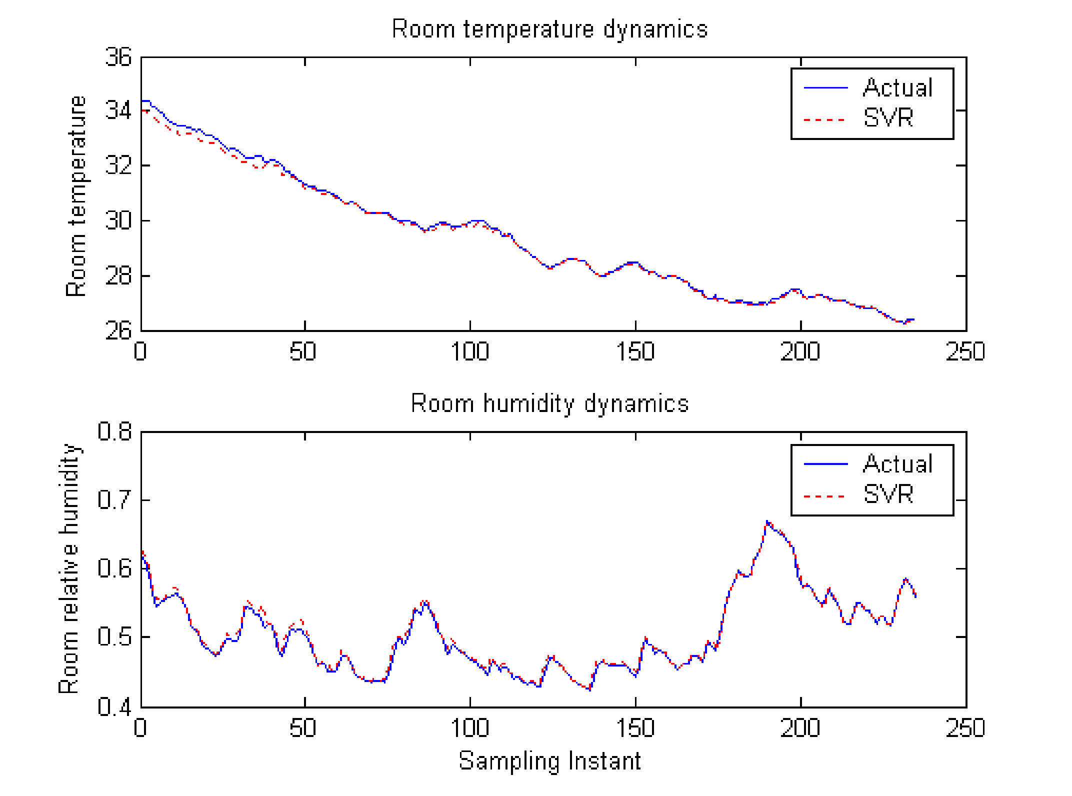

Figure 4.10 Comparison of model and actual data for temperature and RH dynamics

After obtaining the optimal values of C and g, the two models, i.e. temperature and relative humidity dynamics models, were then trained using these optimal hyper-parameter values. These forward dynamic models are used in the nonlinear model predictive control experiment for the HVAC system. To check the generalization property of these models, another data set, which has not been used in training, is used to test the models. A comparison of the model outputs and the actual outputs is shown in Figure 4.10. From these figures, we can see that the model outputs closely follow the actual data. This shows that the SVR forward models have captured the dynamics of the HVAC and have good generalization ability.

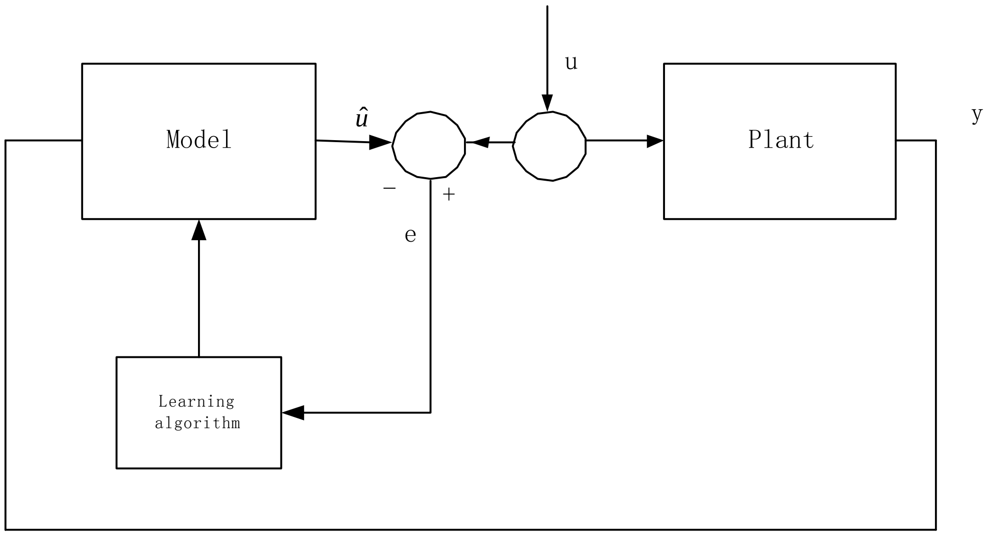

Figure 4.11 inverse modeling

4.4 Obtaining Inverse Dynamic Model of HVAC System

In the SVR inverse control, the SVR directly computes the control signals in order to make the plant output follow the desired setpoint. Given this goal, the learning objective is to model the functionalities between the input and the output of the plant. With the inverse dynamics modeling methodology (Thibault et al., 1991), the training data is obtained in applying input values to the plant in an open-loop structure. These inputs can be randomly generated, but they must preferably cover the entire input domain. The plant input and output are recorded during the experiments. The direct inverse modeling is shown schematically in Figure 4.11. Here, a random control signal u is introduced to the system. The system output is then used as input to the model. The learning algorithm is to obtain a model with minimal complexity that gives the minimal error between the actual control signal u and the model predicted value $\hat{u}$. However, there are some drawbacks to this approach: If the nonlinear system mapping is not one-to-one then the model cannot be trained.

If Eq.(4.4) and (4.5) are invertible, we write the control signals $Fan(k), Valve(k)$ as functions of the other terms:

$$\begin{aligned}Fan(k) = G_1[&RT(k+1), RRH(k+1), RT(k), RRH(k), ST(k),\\ &SRH(k), Valve(k), RT(k-1), RRH(k-1), ST(k-1), SRH(k-1),\\ &Fan(k-1), Valve(k-1), RT(k-2), RRH(k-2), ST(k-2),\\ &SRH(k-2), Fan(k-2), Valve(k-2)]\end{aligned} \tag{4.6}$$

$$\begin{aligned}Valve(k) = G_2[&RT(k+1), RRH(k+1), RT(k), RRH(k), ST(k),\\ &SRH(k), Fan(k), RT(k-1), RRH(k-1), ST(k-1), SRH(k-1),\\ &Fan(k-1), Valve(k-1), RT(k-2), RRH(k-2), ST(k-2),\\ &SRH(k-2), Fan(k-2), Valve(k-2)]\end{aligned} \tag{4.7}$$

Here G_1, G_2 are nonlinear functions in the form of support vector regression.

The same procedure is followed as in the case of forward dynamic model. *k*-fold cross validation will be used to determine the optimal value of *C*, *g* for supply air fan and chilled water valve dynamics models. The optimal values of *C* and *g* are first searched in relatively coarse grids, followed by a search with finer grids. Figure 4.6 is the contour of cross validation squared correlation coefficients for fan dynamics and Figure 4.7 is that for valve dynamics. *C* and *g* were first searched in the range of $C = 2^p, p = -5, -3, \cdots, 15$ and $g = 2^q, q = -15, -13, \cdots, 3$. From these contours of raw searches, a smaller area for a finer search is determined. For fan dynamics, a finer

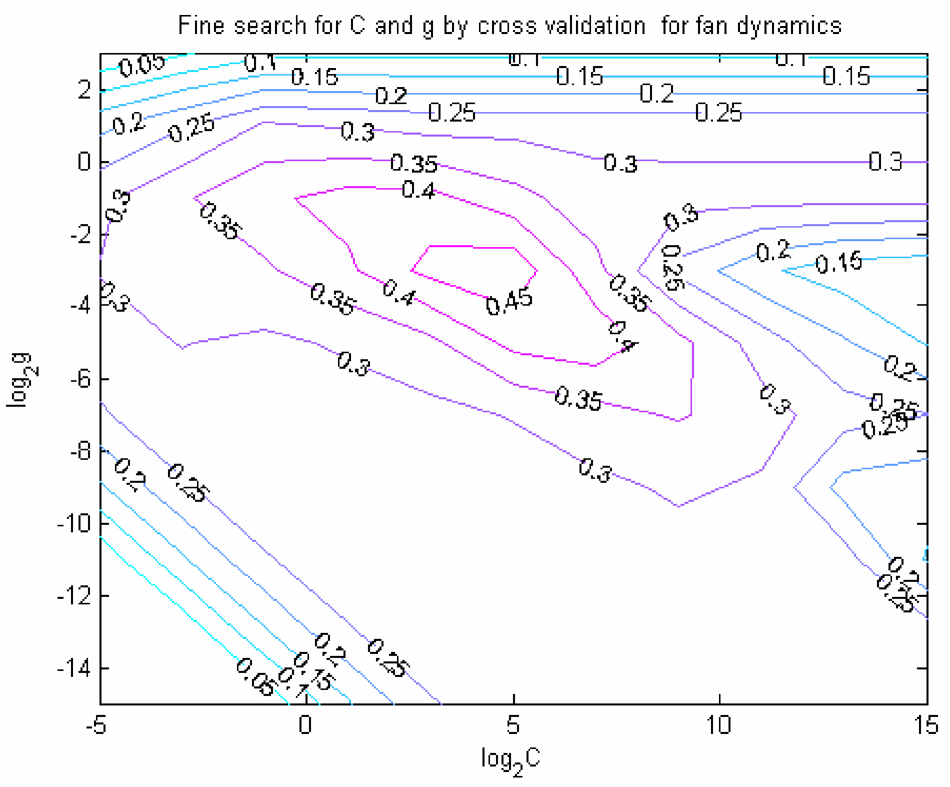

Figure 4.12 Raw search of C and g for fan dynamics

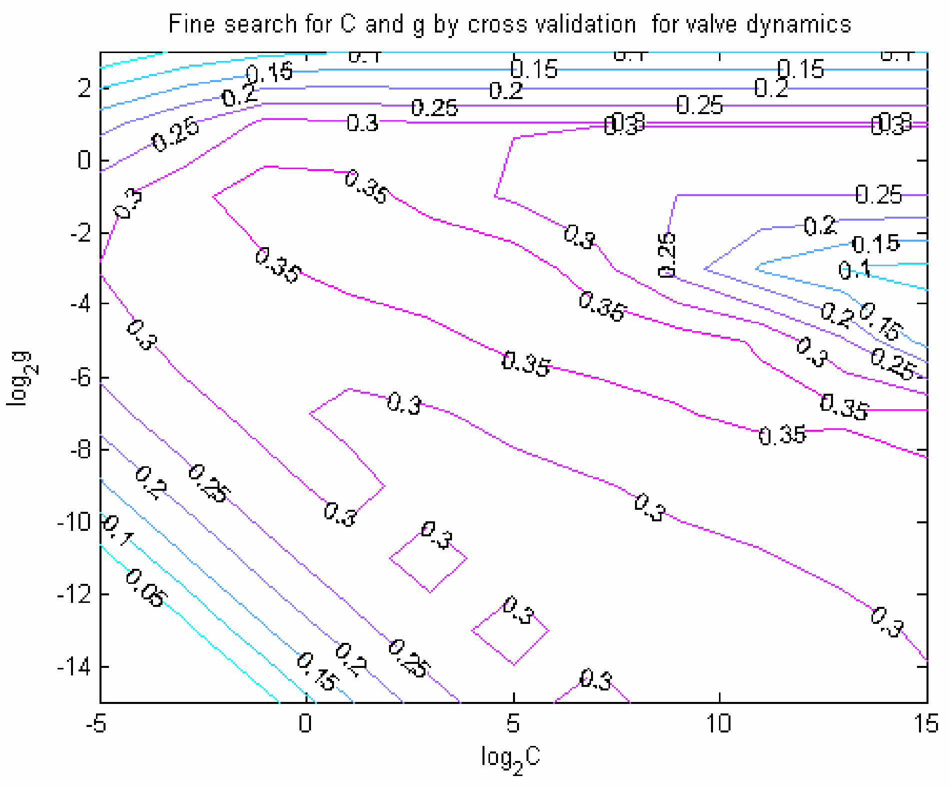

Figure 4.13 Raw search of C and g for valve dynamics

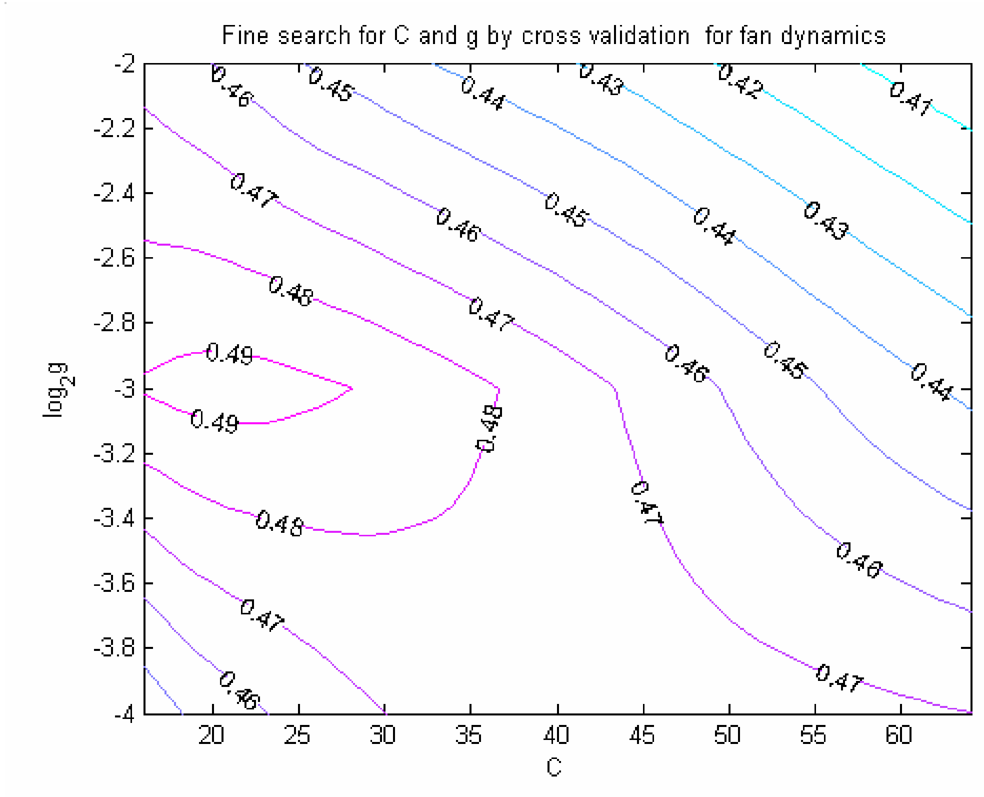

Figure 4.14 Fine search of C and g for fan dynamics

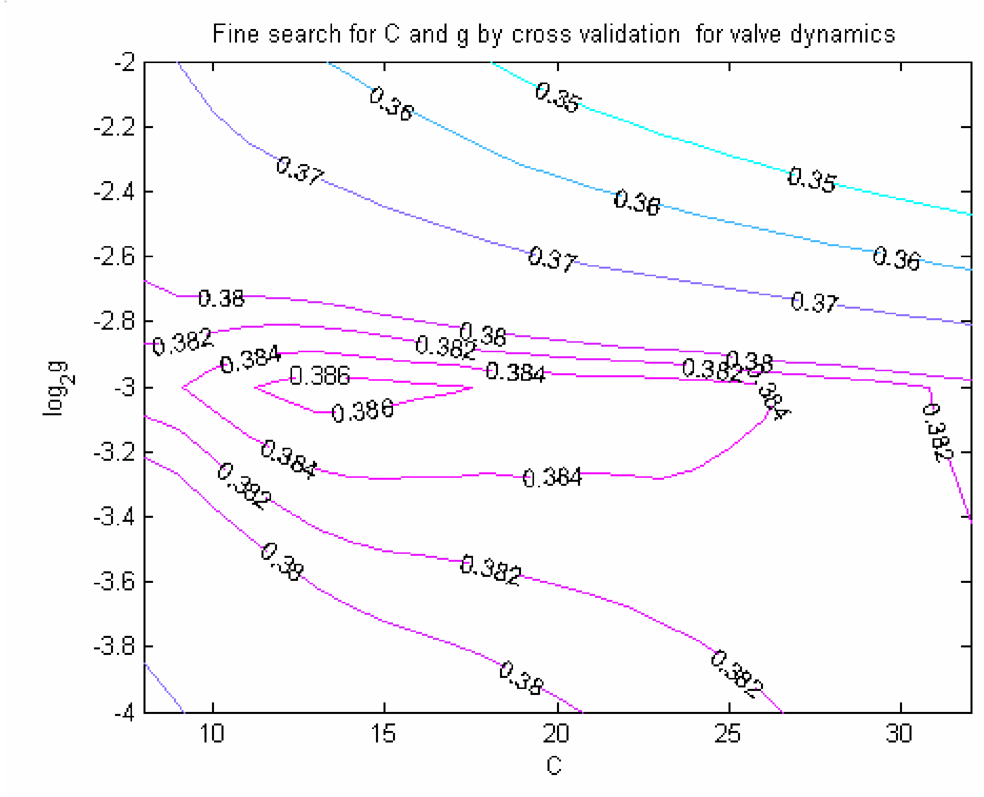

Figure 4.15 Fine search of C and g for valve dynamics

search was performed in the range of $C = 2^p, p = 16,17,\cdots,64$ and $g = 2^q, q = -4,-3,-2$. The highest value found is 0.493994 which is obtained at $C = 21$ and $g = 2^{-3} = 0.125$. For chilled water valve dynamics, a finer search was made in the range of $C = 8,9,\cdots,32$ and $g = 2^q, q = -4,-3,-2$. The best value was found at C=13 and $g = 2^{-3} = 0.125$. The corresponding cross validation squared correlation coefficient is 0.386867.

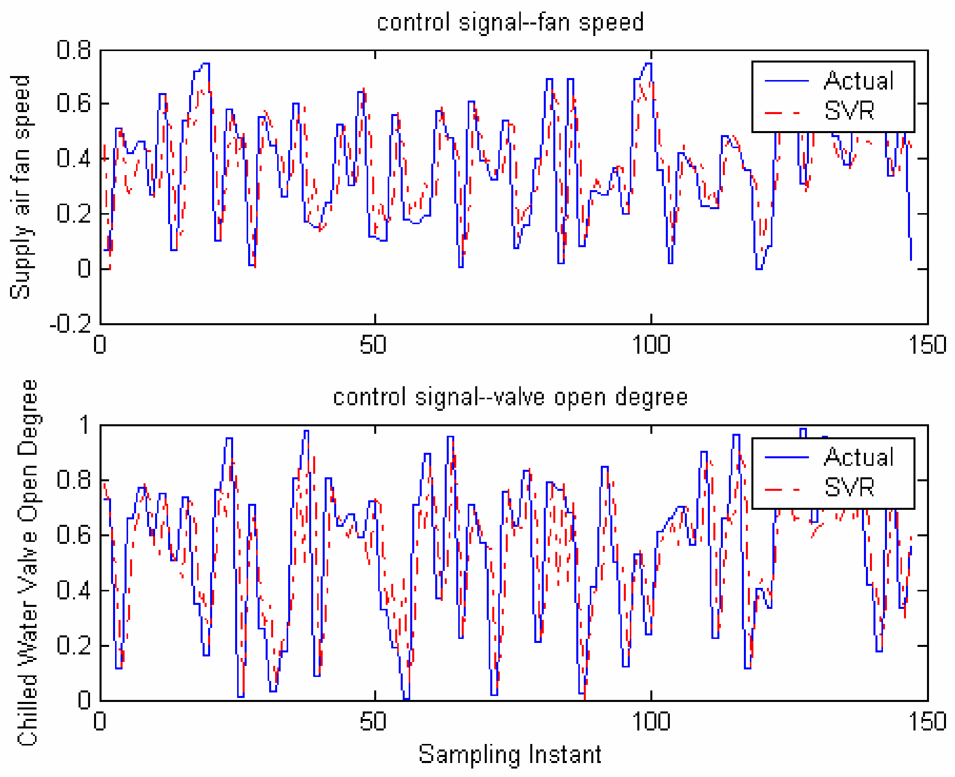

Figure 4.16 Comparison of model and actual data for temperature and RH dynamics

The final models for the inverse dynamics are obtained by training the SVR using these optimal hyper-parameters. A validation data set is used to check the generalization property. A comparison of model outputs and actual values is shown in Figure 4.16.

Compared with the forward dynamic models, the generalization property of the inverse dynamic models is not so good. The SVR model predicted values are almost in the same trend with the actual data, but there still exist small differences in their magnitudes. This also reflected in the lower value of the cross validation squared correlation coefficients.

Chapter 5 Inverse Control Using SVR Model

In the previous chapter, SVR was used to model the inverse dynamics of the HVAC system. In this chapter, a SVR inverse controller is designed based on these inverse models. An appropriate reference model is the key to controller design. The nonlinear equations are solved by Newton methods. This control strategy is tested on the HVAC system.

5.1 Introduction to Inverse Control

In inverse control, the inverse model is simply cascaded with the controlled system in order that the whole system results in identity mapping between desired response and the controlled system output. Thus, the SVR inverse model acts directly as the controller in such a configuration. Clearly, this approach relies heavily on the fidelity of the inverse model used as the controller. For general purpose use, serious questions arise regarding the robustness of direct inverse control (Hunt *et al.* 1992). This lack of robustness can be attributed primarily to the absence of feedback. The problem can be overcome to some extent by using online learning with which the parameters of the inverse model can be adjusted online.

5.2 Design of SVR Inverse Controller

The inverse dynamics of the HVAC system had been obtained in the form of (4.6) and (4.7) in the previous chapter. In this chapter, Equations (4.6) and (4.7) are used as the inverse control laws. We replace the outputs, $RT(k+1)$ and $RRH(k+1)$, at the $(k+1)$ instant with the desired values, $RT_{ref}(k+1)$ and $RRH_{ref}(k+1)$. With these replacements, functions G_1 and G_2 can be rewritten in the following forms:

$$\begin{aligned}
Fan(k) = G_1[&RT_{ref}(k+1), RRH_{ref}(k+1), RT(k), RRH(k), ST(k),\\
&SRH(k), Valve(k), RT(k-1), RRH(k-1), ST(k-1), SRH(k-1),\\
&Fan(k-1), Valve(k-1), RT(k-2), RRH(k-2), ST(k-2),\\
&SRH(k-2), Fan(k-2), Valve(k-2)]
\end{aligned} \tag{5.1}$$

$$\begin{aligned}
Valve(k) = G_2[&RT_{ref}(k+1), RRH_{ref}(k+1), RT(k), RRH(k), ST(k),\\
&SRH(k), Fan(k), RT(k-1), RRH(k-1), ST(k-1), SRH(k-1),\\
&Fan(k-1), Valve(k-1), RT(k-2), RRH(k-2), ST(k-2),\\
&SRH(k-2), Fan(k-2), Valve(k-2)]
\end{aligned} \tag{5.2}$$

After obtaining the two inverse models, a controller can be designed based on these models. The new control signals are computed at every sampling instant. Due to the slow dynamics of the HVAC plant, the room temperature and the room relative humidity cannot change by too large a magnitude within one sampling interval. It is therefore important to design an appropriate reference trajectory so that the room temperature and the room relative humidity can reach their set points in a smoother manner with the physical limitation of the plant. The reference trajectories, based on first-order models, are calculated as shown by the quasi-code in Table 5.1 :

Table 5.1 Reference model for inverse control

```
If $\left|T_{ref}(k+1) - RT(k)\right| > \Delta T$
    If $RT(k) > T_{ref}(k+1)$
        $T_{ref}(k+1) = RT(k) - \Delta T$
    else
        $T_{ref}(k+1) = RT(k) + \Delta T$
If $\left|T_{ref}(k+1) - RT(k)\right| \le \Delta T$
$T_{ref}(k+1) = RT(k)$

If $\left|RH_{ref}(k+1) - RRH(k)\right| > \Delta RH$
    If $RRH(k) > RH_{ref}(k+1)$
        $RH_{ref}(k+1) = RRH(k) - \Delta RH$
    else
        $RH_{ref}(k+1) = RRH(k) + \Delta RH$
If $\left|RH_{ref}(k+1) - RRH(k)\right| \le \Delta RH$
$RH_{ref}(k+1) = RRH(k)$
```

By setting $\Delta T = 0.5, \Delta RH = 0.05$, and following the above procedures, we can get the control signals $Fan(k), Valve(k)$.

The Equations (5.1) and (5.2) are a set of simultaneous nonlinear equations for which the solutions for $Fan(k)$ and $Valve(k)$ are needed. Note that there two parameters also appear on the right hand side of the equations. We use the Newton methods to solve this problem (Yakowitz and Szidarovszky, 1989).

This is shown as follows. At any sampling instant, k, only $Fan(k), Valve(k)$ are unknown, other parameters are all known. Replacing $Fan(k)$ and $Valve(k)$ by x_1 and x_2, respectively. (5.1) and (5.2) can be rewritten as follows:

$$\begin{cases} x_1 = g_1(x_2) \\ x_2 = g_2(x_1) \end{cases} \tag{5.3}$$

where $g_1(x_2)$ and $g_2(x_1)$ are SVR functions of x_2 and x_1, respectively. Define:

$$\begin{cases} f_1(x_1, x_2) = x_1 - g_1(x_2) = 0 \\ f_2(x_1, x_2) = x_2 - g_2(x_1) = 0 \end{cases} \tag{5.4}$$

By Taylor's expansion about the point $x_1 = x_1^{(k)}$ and $x_2 = x_2^{(k)}$, we have

$$\begin{aligned} \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} &= \begin{bmatrix} f_1(x_1^{(k)}, x_2^{(k)}) \\ f_2(x_1^{(k)}, x_2^{(k)}) \end{bmatrix} + \begin{bmatrix} \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & \dfrac{\partial f_1(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \\ \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_1} & \dfrac{\partial f_2(x_1^{(k)}, x_2^{(k)})}{\partial x_2} \end{bmatrix} \begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \tag{5.5}$$

or

$$\begin{bmatrix} f_1(x_1^{(k)},x_2^{(k)}) \\ f_2(x_1^{(k)},x_2^{(k)}) \end{bmatrix} + \begin{bmatrix} 1 & -\dfrac{\partial g_1(x_1^{(k)},x_2^{(k)})}{\partial x_2} \\ -\dfrac{\partial g_2(x_1^{(k)},x_2^{(k)})}{\partial x_1} & 1 \end{bmatrix} \begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.6}$$

As long as

$$\begin{vmatrix} 1 & -\dfrac{\partial g_1(x_1^{(k)},x_2^{(k)})}{\partial x_2} \\ -\dfrac{\partial g_2(x_1^{(k)},x_2^{(k)})}{\partial x_1} & 1 \end{vmatrix} \neq 0 \tag{5.7}$$

We can obtain a unique solution of (5.5). A better solution is then given by

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} + \Delta x_1^{(k)} \\ x_2^{(k)} + \Delta x_2^{(k)} \end{bmatrix} \tag{5.8}$$

The process is repeated until the desired accuracy is reached. In our case, the process is stopped when both $\left|\Delta x_1^{(k)}\right|$ and $\left|\Delta x_2^{(k)}\right| < 10^{-6}$.

In Eq. (5.3), the SVR functions, $g_1(x_2)$ and $g_2(x_1)$, are in the form of:

$$g_1(x_2) = \sum_{i=1}^{m_1} \alpha_{1i} \exp\left(-\frac{\left\|v_1 - v_{1i}\right\|^2}{2\sigma^2}\right) = \sum_{i=1}^{m_1} \alpha_{1i} \exp\left(-\frac{(x_2 - x_{2i})^2 + A_1}{2\sigma^2}\right) \tag{5.9}$$

$$g_2(x_1)=\sum_{i=1}^{m_2}\alpha_{2i}\exp\left(-\frac{\|v_2-v_{2i}\|^2}{2\sigma^2}\right)=\sum_{i=1}^{m_1}\alpha_{2i}\exp\left(-\frac{(x_1-x_{1i})^2+A_2}{2\sigma^2}\right) \tag{5.10}$$

where v_1 is the input vector to the SVR function G_1, whose elements are listed in the parenthesis of Eq. (5.1). $v_{1i}, i=1,\ldots,m_1$ are the support vectors in G_1. x_2, i.e. $valve(k)$, is the 7th element of v_1. $x_{2i}, i=1,\ldots,m_1$ are the 7th elements in support vector $v_{1i}, i=1,\ldots,m_1$. Since other elements except x_2 are all known, $\|v_1-v_{1i}\|^2$ can be expressed as a summation of $(x_2-x_{2i})^2$ and A_1. A_1 is a constant with respect to x_2. Similarly, v_2 is the input vector to the SVR function G_2, whose elements are listed in the parenthesis of Eq. (5.2). $v_{2i}, i=1,\ldots,m_2$ are the support vectors in G_2. x_1, i.e. $fan(k)$, is the 7th element of v_2. $x_{1i}, i=1,\ldots,m_2$ are the 7th elements in support vector $v_{2i}, i=1,\ldots,m_2$. A_2 is a constant with respect to x_1.

The partial derivative terms required in Eq. (5.5) are given by

$$\begin{aligned}\frac{\partial g_1(x_2)}{\partial x_2}&=\sum_{i=1}^{m_1}\alpha_{1i}\exp\left(-\frac{\|v_1-v_{1i}\|^2}{2\sigma^2}\right)\left(-\frac{x_2-x_{2i}}{\sigma^2}\right)\\&=\sum_{i=1}^{m_1}\alpha_{1i}\exp\left(-\frac{(x_2-x_{2i})^2+A_1}{2\sigma^2}\right)\left(-\frac{x_2-x_{2i}}{\sigma^2}\right)\end{aligned} \tag{5.11}$$

$$\begin{aligned}\frac{\partial g_2(x_1)}{\partial x_1}&=\sum_{i=1}^{m_2}\alpha_{2i}\exp\left(-\frac{\|v_2-v_{2i}\|^2}{2\sigma^2}\right)\left(-\frac{x_1-x_{1i}}{\sigma^2}\right)\\&=\sum_{i=1}^{m_1}\alpha_{2i}\exp\left(-\frac{(x_1-x_{1i})^2+A_2}{2\sigma^2}\right)\left(-\frac{x_1-x_{12i}}{\sigma^2}\right)\end{aligned} \tag{5.12}$$

The solution process starts with some appropriate initial value $\begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix}$, and iterates to the final solution, i.e. $Fan(k), Valve(k)$. This computed values of the control signals are the input to the plant. The same procedure is repeated at each sampling instant.

5.3 Experimental Results

Only after the startup of the system, will the chiller begin to cool down the chilled water. During the process of the experiment, the chiller will turn on or off according to the temperature of the chilled water leaving the chiller. The plant controller itself can not control the switch of the chiller. As such there will be fluctuations in the chilled water temperature. Besides the two manipulated variables, i.e. supply air fan speed and chilled water valve opening, that affect the temperature and relative humidity of the supply air, a third parameter, the chilled water temperature, also affects the state of the supply air. In the experiment, the chilled water temperature is considered as a measurable disturbance.

The experimental results are shown in Figure 5.1 and Figure 5.2. The room temperature and relative humidity were initially set to 24^oC and 65% respectively. From the results shown in Figure 5.1, we note that the room temperature and the room relative humidity can reach the set points in a very smooth manner, with very small overshoots. After reaching the set points, the controller can keep this steady state with a temperature error of less than 0.2^0C and relative humidity error of less than 1%. The changes of the control signals, i.e. supply air fan speed and chilled water valve opening, is relatively drastic during the startup, then the control signals decrease gradually to a small value in the steady state. The small fluctuations in the control signals are meant to offset the fluctuations in the chilled water. This is the demonstration that SVR inverse controller has the ability of disturbances rejection.

The reference settings were changed to $23\,^0C$ and 60% at the 390th sampling instant. From Figure 5.2, the control signals increase to react to the change of setting. After about 100 sampling instants (16.7 minutes), the room temperature and relative humidity have settled to the new settings, with small steady-state errors. The third setting of $22\,^0C$ and 65%, made at the 570th sampling instant, can also been reached at around the 730th sampling instant. From these responses, it is noted that there is negligible overshoots for the temperature while the relative humidity has some small overshooting before the temperature has reached its setpoint. This is due to the fact that the dynamics of relative humidity is faster than that of temperature.

5.4 Conclusion of SVR Inverse Control

The feasibility of the SVR inverse control strategy has been investigated. The design of SVR inverse controller consists of two steps: the first is building up the inverse SVR model of the system, and the second is the design of the SVR inverse controller. The strategy has been shown to be successful experimentally. The controller has the ability of set point tracking and disturbance rejection. The controller can work effectively in the start up period which is difficult to be described by a linear model around some operating point. The good performance of the controller is based on the availability of training data within the plant operating range. The control strategy can be used to produce the

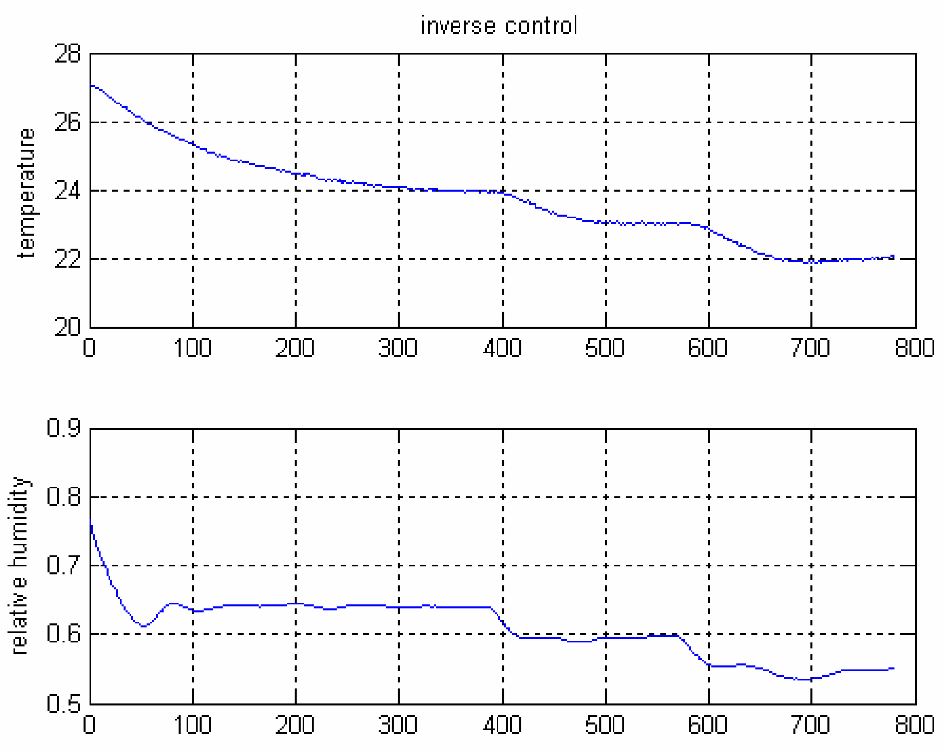

Figure 5.1 Changes of room temperature and relative humidity using SVR controller

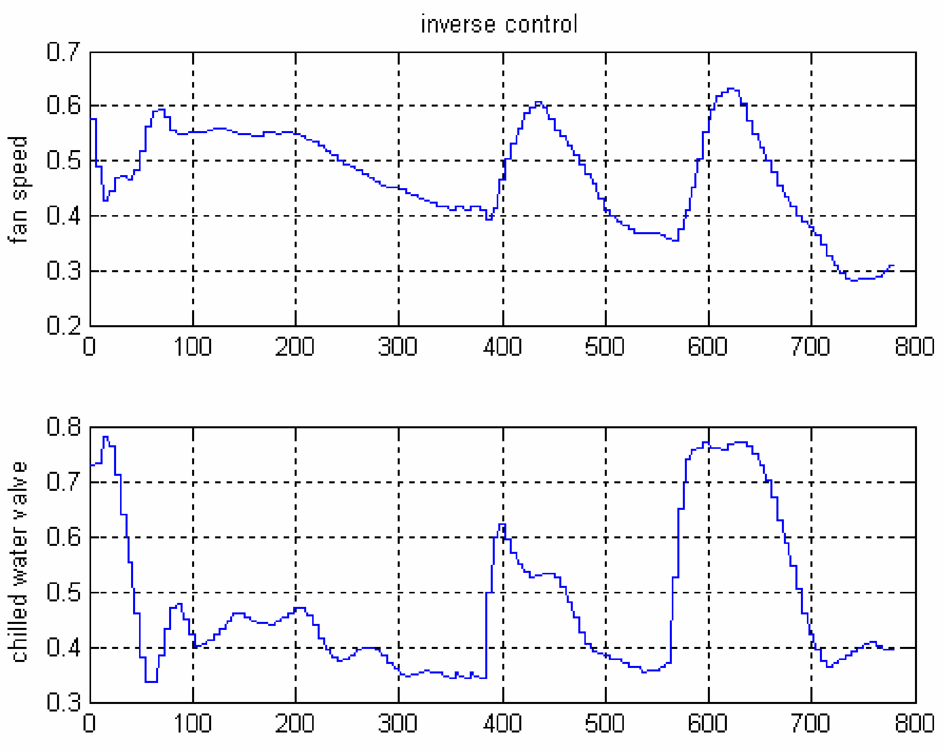

Figure 5.2 Changes of supply air fan speed and chilled water valve opening

environmental space where accurate temperature and relative humidity control is required.

However, the disadvantage of the SVR inverse control is that the response time is quite long (roughly two times that of MPC control which will be discussed in the next chapter). Here, the cooling capacity is not exploited to its full potential, which is demonstrated by the fact that the fan never runs at its maximum speed and the valve is never fully open. Although the controlled system has good reference tracking ability and small steady state errors, the slow response makes it less desirable to be used in practical situations. Also, few choices are available to tune the performance of the controller. So in this project, no further efforts were put into improving the design of the SVR inverse controller. Instead, research is focused on the MPC using SVR forward models, which will be discussed in more details in next chapter.

Chapter 6 Model Predictive Control Using SVR Models

Model predictive control has been increasingly used in industry because of its attractive properties. The model itself has significant impact on model predictive control. The excellent generalization property of SVR forward models motivates their use in model predictive control. In this chapter, we will discuss the model predictive control using SVR forward models.

6.1 Introduction of Model Predictive Control

The past two decades have witnessed great success in the use of model predictive control (MPC) in a variety of industrial processes (Camacho and Bordons, 1999). MPC is sometimes referred to as receding horizon control (RHC) in the literature. MPC is a form of control in which the current control action is obtained by solving on-line a finite horizon open-loop optimal control problem at each sampling instant, using the current state of the plant as the initial state (Mayne *et al*., 2000). Only the first action of the optimal control consequence, which is computed as a result of the optimization, is actually applied to the plant. This procedure is repeated at every sampling instant using the updated information (measurements) of the process. A key advantage of MPC over other control schemes is its ability to deal with constraints in a systematic and straightforward manner. Within the framework of constrained optimization, the constraints that can be handled include not only saturation limits but also other performance and safety constraints on inputs, outputs and state variables of the process.

Different norms can be used in the objective function that leads to different optimization problems. The most popular choice is the 2-norm type objective function which contains quadratic terms in outputs errors and control efforts. In general, the 2-norm objective function leads to a quadratic programming (QP) type optimization problems the solution of which gives the control law.

6.1.1 MPC Strategy

The basic ideas of MPC is characterized by the following strategy (Camacho and Bordons, 1999), represented in Figure 6.1.

The future outputs for a pre-determined horizon P, called the prediction horizon, are predicted at each instant k using the process model, The predicted outputs $\hat{y}(t+k\mid t)$ for $k=1,\dots,N$ depend on the known values up to instant t (past inputs and outputs) and on the future control signals $u(t+k\mid t), k=0,\dots,P-1$.

The set of future control signals is computed by optimizing a specified criterion so as to keep the process as close as possible to the reference trajectory $r(t+k)$ subject to certain constraints. This criterion usually takes the form of a quadratic function of the errors between the predicted output signal and the predicted reference trajectory. Control effort, rate of change of control signals and terminal cost are usually included in the objective function.

The first computed optimized control $u(t\mid t)$ is sent to the process whilst the subsequent control signals that were computed are discarded. At the next sampling instant, $y(t+1)$

will become known and step 1 repeated with this new value and all the sequences are brought up to date. Thus a new $u(t+1|t+1)$ is calculated (which in principle will be different from the $u(t+1|t)$ because of the additional new information available) using the receding horizon concept.

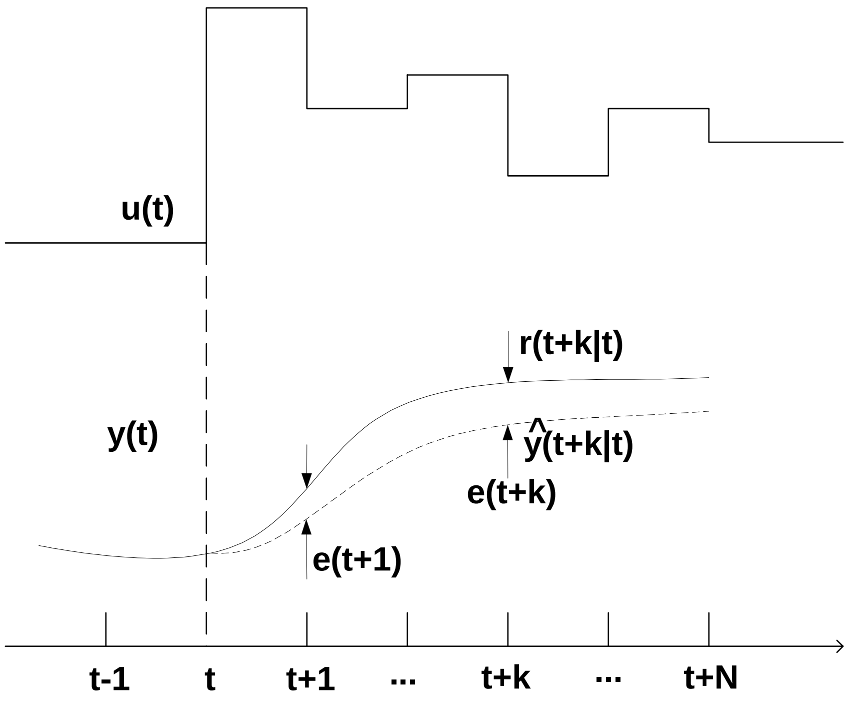

Figure 6.1 MPC strategy

6.1.2 Nonlinear Models

Although industrial processes usually contain complex nonlinearities, most of the MPC algorithms are based on a linear model of the process. Linear model such as step response and impulse response models are preferred, because they can be identified in a straightforward manner from process data. In addition, the goal for most of the applications is to maintain the system at a desired steady state, rather than moving rapidly between different operating points. A precisely identified linear model is therefore sufficiently accurate in the neighborhood of a single operating point. But if the process is highly nonlinear and subject to large frequent disturbances, a non-linear model will be necessary to describe the behavior of the process. Also in servo control problems where the operating points are changing, a nonlinear model of the plant is necessary.

Because of its universal approximation ability, neural networks (NN) have been used in model predictive control. Potocnik and Grabec (2002) proposed a nonlinear MPC which combines a neural network model and a genetic algorithm based optimizer in a simulated chaotic cutting process. Duarte *et al.* (2001) investigated a direct neural network multivariable predictive controller applied to a grinding plant. Gu and Hu (2002) presented a path tracking scheme for a car-like mobile robot based on neural predictive control. Possessing the similar universal approximation ability, SVR can also be used to model nonlinear processes, just as neural networks are. Support vector regression has been reported to be used in the model predictive control area. Miao and Wang (2002) studied the MPC using SVR model for a SISO system.

6.2 Problem Formulation

Nonlinear models are used to predict the effect of sequences of control actions on the controlled variables. The aim is to derive an optimal set of control sequences, which will drive the outputs to the desired state setpoints, based on optimizing some specified performance index. For the HVAC system under control, the performance index is devised as follows:

$$\min_{u}\sum_{k=0}^{P-1}\left\{\sum_{i=1}^{2}\left[\gamma_i\left(\frac{y_i(k)-r_i}{\bar{y}_i-r_i}\right)^2+\theta_i u_i^2(k)+\phi_i\Delta u_i^2(k)\right]\right\}+\sum_{i=1}^{2}\eta_i\left(\frac{y_i(P)-r_i)}{\bar{y}_i-r_i}\right)^2 \qquad (6.1)$$

$$y_i(k+1)=\sum_{j=1}^{m_i}\alpha_{ij}\exp\left(-g\left\|v(k)-v_{ij}\right\|\right)+b_i, i=1{,}2$$

$$\text{s.t. } 0.1\le u_1\le 0.75 \qquad (6.2)$$

$$0.1\le u_2\le 1$$

In the above equation, y_1 and y_2 are the room temperature and the room relative humidity respectively, $\bar{y}_1$ and $\bar{y}_2$ their respective initial values that is at the start of the sampling period, r_1 and r_2 their respective reference values, u_1 and u_2 are the supply air fan speed and the chilled water valve opening respectively, and $\gamma_i, \theta_i, \varphi_i, \eta_i$ are the tuning coefficients.

The variables $\bar{y}_i$ are such that:

$$\begin{aligned}\left|\bar{y}_i-r_i\right| &= \left|y_i(0)-r_i\right| && \text{for } \left|y_i(0)-r_i\right|\ge\varepsilon_i>0 \\ &= \varepsilon_{\mathrm{i}} && \text{otherwise}\end{aligned} \qquad (6.3)$$

The above condition is introduced to avoid the denominators, $\left|\bar{y}_i-r_i\right|$, becoming too small as the controlled system reaches steady-state values. The parameters ε_i are chosen based on the relative importance, or precision, of parameters under control. In this project, ε_1 is chosen as 1 ^0C and ε_2 is chosen as 5.5%. At each sampling instant, the initial conditions, $\bar{y}_1$ and $\bar{y}_2$, are updated by Eq. (6.3). This method is referred to as the strategy of updating conditions in later chapters.

From the formulation of the problem, it is clear that MPC involves nonlinear programming which is to minimize a performance index subject to system dynamics and control constraints. The performance index contains the cost incurred from 0 to *P-1* stage and the terminal cost at *P* stage. From 0 to *P*-1 stage, the three items are 1) squared errors between the current outputs and the reference values, 2) squared magnitude of the control signals and 3) squared change rates of the control signals. To make the different items in the performance index comparable, it is important to make different items dimensionless. $\gamma_i, \theta_i, \varphi_i, \eta_i$ are the coefficients that allow the relative importance of different items to be adjusted. In the experiments, it is found that these coefficients have significant influence on the performance of the controller. The details of the effects of the coefficients will be discussed in later sections.

Although in the identification phase, the lower bounds for both controls, u_1 and u_2 are 0, in the control experiments these lower bounds were all modified to 0.1 empirically to prevent the control process from getting stuck at certain points. This is attributed to the fact that the sensors of the supply air temperature and relative humidity are installed in the ducting after the air-handling unit. If the supply fan almost stops and the chilled water valve is almost closed, then very little heat exchange will happen between the air and the chilled water and the supply air will remain to be at its original state.

At each sampling instant, iterative dynamic programming (IDP) was used to compute the optimal control sequences and the first control action is actually implemented. IDP will be discussed in details in the next section.

6.3 Iterative Dynamic Programming

6.3.1 Brief Introduction to Iterative Dynamic Programming

The initial ideas on iterative dynamic programming (IDP) were developed and tested by Luus (1990), who gave a comprehensive coverage in a monograph (Luus, 2000). IDP has attracted the attention of researchers due to its many very favorable properties. It is easy to implement, is quite robust, does not involve solution of a nonlinear programming (NLP) problem even in the case of input constraints, is reported to be capable of finding a global optimum and does not require any differentiation of process equations that is sometimes very difficult (Rusnák *et al.*, 2001). The original IDP method was developed for systems described by a set of differential equations. Its detailed theoretical analysis can be found in Bojkov and Luus (1993) and Dadebo and Mcauley (1995). Rusnák *et al.* (2001) proposed a modified version of IDP which uses discrete input-output models rather than differential equations.

As its names suggests, IDP is based on Bellman’s principle of optimality. Its basic principle is to optimize P single control stages in turn by starting at the last stage instead of optimizing all P stages simultaneously. Thus, whenever the final i stages of the optimal control have been established, the preceding stage may be obtained by simply considering one new stage and then continuing with the already established control policy in the remaining stages.

6.3.2 IDP Problem Formulation for Discrete Time Models

The IDP algorithm in this project is modified from the algorithm proposed by Rusnák *et al.* (2001). While the latter one is implemented in one pass, the former is done in a multi-pass fashion. Consider the discrete system:

$$y(t) = f(y(t-1), y(t-2), \ldots, u(t-1), u(t-2), \ldots) \tag{6.4}$$

where t is the current instant and suppose that the input $u(t+j)$ has to be within limits:

$$u_j^{\min} \le u(t+j) \le u_j^{\max}, \, j = 1, 2, \cdots, N_u \tag{6.5}$$

and N_u is the number of inputs. The associated performance index to be optimized is:

$$J = F(\hat{y}(t+1), \ldots, \hat{y}(t+P), u(t), u(t+1), \ldots, u(t+P-1)) \tag{6.6}$$

$\hat{y}(t+1), \ldots, \hat{y}(t+P)$ denote the model predicted values. The optimal control problem is to find the piece-wise constant control policy $u(t+j), j = 0, \ldots, P-1$ such that the performance index given is minimized.

6.3.3 IDP Algorithm

Before discussing an approach to solving the IDP problem, the following parameters are defined:

P, number of stages (or equivalently prediction horizon);

M, number of randomly chosen control candidates;

N number of y-grid points;

r^0, initial size of control region;

φ_1, contraction factor after each iteration, $\varphi_1 \in [0.7, 0.9]$;

φ_2 , restoration factor after each pass which consists of several iterations, $\varphi_2 \in [0.7, 0.9]$;

N_i, number of iterations in each pass;

N_p, number of passes.

Some detailed descriptions that are helpful to understand the implementation of IDP are given in Appendix. The proposed algorithm can be concisely described by the following steps:

1) Choose an initial control trajectory and let initialize the iteration counters by setting up $i = 1$ and $p = 1$.

2) Set the control region by

$$r^i = r^0 \varphi_2^{p-1} \tag{6.7}$$

where p is the pass index.

3) Choose N control trajectories by perturbing the optimal (or initial if i=1 and p =1) control trajectory $u^{i-1} = \left[u^{i-1}(t), \ldots, u^{i-1}(t + P - 1)\right]$ uniformly inside the

admissible region that is given as the intersection of the control region r^i and the constraints Eq. (6.5).

4) Use the N control trajectories to obtain N model output trajectories for the interval within the prediction horizon P and store the system output trajectories.

5) Start at the last stage P and generate for each grid point $\hat{y}(t+P-1,n)$ ($n=1,\dots,N$) M admissible values for control by

$$u_j^i(t+P-1)=u_j^{i-1}(t+P-1)+d_j r_j^i,\ j=1,\dots,N_u \tag{6.8}$$

where i is the iteration index, and d_j is a random number with the range of [-1, 1]. $u_j^{i-1}(t+P-1)$ is the jth element of $u^{i-1}(t+P-1)$, which is the best value vector of the control action for the particular $\hat{y}(t+P-1,n)$ grid point obtained in the previous iteration. Eq. (6.8) is used M times for each grid point $\hat{y}(t+P-1,n)$ to generate M allowable control $u^i(t+P-1)$. Also, $u^i(t+P-1)$ must satisfy the control constraints Eq. (6.5). If the limits are exceeded, its value are truncated to its nearest bounds. Apply these control actions to the process model and obtain the set of corresponding output predictions $\hat{y}(t+P,n,M)$ at stage $P+1$. Compute M values of the cost function at stage $P+1$ and choose the control action $u^i(t+P-1,n)$ that minimizes it. Store the best control actions $u^i(t+P-1,n)$ for the next step This comparison of M values of J is repeated for N times for each grid point $\hat{y}(t+P-1,n)$, $n=1,\dots,N$.

6) Step back to stage P-1. For each y-grid $\hat{y}(t+P-2,n)$, $n=1,\dots,N$ calculate the predictions by Eq. (6.4) from stage P-1 to stage P once with each of the M allowable values for control. To continue the prediction from stage P to stage

P+1, choose the optimal control obtained from step 4) that corresponds to the closest grid point at stage *P*. For each y-grid point at stage *P*-1, compare with the *M* values of *J* and store the control $u^i(t+P-2,n)$ that gives the minimum value. This comparison is also repeated for *N* y-grid points $\hat{y}(t+P-2,n)$, $n=1,\ldots,N$.

7) Repeat the previous step until the initial time (stage 1) is reached. At stage 1, there is only one y-grid point $y(t,1)$. Among *N* different control trajectories choose the one that minimizes the performance index.

8) Reduce the region for the admissible control values by an amount φ_1

$$r^{i+1} = \varphi_1 r^i \tag{6.9}$$

and increase the iteration index, $i = i+1$. If the maximum number of iterations in each pass in not reached, i.e. $i \le N_i$, then go step (3).

9) Increase the pass index, $p = p+1$. If the maximum number of passes is not reached, i.e. $p \le N_p$ then repeat from step (2).

6.3.4 Online Implementation of IDP

Due to the online nature of MPC, the speed of computation for the optimization of MPC is very important. From the above discussion, it is clear that the tuning parameters will determine the online computational load. These parameters are the length of prediction horizon *P*, the number of randomly chosen control candidates *M*, and the number of y-grid points *N* . Fortunately, Luus (2000) made a fairly comprehensive discussion for the choices of these parameters.

In the early years of IDP, control candidates were chosen uniformly inside the given region. For each control variable, we therefore have a minimum of 3 values, namely –r, 0, and r distance from the optimum value obtained at the previous iteration, where r is the region size. This method is easy to be implemented. However, with an increase in the number of control variables, the computational load will increase very fast. So it is recommended to randomly choose control variables inside the region. With this random choice strategy, it was reported that IDP has been successfully used in a system with 130 differential equations and 130 control variables (Luus, 1993).

The number of y-grid points will also affect the computational load. It has been reported that the use of more than 3 y-grid points at each stage will not increase the possibility of obtaining the optimal value (Luus, 2000). Without sacrificing convergence rate, we will use the minimum number of one, whenever possible. In this project, the HVAC system is a 2×2 system. During the simulations and the experiments, one y-grid point at each stage was found to be good enough.

The computational parameters used for this online MPC are listed as follows:

Number of randomly chosen control candidates M is 5;

Number of y-grid points N is 1;

Number of passes N_p is 2;

Number of iterations per pass N_i is 10;

Region contraction factor after each iteration in a pass φ_1 is 0.75;

Region restoration factor after each pass φ_2 is 0.80.

6.4 Experimental Results

Generally speaking, the performance of the SVR MPC controller is much better than that of the SVR inverse controller. A typical MPC control experiment is shown in Figure 6.2 and Figure 6.3. The experiment begins with the initial condition of $RT = 27.3^0 C$ and $RRH = 83\%$. The initial setpoint is $RT = 24^0 C, RRH = 65\%$. The tuning parameters used are listed in Table 6.1.

Table 6.1 Typical values in performance index

P	γ_1	γ_2	θ_1	θ_2	ϕ_1	ϕ_2	η_1	η_2
1	2	1	0	0	$\frac{0.6}{0.75^2}$	0.6	4	4

From Figure 6.2, we can see that both the room temperature and the room relative humidity slowly reach their first setpoint accurately at around the 150th sampling instant, which is 25 minutes. It is noted that the dynamics of the room relative humidity is much faster than that of the room temperature. There are some oscillations of the room relative humidity before the room temperature reaches its setpoint. Another noticed phenomenon is that the chilled water valve demonstrates violent oscillations in the transition period. Because of the faster dynamics of the room relative humidity compared with that of the room temperature, the relative humidity reaches its setpoint earlier than the temperature. If the chilled water valve keeps a large opening degree while the supply air fan run at high speed, the room relative humidity will further go down from its setpoint. Thus the amplitude of the error between the room relative humidity and its reference will increase. By using the strategy of updating initial conditions, the items of the error of the room relative humidity still have significant relative importance in the performance index, which prevents the room relative humidity from deviating from its setpoint. If the error of the room relative humidity exceeds a certain value, the chiller water valve opening must be decreased to prevent the relative humidity from further straying away from its setpoint. Thus, keeping the relative importance of the error of the relative humidity in the performance index causes the oscillations of the chilled water valve opening in the transition period.

It is found that the choices of the tuning parameters in the performance index have significant impact on the performance of the controller. In the next few subsections, the effects of some of these tuning parameters will be discussed in details.

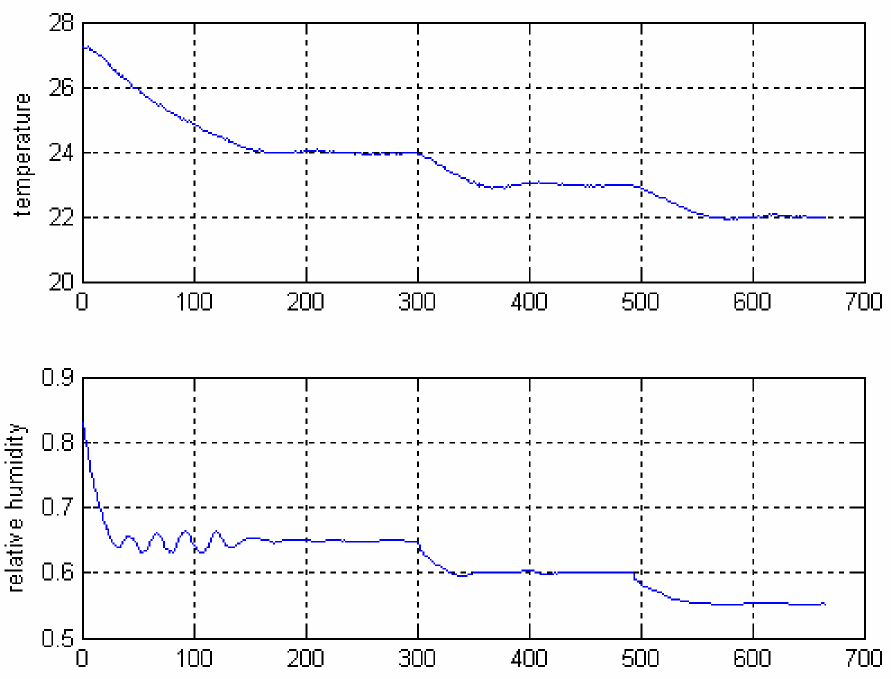

Figure 6.2 Typical MPC control for room temperature and RH

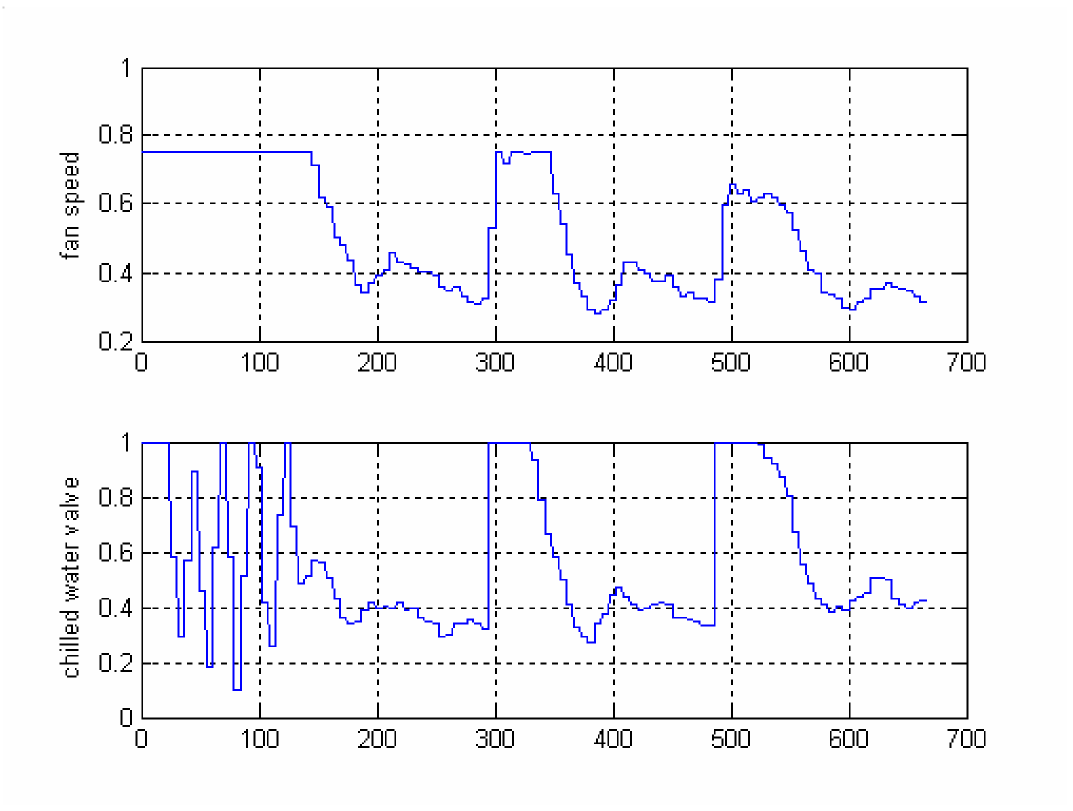

Figure 6.3 Typical MPC control signals

6.4.1 Penalty on the Rate of Change of Control Signals

The allowable control candidates are generated in a random fashion when IDP is used to solve the online MPC problem. In the earlier tries of the control experiments for the HVAC system, no penalty was imposed on the rate of change of the control signals. When the setpoint is $RT = 21^0C$ and $RRH = 60\%$, the experimental results are shown in Figure 6.4 and Figure 6.5. It can be seen that there are serious oscillations in the control variables in the steady state. Thus, the controller performance is not good. The controlled variables oscillate around the setpoint and the system cannot settle down at its setpoint. So imposing a penalty on the rate of change of the control signals is important to ensure that the controller has good reference tracking ability and achieve steady state.

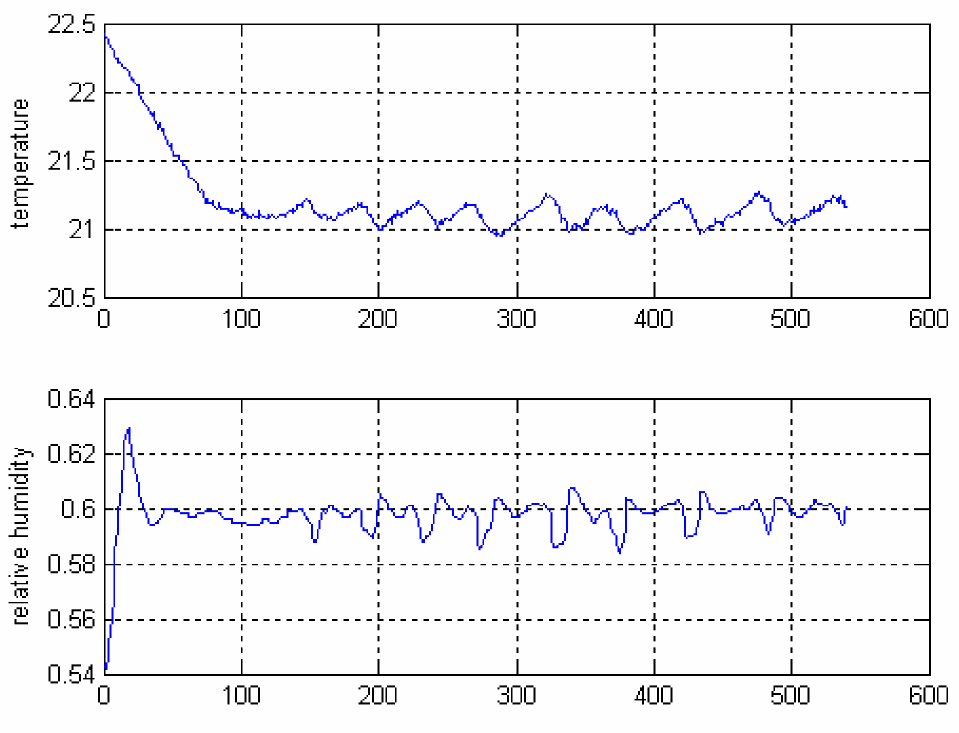

Figure 6.4 Room temperature and RH when no penalty imposed on change of rate

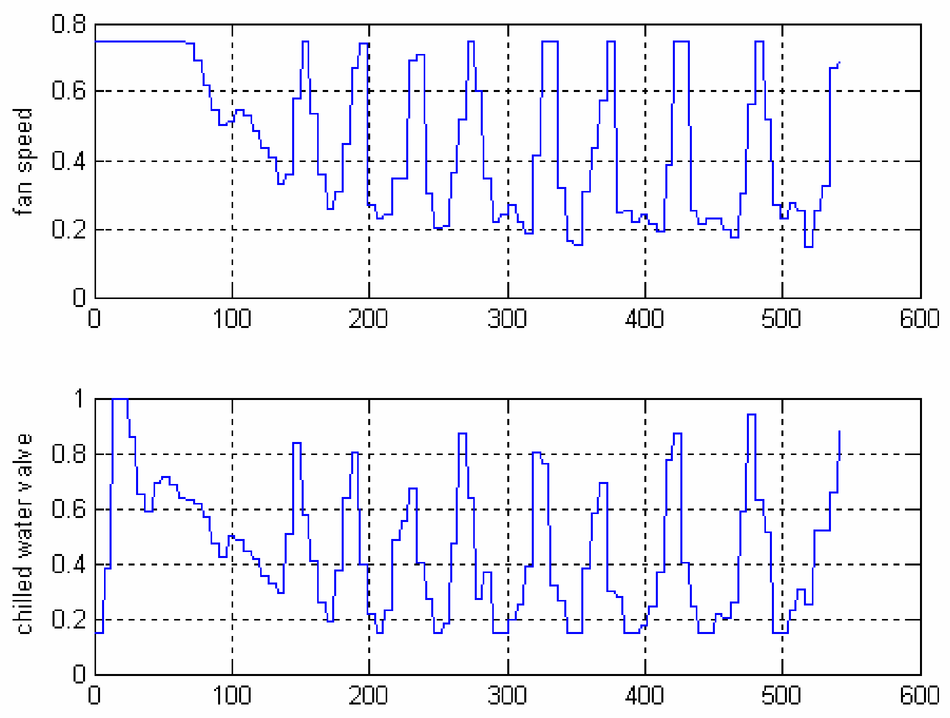

Figure 6.5 control signals when no penalty imposed on change of rate

6.4.2 Prediction Horizon

Increasing the prediction horizon will significantly increase the burden of computation. The computer that was used in this project is a Pentium III, 800 MHz PC. Limited by its computational ability, the maximum prediction horizon tested in this project is 4. From the experiments, it was found that the prediction horizon of 1 is good enough for the control. The response of the system with a prediction horizon of 1 has been shown in Figure 6.2 and Figure 6.3. An increase in the prediction horizon to 2 made some, but not very significant, improvement to the control performance. When the prediction horizon is 2, as shown in Figure 6.6 and Figure 6.7, the trajectory of the relative humidity is found to be smoother as compared with others and there is no overshooting.

Beyond 2, an increase in the prediction horizon causes system performance to degrade. For a prediction horizon of 3, there are significant overshoots in the room relative humidity, as shown in Figure 6.8 and Figure 6.9. This becomes worse when the prediction horizon was increased to 4 as shown in Figure 6.10 and Figure 6.11. The system can not track the setpoint of the room relative humidity and the steady error is quite big. The reason for the deterioration of the system response when the prediction horizon is increased beyond a certain point is due to the following reasons. Firstly, in MPC it is assumed that the supply air temperature and the relative humidity remain unchanged with the prediction horizon. However, in actual fact, these values change during this period and as the period is increased, the effect of the error caused by this assumption becomes more significant. A second reason is that when the prediction horizon is larger, the computational time required is larger. This computation time, which causes a time delay in sending the control signal, can become significant, particularly by with a slow computer, and thus cause error in the control performance.

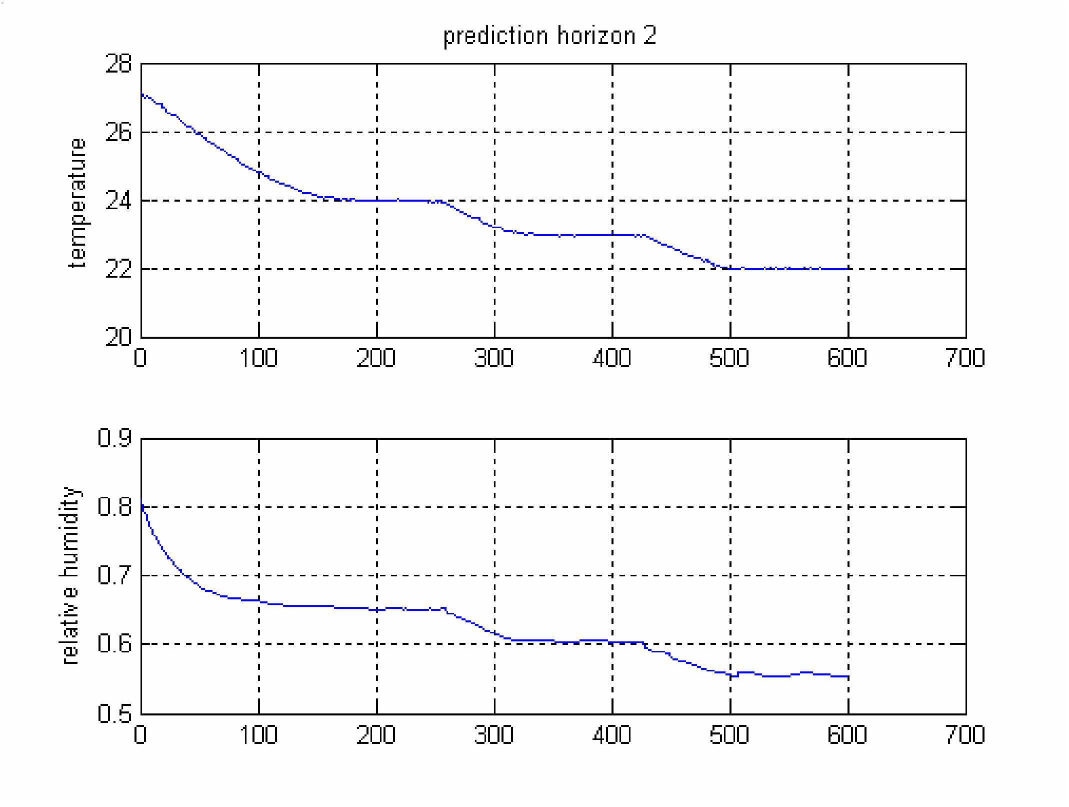

Figure 6.6 Control performance when P=2

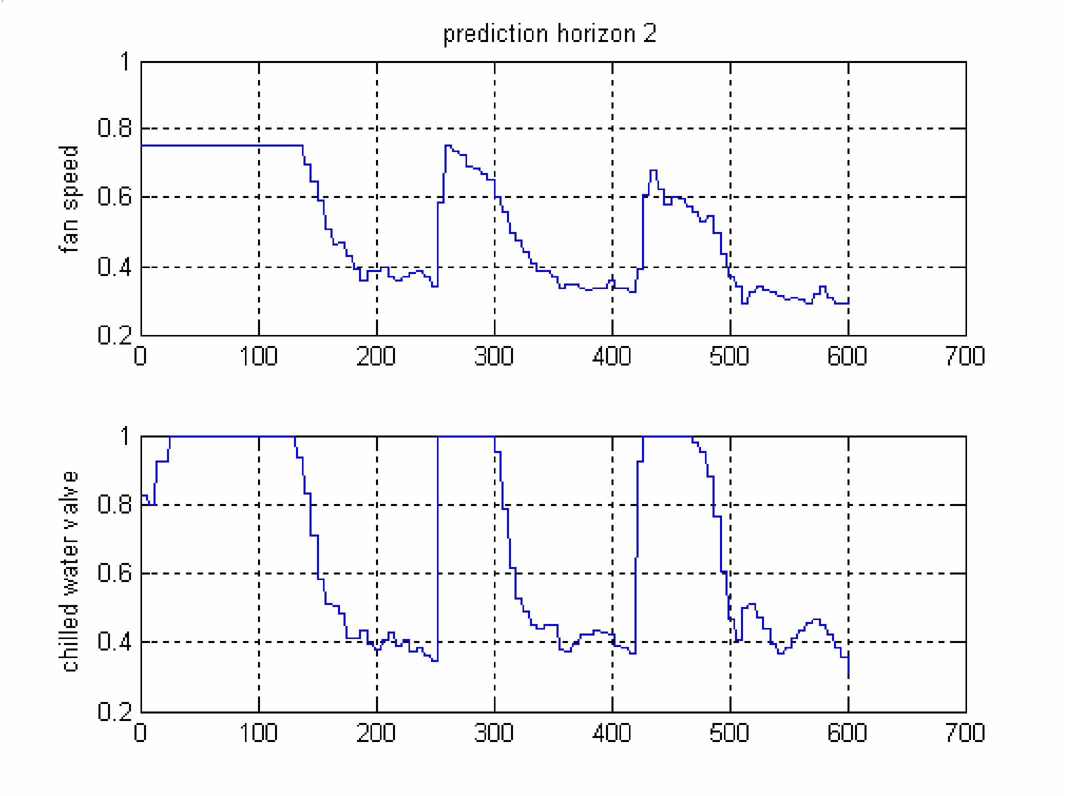

Figure 6.7 Control signals when P=2

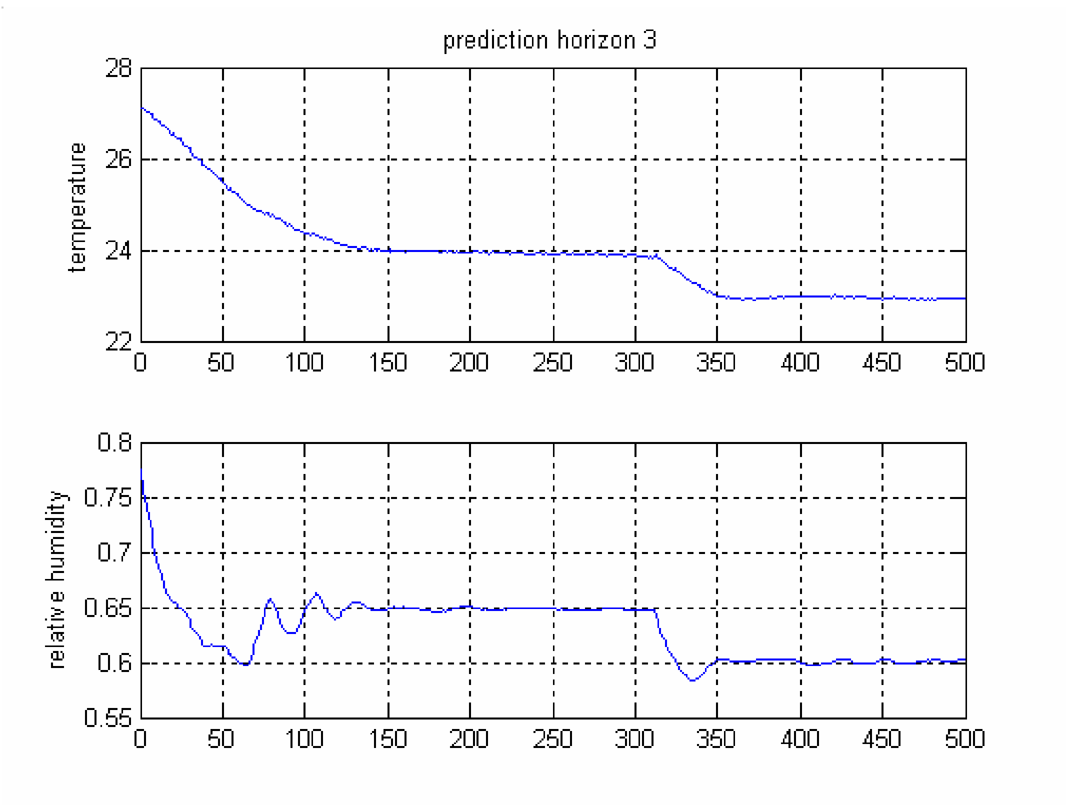

Figure 6.8 Control performance when P=3

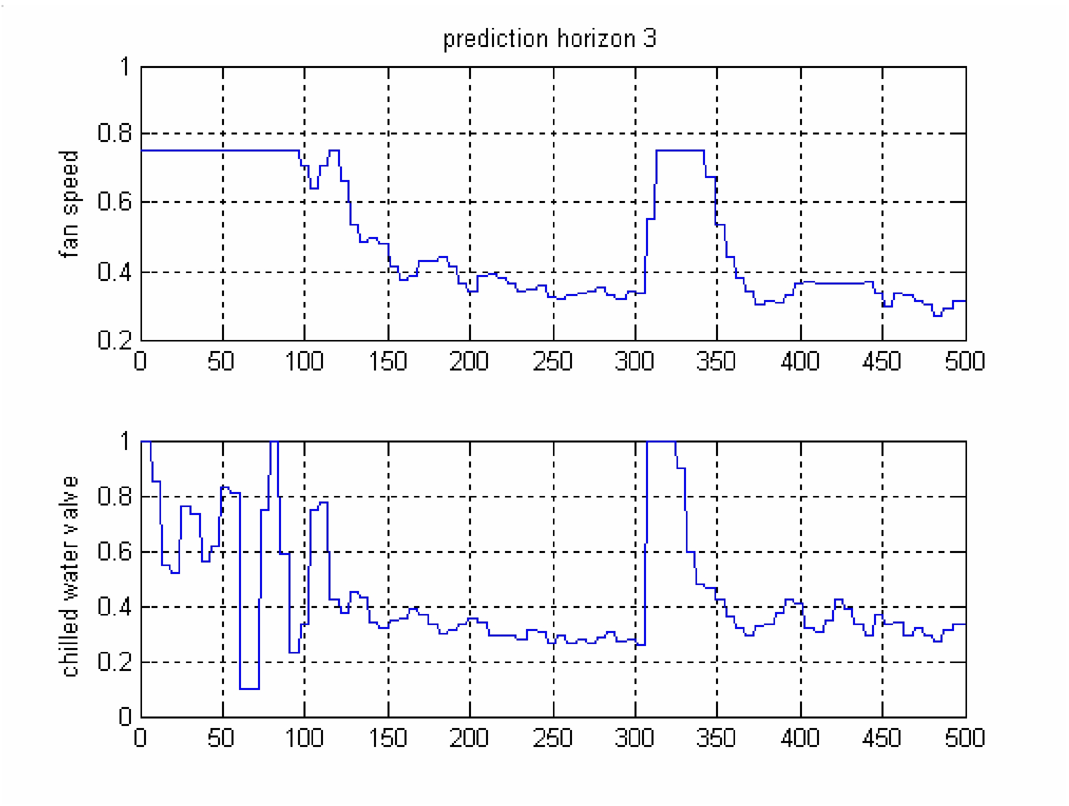

Figure 6.9 Control signals when P=3

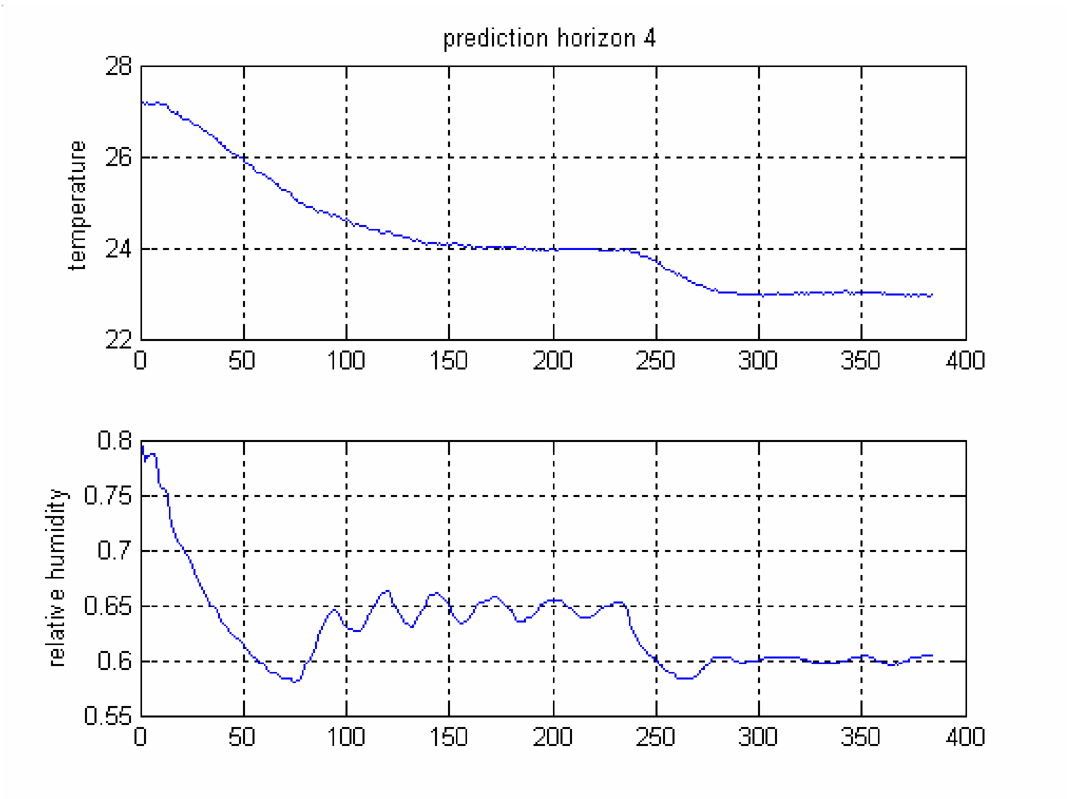

Figure 6.10 Control performance when P=4

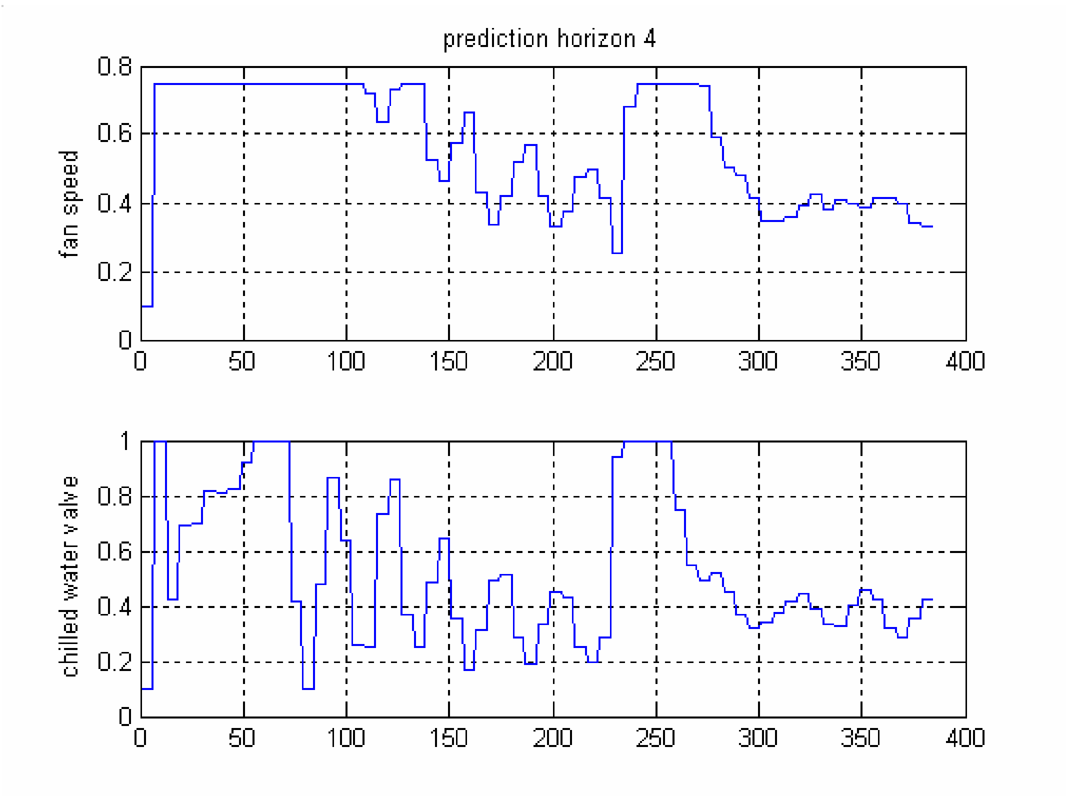

Figure 6.11 Control signals when P=4

6.5 Conclusion of SVR Model Predictive Control

By using MPC strategy, the control performance is found to be much better than that for inverse control. The iterative dynamic programming (IDP) is found to be a very reliable way to solve the online optimization problem for MPC. For IDP, constraints are necessary starting points of the computation. In this sense, IDP is suitable to the solution of the constrained MPC problem. Experimental results show that both room temperature and relative humidity are accurately controlled to their desired values respectively within the system operating range. The control performance is quite satisfactory in terms of reference tracking ability, steady-state error and amplitude of overshooting.

Chapter 7 Conclusion

This project focuses on the simultaneous control of temperature and relative humidity of a conditioned space, which is required by some industrial and scientific applications. From the step responses, it is noticed that both control signals, the supply air flow rate and the chilled water flow rate, affect the sensible and latent cooling capacity of the HVAC system. The system under control has strong couplings between the inputs and the outputs. The simultaneous control of room temperature and relative humidity is carried out by the means of varying both the supply airflow rate and the chilled water flow rate.

HVAC plants are typical nonlinear systems and obtaining accurate models for these systems is a difficult and challenging task. In this project, a new tool—support vector regression— is used to model the inverse and forward dynamics of this highly nonlinear system.

Support vector regression is a type of model that is optimized so that prediction error and model complexity are simultaneously minimized. Because of its universal approximation ability, support vector regression can be used to model nonlinear processes, just as neural networks are. One advantage of SVR over neural networks is that SVR formulates regression as a quadratic optimization problem which ensures that there is only one global minimum while the training of neural networks may get "trapped" at a local minimum. Another advantage is that training of the SVM is faster than that of neural networks. This is a desirable property for most applications in general and online applications in particular.

Both the SVR inverse control and SVR model predictive control consist of two stages. The first is the system identification for the HVAC system. For inverse control, SVR inverse models are needed while for model predictive control, SVR forward models are needed. Choosing optimal hyper-parameters for the models is an important step in the identification stage. *k*-fold cross validation is a reliable way to determine the optimal hyper-parameters. The optimal values are firstly searched in coarse grids, and then searched in finer grids. The final models are obtained after training the SVRs using these optimal hyper-parameters. The models obtained this way are found to have good generalization property.

In inverse control, the inverse model is simply cascaded with the controlled system in order that the composed system results in identity mapping between the desired response and the controlled system output. Thus, SVR models act directly as the controllers in such a configuration. It is important to design an appropriate reference for the system to follow. The controller has the ability of set point tracking and disturbance rejection. The controller can work effectively in the start up period which is difficult to be described by a linear model around certain operating points. However, the disadvantage of the SVR inverse controller is that the response time is quite slow. The cooling capacity is not exploited in its full potential, which is demonstrated by the fact that the fan never runs at its maximum speed and the valve is never fully open.

The basic idea of MPC is to predict the controlled variables over a future horizon using a prediction model of the process, the control signals are then computed by minimizing an objective function, and only the first control action is finally applied to the process. The procedure is repeated at every sampling instant using the updated information (measurements) of the process. A key advantage of MPC over other control schemes is its ability to deal with constraints in a systematic and straightforward manner. The online MPC problem is solved by iterative dynamic programming (IDP).

The items in the performance index are found to have significant impact on the controller performances. Because variables in the performance index have different dimensions and units, making the items of the performance index dimensionless will make the tuning process easier. It has also been found, in the work done here, that putting some penalty on the rate of change of the control signals helps to reduce fluctuations in the outputs once they have settle down. This is, in a way, akin to having derivative feedback. The strategy of updating initial conditions is found to be important for the controller to have faster response and good reference tracking ability. The length of the prediction horizon can affect both control performance and the computational burden. Initially, increasing the prediction horizon can help to improve control performance. However, beyond a certain point, increasing this will cause control performance to deteriorate.

The MPC strategy has been proved to be successful experimentally. Experimental results show that both the room temperature and the room relative humidity are accurately controlled to their desired values respectively within the system operating range. The control performances are quite satisfactory in terms of reference tracking ability, steady-state error, amplitude of overshooting and consideration of control constraints.

Future research directions could include adaptive SVR controller and robustly stable model predictive control (MPC).

1. Adaptive SVR model predictive control

Support vector model predictive control is data-based. Therefore, the performance of the controller is very sensitive to the quality of the data. The dynamics of a HVAC system will change after a period of operation. For example, the dust in air will adhere to the coils of AHU, so heat transfer coefficient will decrease by some extent. In order to adapt to the changing dynamics, it is necessary to adopt the adaptive SVR controller. It could be performed in such a way that the support vectors of the model will be updated periodically so that the SVR model will reflect the change of the plant dynamics.

2. Stability and robustness analysis

In this project, the tuning process of the items in the performance index is just based on trial and error. Actually, the stability issue of MPC has reached to a fairly mature stage. It could be possible to be used to design a stable MPC controller. While the stability issue of MPC has reached a mature stage, the robustness is still an open topic. A promising way is to combine H_∞ control, which ensures robustness, and MPC, or receding horizon control, which is computationally feasible.

Reference

Arima, M. and E.H. Hara and J.D. Katzberg, A fuzzy logic and rough sets controller for HVAC systems. IEEE Conf. Communications, Power, and Computing. Vol. 1:133-138, 1995.

ASHRAE Handbook. Heating, Ventilating, and Air-Conditioning Applications. Atlanta, GA: American Society of Heating, Refrigerating and Air Conditioning Engineers. 1999.

Bojkov, B., and R. Luus, Evaluation of the parameters used in iterative dynamic programming. Can. J. Chem. Eng., Vol. 71, pp. 451–459, 1993.

Cai, Z. X., Intelligent control: Principles, Techniques and Application, World Scientific, Singapore, 1997.

Camacho, E.F. and C. Bordons, Model Predictive Control, Springer, London, 1999.

Dadebo, S. A., and K. B. Mcauley, Dynamic optimization of constrained chemical engineering problems using dynamic programming. Computers and Chemical Engineering, Vol. 19, pp. 513–525, 1995.

Duan, K., S.S. Keerthi and A.N. Poo, Evaluation of simple performance measures for tuning SVM hyperparameters, *Neurocomputing*, Vol. 51, pp.41-59, April 2003.

Duarte, M., A. Suarez and D. Bassi, Control of grinding plants using predictive mulitivariable neural control, Power Technology, Vol. 115, pp.193-206, 2001.

Flake, Gary William and Steve Lawrence, Efficient SVM regression training with SMO, Machine Learning, Vol. 46, pp. 271-290, 2002.

Fletcher, R., Practical Methods of Optimization, Second Edition, (A Wiley-Interscience Publication), John Wiley and Sons, Tiptree, Essex, UK, 1987.

Gu, Dongbing and Huosheng Hu, Neural predictive control for a car-like mobile robot, Robotics and Autonomous Systems, Vol. 39, pp. 73-86, 2002.

Haykin, S. Neural Networks: A Comprehensive Foundation (2nd ed.). Upper Saddle River: Prentice Hall, 1999.

Henson, Michael A. and Dale E. Seborg, Nonlinear Process Control, Prentice Hall PTR, Upper Saddle River, NJ, 1997.

Hunt, K.J., D. Sbarbaro, R. Zbikowski and P.J. Gawthrop, Neural Networks for Control Systems—A Survey, Automatica, Vol. 28, No. 6, pp 1083-1112, 1992.

Isermann, R., Digital Control Systems, Springer-Verlag, 1981.

Kasahara, M., T. Matsuba, Y. Kuzuu, T. Yamazki, Y. Hashimoto, K. Kamimura and S. Kurosu. Design and Tuning of Robust PID Controller for HVAC Systems. ASHRAE Transactions, Vol. 105 (2), pp. 154-166. 1999.

Keerthi, S. Sathiya, Lecture notes for Neural Networks, National University of Singapore, 2002.

Keerthi, S.S. and E.G. Gilbert, Optimal infinite –horizon feedback laws for a general class of constrained discrete-time system: stability and moving –horizon approximation, Journal of Optimization Theory and Application, Vol. 57, No. 2, pp. 265-293, 1988.

Khalid, M., S. Omatu and R. Yusof, Temperature regulation with neural networks and alternative schemes. IEEE Transaction on neural networks, Vol. 6. No. 3, 572-582, 1995.

Krakow, K.I., S. Lin. and Z. Zeng. Temperature and Humidity Control During Cooling and Humidifying by Compressor and Evaporator Fan Speed Variation. ASHRAE Transactions, Vol. 101, pp. 292-304, 1995.

Kruif, B. J. de and Theo J.A. de Vries, On using support vector machine in learning feed-forward control, 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics Proceedings, Corno, Italy, 8-12 July 2001.

Li. Q., A.N. Poo, C.M. Lim and M. Ang, Neuro-based adaptive internal model control for robot manipulators, IEEE International Conference on Neural Networks, Perth 1995.

Ljung, L., System Identification: Theory for the User, Upper Saddle River, NJ: Prentice Hall, 1999.

Luus, R., Optimal control by dynamic programming using systematic reduction in grid size, International Journal of Control, 19, 995-1013, 1990.

Luus, R.: Application of iterative dynamic programming to very high-dimensional systems, Hung. J. Ind. Chem. 21, 243-250, 1993.

Luus, Rein, Iterative Dynamic Programming, Chapman & Hall/CRC, 2000.

Mayne, D.Q. and H. Michalska, Receding horizon control nonlinear system. IEEE Transactions on Automatic Control, Vol. 35, pp. 814-824, 1990.

Mayne, D.Q., J.B. Rawlings, C.V. Rao, P.O.M. Scokaert, Constrained model predictive model control: Stability and optimality, Automatica, Vol. 36, pp 789-814, 2000.

Miao, Qi and Shi-Fu Wang, Nonlinear model predictive control based on support vector regression, Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 4-5 November 2002.

Morari, Manfred and Jay H. Lee, Model predictive control: past, present and future, Computers and Chemical Engineering, Vol. 23, pp 667-682, 1999.

Platt, J.C., Fast training of support vector machine using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, Advances in Kernel Methods--Support Vector Learning, pages 185-208, MIT press, Cambridge, MA, 1998

Potocnik, Primoz and Igor Grabec, Nonlinear model predictive of a cutting process, Neurocomputing, Vol. 43, pp.107-126, 2002.

Rosandich, R. Understanding Controllers and Control Terminology. ASHRAE Journal, Vol. 39, No. 9, pp. 22-25. 1997.

Rusnák, A., M. Fikar, M. A. Latifi and A. Mészáros, Receding horizon iterative dynamic programming with discrete time models, Computers and Chemical Engineering, 25, 161-167, 2001.

Schölkopf, B. and A. Smola, Learning with Kernels: Support Vector Machines, Regularization,Optimization and Beyond, MIT Press, Cambridge, MA, 2002.

Shepherd, Keith, VAV air conditioning systems, Blackwell Science, 1998.

Shevade, S.K., S.S. Keerthi, C. Bhattacharyya and K.R.K.Murthy, Improvements to the SMO algorithm for SVM regression, IEEE Transactions on Neural Networks, Vol. 11, pp.1188-1194, Sept. 2000.

Smola, A. and B. Schölkopf, A Tutorial on Support Vector Regression, NeuroCOLT Technical Report TR 1998-030, Royal Holloway College, London, UK, 1998

Suykens, J. A. K. , Vandewalle, J. and Moor, B. De, Optimal control by least squares support vector machines, Neural Networks, Volume 14, Issue 1, January 2001, Pages 23-35

Thibault, J.and B.P.A. Grandjean (1991). Neural Networks in process control - A survey. In *IFAC International Symposium Advanced Control of Chemical Process (ALXHEMpl)*, Toulouse, France, 295-304.

Vapnik, V., The Nature of Statistical Learning Theory. Springer Verlag, New York, 1995.

Yakowitz, Sidney and Ferenc Szidarovszky, an Introduction to Numerical Computations, Second Edition, Macmillan Publishing Company, New York, 1989.

Zadeh, L. A., Fuzzy sets, Information and Control, Vol. 8, pp.338-353, 1965

Appendix

Implementation of Iterative Dynamical Programming

1. First Iteration

From the given initial condition $y(0)$ and constraints specified by Eq. (6.5), we can choose the centre point for the y-grid and allowable range for control.

1) Stage *P*

Let us start the calculation at stage *P*. For each y-grid point, evaluate *M* values of the performance index, where each of the *M* values of control used for $u(P-1)$ in turn. Compare these *M* values of the performance index and choose the particular value of $u(P-1)$ that gives the minimum value. This is the best control to use at that particular y-grid point.

2) Stage *P*-1

Now step backward to stage *P-1*. For each grid point, we again consider *M* allowable values for control. However, when we calculate the stage from *P-1* to *P* stage, it is unlikely that the stage $y(P)$ will be exactly one of grid points at stage *P*, The problem of not hitting a grid point exactly is illustrated in Figure A.1. For simplicity we have taken $n = 2$, $N = 5$, $M = 4$. Therefore the grid consists of a (5×5) matrix. At the grid point (2, 3) of stage *P*-1 we have shown 4 trajectories to stage *P*, corresponding to the use of the four allowable values of control, namely *u=a, b, c* and *d.* None of these trajectories hits a grid point at stage *P*.

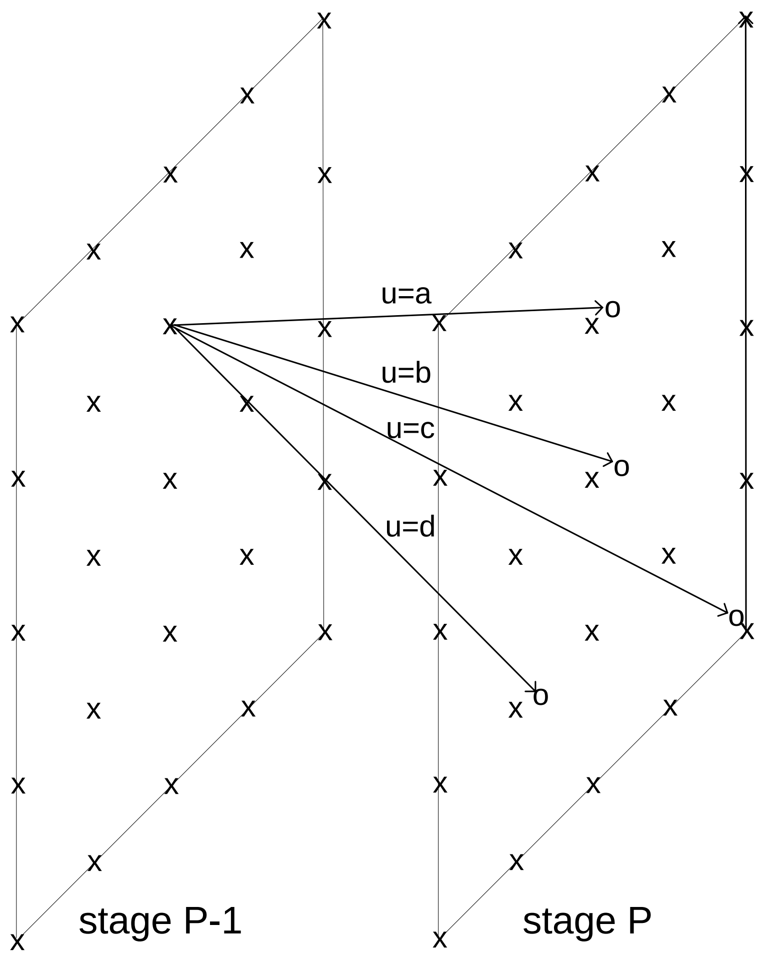

Figure A.1 Illustration of the difficulty of reaching the grid points by assigning 4 values for control

To continue the prediction to the final stage *P*, we take the optimal control policy corresponding to the grid point that is closest to the state $y(P)$. This gives a good approximation if a sufficiently large number of grid points and allowable values for control are used. As shown Figure A.1, to continue the prediction for the first trajectory *u*=*a*, we use the optimal control at stage *P* corresponding to grid (2, 3); to continue the second trajectory corresponding to *u*=*b*, we use the best control policy at (3, 3); for *u*=*c* we use (5, 5); and to continue the trajectory corresponding to *u*=*d*, we use the optimal control policy established for the grid point (4, 2) at stage *P*, At the stage *P*+1, we have four values for the performance index to compare and we select the control policy that gives the minimum value. Therefore, the control policy for the grid (2, 3) at stage *P*-1 is established. This is continued for the remaining 24 grid points to finish the calculation for stage *P*-1.

3) Continuation in with backward direction

We proceed in this manner with stage *P*-2, *P*-3, …, etc., until stage 1 is reached. A stage 1 the grid consists only of the initial condition $y(0)$. At this stage we compare the *M* values of the performance index and pick the control policy that gives the minimum value. This finishes the first iteration. Even if a reasonably large number of grid points and allowable values for control are chosen, the optimal control policy obtained is quite far from the global optimal solution. Therefore, it is necessary to improve the control policy obtained the first iteration, and we proceed to the main part of optimization procedure.

2. Iterations and Passes with Systematic Reduction in Region Size

The optimal trajectory from the first iteration provides the centre for the y-grid at each stage, and the optimal control policy from the first iteration gives the central value for the allowable values for control at each stage. The corresponding regions are contracted by a small amount to provide a finer resolution and the procedure is continued for a number of iterations. Several iterations consist of one pass. One method of preventing the collapse of the search region is to use the iterative dynamic programming in a multi-pass fashion, so that the region is restored to fraction of its size at the beginning of previous pass. When this procedure is carried out for a sufficiently large number of passes, it is expected that convergence to the optimal policy is obtained with sufficient accuracy.