Master's Thesis 2014

Candidate: Sharamnsha Bhandari

Title: Evaluation and Comparison of MPC Algorithms applied to a real air heater

Telemark University College
Faculty of Technology
Kjølnes
3914 Porsgrunn
Norway
Lower Degree Programmes – M.Sc. Programmes – Ph.D. Programmes

Telemark University College
Faculty of Technology
M.Sc. Programme

MASTER'S THESIS, COURSE CODE FMH606

Student:	**Sharamnsha Bhandari**
Thesis title:	**Evaluation And Comparison of MPC Algorithms applied to a Real Air Heater**
Signature:	. . . . .
Number of pages:	73
Keywords:	Model Predictive Control, Non-linear MPC, Linear MPC, Air heater, fmincon, quadprog

Supervisor:	Finn Aakre Haugen	sign.:	. . . . .
2nd Supervisor:	<name>	sign.:	. . . . .
Censor:	<name>	sign.:	. . . . .
External partner:	<name>	sign.:	. . . . .
Availability:	<Open/Secret>		

Archive approval (supervisor signature)**:** sign.: . . . . . **Date :** . . . . . .

Abstract:

With a remarkable increase in demand for safe, efficient and stable control operation, process control for process plants has become more important. Although Conventional PID controllers have been used in the process industries for the longest time, the need of advanced control strategies such as MPC has been felt in not only the process industries but also in many academic and research activities. This thesis tries to draw attention towards basic idea of MPC theory, the controller performance and its practical implementation in the real physical process. For the implementation purpose, Linear and non-Linear MPC has been developed in the simulated process using the process model before its implementation on a real physical process.

The MPC control algorithm has been tested in the air heater system, where the objective is to keep the temperature of the air heater to the desired reference temperature. The two predictive control algorithms have been compared after applying controller to the process model/process in terms of relative stability margins (Phase Margin and Gain Margin), set point tracking (IAE criterion used as a measure), Horizon length and computational time.

It is noticed that non-linear MPC is better than linear MPC in terms of IAE as the error computed in non-linear MPC is found to be less. In addition, it is observed that non-linear MPC requires shorter horizon length as compared to linear MPC to give equally good, if not better results. However, if same horizon length is used for both linear and non-linear MPC, linear MPC requires less computational time than that of non-linear MPC. Based on the results obtained, it is observed that neither of the predictive control algorithms is that robust against any model change, which is analyzed after calculating Phase Margin for the controllers. In addition, the concept of Gain Margin is not that clear in MPC as it is difficult to calculate Gain Margin, as the controller response is not subjected to the stability limit even with the increase in loop gain that is inserted between controller and the process.

Kalman filter as the state estimator is used especially to estimate the ambient temperature, which cannot be measured directly. It makes sense to use kalman filter to estimate the states as the temperature at the surroundings influences the true air heater temperature.

Table of Contents

PREFACE **5**

NOMENCLATURE **6**

1 INTRODUCTION **9**

1.1 BACKGROUND 9
1.2 OBJECTIVES OF THE THESIS 9
1.3 APPROACH 10
1.4 PREVIOUS WORK 10

2 LITERATURE REVIEW **11**

2.1 LITERATURE REVIEW OF MPC 11
2.2 LINEAR AND NON LINEAR MPC 12

3 SYSTEM DESCRIPTION **14**

3.1 MATHEMATICAL MODEL 14
3.1.1 Time Delay *16*

4 STATE ESTIMATION **19**

4.1 INTRODUCTION 19
4.2 EXTENDED KALMAN FILTER ALGORITHM 20
4.3 TUNING OF KALMAN FILTER 22

5 MODEL PREDICTIVE CONTROL **23**

5.1 INTRODUCTION 23
5.1.1 Performance Index *23*
5.1.2 Constraints *24*
5.1.3 Process Model *24*
5.2 ADVANTAGES OF USING MPC 24
5.3 MPC IMPLEMENTATION 25
5.3.1 Implementation of Linear MPC *25*
5.3.2 Implementation of Non - Linear MPC *28*

6 RESULTS **30**

6.1 SIMULATION 30
6.1.1 Linear MPC *30*
6.1.2 Non-Linear MPC *33*

7 DISCUSSION **37**

7.1 STABILITY MARGINS 37
7.2 SET POINT TRACKING 39
7.3 HORIZON LENGTH 40
7.4 COMPUTATIONAL TIME 40

8 PRACTICAL RESULTS **41**

9 CONCLUSIONS AND RECOMMENDATION **43**

9.1 CONCLUSION 43

9.2 FUTURE RECOMMENDATIONS 44

Preface

This thesis is submitted for the academic degree of Master of Science in the field of system and Control Engineering at Telemark University College, Porsgrunn.

I would like to express my sincere thanks to my supervisor Associate Professor Finn Aakre Haugen for his constant support and motivation throughout the thesis, and for providing me the necessary materials to complete the thesis.

I am highly indebted to Associate Professor David Luigi Ruscio, as the foundation of the thesis is based on his Lecture notes, papers and exercises.

I would like to thank to Hans Peter Halvorsen for allowing me to use the lab, and for providing the necessary equipments needed for the thesis.

Finally, I would like to thank my friends and family for their support during the thesis.

Porsgrunn, 4th June, 2014

Sharamnsha Bhandari

Nomenclature

List of Symbols:

Symbol	Description	Unit
T_{out}	Temperature at the outlet of the air heater system	[deg C]
θ_t	Time constant	[s]
θ_d	Time Delay	[s]
K_h	The Heater Gain	[deg C/V]
T_{env}	Ambient Temperature	[deg C]
u	Control Input	[V]
Ts	Sampling Time	[s]
v	Process noise	
w	Measurement noise	
K	Kalman gain	
x_{apost}	Aposteriori state vector	
P_{apost}	Covariance of estimation error	
Q	Process noise auto covariance	
R	Measurement noise auto covariance	
J	Performance Index	
Q	Output Error weight Matrix	
P	Control Weight Matrix	
L	Prediction Horizon	
r	Reference value	

Abbreviations:

MPC : Model Predictive Control

Prbs: pseudo Random Binary Signal

IAE: Integral Absolute Error

PM: Phase Margin

GM: Gain Margin

SISO: Single Input Single Output

MIMO: Multiple Input Multiple Output

ARMAX: Auto Regression Moving Average With Exogenous

List of Figures

Figure 1:Basic Concept for model predictive control(Predictive et al., n.d.) 11
Figure 2:Block Diagram of Kalman Filter 19
Figure 3: Kalman fitler estimation of the state variables 21
Figure 4: Estimation error 21
Figure 5: Block Diagram of Optimization and Control system 25
Figure 6: Unconstrained MPC control action for Linear MPC 30
Figure 7:Constrained MPC Control action for Linear MPC 31
Figure 8:Unconstrained linear MPC control where states are updated by kalman Filter 32
Figure 9:Constrained linear MPC control where states are updated by kalman Filter 32
Figure 10:Unconstrained MPC control action for Non-Linear MPC 33
Figure 11:Constrained MPC control for non-linear MPC 34
Figure 12: Unconstrained non-linear MPC control where states are updated by kalman Filter 35
Figure 13:Constrained non-linear MPC control where states are updated by kalman Filter 35

1 Introduction

1.1 Background

To achieve a better performance for a process with tighter specifications requirement, process control has become more significant. Conventional PID controllers have been used in the process industries, as it also deliver a good performance, provided that it is finely tuned. But, there is a growing demand for process control systems to become more safe, strict and efficient to maintain a process at the desirable operating conditions(Kwee, Tan, & Wern, 2006). Model Predictive Control can tackle with the growing industrial demand for better process control.

Conventional feedback control tries to adjust the control signal only when it observes some changes in the desired output. However, MPC is a technique, which tries to adjust the control signal before occurrence of any changes in the set point. Therefore, with the combination of conventional feedback operation along with the prediction ability, it gives controller more flexibility to make control adjustments more smoother and close to optimal control action values(H. P. Halvorsen, 2011).

The purpose of this thesis is to have an understanding of predictive control algorithm and investigate the behavior of MPC controller applied to a physical process. The system used for the demonstration is the air heater system that has been used for control and monitoring purposes at Telemark University College. The description of the laboratory scale Air heater system is given in (F. Haugen, n.d.).

The continuous time state space mathematical model of the air heater, which represents the real air heater system, is discretized to get a discrete time state space model to implement MPC controller. The results that are obtained by applying controller to both the simulated and real system have been discussed and compared.

1.2 Objectives of the thesis

The main objectives set for this thesis are as follows:

- Develop a MPC controller for keeping Tout, which represents the temperature at outlet of the air heater system, to a reference temperature.
- Investigate and compare the controller performance of Non-linear MPC and Linear MPC applied to the real system of air heater.
- Implement a state estimator to estimate the state variables in the process, i.e. the ambient temperature that cannot be measured directly.

1.3 Approach

The controller performance has been tested on the real physical process . However, to begin with, controller is tested on the simulated process. The mathematical process model of the real air system is used for the simulation purpose. MATLAB has been used as a software for the entire thesis.

The non-linear mathematical model, non-linear in a sense that the model includes time delay in it, is used. For, the sake of generality, state space modeling is used while implementing both the linear and non Linear MPC.

To implement Linear MPC, LQ optimal controller of incremental form is developed, with an assumption that the controller is insensitive to slowly varying process and measurement noise. And, to implement non linear MPC, 'fmincon' is used for the optimization purpose. In both the cases, both unstrained and constrained MPC control is designed in a simulated process. However, to implement the controller in a physical process, constrained MPC control is used with imposing constraints in the manipulated variable.

Kalman filter as a state estimator is used to estimate the state variables in the process. The estimator is required especially to estimate the ambient temperature.

The controller performance has been analyzed and compared with focus on stability margins, computational time, set point tracking and horizon length.

1.4 Previous Work

Linear MPC implementation presented in this thesis is based on (Ruscio, 2012b). In addition, reference is taken from (F. A. Haugen, n.d.) for the implementation of non-linear MPC. Beside this, there has been a substantial amount of literature available for the implementation of MPC to a process(Processes, 2010)(Strand & Sagli, n.d.),(Nagy & Braatz, 2003). In addition, various lab works (H. P. Halvorsen, 2011)(H. Halvorsen, n.d.), have been carried out at Telemark University College for the control operation of temperature of air heater system, including MPC.

2 Literature Review

2.1 Literature Review of MPC

MPC refers to a class of optimal control theory developed in the 1960 and latter(Ruscio, 2010). As an advanced control strategy, there is a growing interest to use MPC in the industries, mainly due to its ability of handling process constraints, which has been taken out of consideration by most of the conventional controllers. In addition, it has the ability to handle long delay time. MPC has been implemented successfully in many industrial application for last 30 years.(Qin & Badgwell, 2003). According to (Qin & Badgwell, 2003), there were over 4500 applications worldwide by the end of 1999, mostly used in oil refineries and petrochemical plants. The transition of MPC adoption over three decades has been presented by (Lee, 2011) . The paper gives an insight about the history of inception of thought of using MPC to the familiarity of the term "fast MPC" in modern days.

The limitation with MPC is its dependence on the accuracy of the model. Nevertheless, if a reasonably accurate dynamic model of a process is available, the model along with the present measurement can be used to predict future values of the outputs(Dale R. Seborg, Thomas F. Edgar, n.d.).

The basic concept of MPC is illustrated in Figure 1.

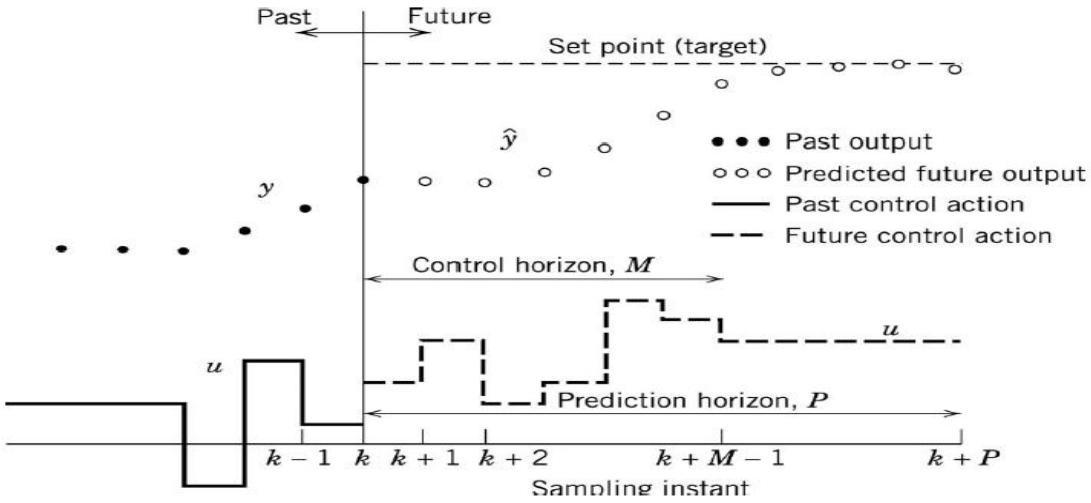

Figure 1:Basic Concept for model predictive control(Control, 2011)

The MPC control calculations are based on determining sequence of control moves so that the predicted response tracks the set point in an optimal manner. The actual output y, predicted output $\hat{y}$ (Controlled variable) and the control input (manipulated variable) is shown in the figure 1. The control operation is such that at k^{th} sampling instant the sequence of manipulated variable, u at the next M sampling instants, i.e. {u(k), u(k+1)…..u(k+M-1)} are

calculated. This set of control inputs is calculated to observe the predicted outputs to track the reference trajectory in an optimal manner. An objective function is defined and the control calculations are based on optimization of this objective function. The number of Predictions P is termed as Prediction horizon, while the number of control moves M is termed as the control Horizon. Although a sequence of M control moves are calculated at each sampling instant, only the first control move , i.e. u(k) is implemented in the system. Then at next sampling instant, a new sequence of control input is calculated and the first control move is implemented. The algorithm is repeated at each sampling instant. Due to this feature in MPC, it is also termed as Receding Horizon approach(Predictive et al., n.d.).

A detail review of MPC theory is provided in (Mayne et al., 2000). A literature review of robustness, stability and controller performance in MPC is provided in (Bemporad & Morari, n.d.).

2.2 Linear and Non linear MPC

Linear MPC refers to a class of MPC strategies where linear models are used to predict the dynamics of the system, even though the dynamics of the system non- linear due to the presence of constraints. But, most of the systems are inherently non-linear in nature and with narrow specifications, linear models may be insufficient to describe the system dynamics accurately. This encourages using non-linear Model predictive control to observe the controller performance for describing the system dynamics. (Findeisen & Allg, n.d.).

(Tricaud, 2008) has presented the feasibility study of Model predictive control technique in a DC motor and has suggested that the non linear MPC can be used to a field of canal integration where only linear MPC has been used, as the model used has non linear dynamics in it.

Various studies of Linear and non linear MPC has been done in a physical process, namely in control of distillation column. (Ramesh, Hisyam, Aziz, & Shukor, 2012) has presented the implementation of non linear MPC for the control of distillation column, where the state variables are estimated using Unscented Kalman filter. A comparison of linear MPC over conventional PID controller is presented in (Manimaran, Arumugam, Balasubramanian, & Ramkumar, n.d.) Where linear MPC has been implemented for the control operation of Distillation columns. An interesting perspective on non linear MPC has been presented by (Bequette, 2007) which discusses about the challenges due to non convex nature of the optimization problem in non linear MPC and encourages readers to give more attention to disturbance estimation and prediction. In addition, there are various existing literatures that can be found which show the implementation of linear and non-linear MPC in simulated and real physical process.

Non-linear MPC takes account of inherent non linear dynamics of the process. Although, the need of non linear MPC has been felt in many academic research activities or in industrial process, Linear MPC is still more popular than non linear MPC in industries(Kwee et al., 2006).

3 System Description

3.1 Mathematical Model

The dynamics of the air heater system is governed by mathematical model given as in the equation (1)(F. Haugen, n.d.).

$$\dot{T}_{out} = \frac{1}{\Theta_t}\{-T_{out} + [K_h u(t-\Theta_d) + T_{env}] \quad (1)$$

Where,

T_{out} represents the air temperature at the tube outlet that we wish to control

$u[V]$represents the control signal to the air heater

$\Theta_t[secs]$ represents the time constant

$K_h[degC/V]$ represents the heater gain

Θ_d represents the time delay

T_{env} represents the ambient temperature

Here, the manipulated variable is a control signal applied to the air heater. In addition, the temperature at the tube outlet represented by T_{out} in the equation (1) represents the ordinary state variable. Since, we wish to estimate the ambient temperature (T_{env}), it makes sense to augment the ambient temperature to the ordinary state vector. It is assumed that the ambient temperature is slowly time varying variable and is almost constant.

The states input and the parameters that are represented in equation (1) can be listed in a state vector, input vector, and Parameter vector as:

X= $[T_{out}, T_{env}]^T$, $\in \mathbb{R}^{nx2}$ represents state vector

u= $[u]^T$, u$\in \mathbb{R}^{rx1}$ represents input vector

Θ= [K_h, Θ_t] represents parameter vector

The mathematical model given in equation (21) can be put into continuous time state space form. As, there are two states in the model,

Let us suppose,

x_1= T_{out} and

x_2 = T_{env}

With the use of an assumption made for the ambient temperature,

$$\dot{x_2} = 0 \tag{2}$$

$$\dot{x_1} = \frac{-1}{\theta_t}x_1 + \frac{1}{\theta_t}x_2 + \frac{K_h}{\theta_t}u(t-\theta_d) \tag{3}$$

The Output is the temperature at the outlet of the air heater system given by,

$$y = x_1 \tag{4}$$

Therefore, final deduced continuous time state space model obtained using equation (2) , equation (3) and equation (4) for the system is,

$$\begin{bmatrix}\dot{x_1}\\ \dot{x_2}\end{bmatrix} = \begin{bmatrix}\frac{-1}{\theta_t} & -\frac{1}{\theta_t}\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}\frac{K_h}{\theta_t}\\ 0\end{bmatrix}u(t-\theta_d) \tag{5}$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} \tag{6}$$

The discrete state space model can be obtained using equation (5) and equation (6). The discretization is done using Forward Euler Method.

$$\frac{x_1(k+1)-x_1(k)}{T_s} = \frac{-1}{\theta_t}x_1(k+1) + \frac{1}{\theta_t}x_2(k+1) + \frac{K_h}{\theta_t}(k-\theta_d)$$

$$x_1(k+1) = \left(1-\frac{T_s}{\theta_t}\right)x_1(k) + \frac{T_s}{\theta_t}x_2(k) + \frac{T_sK_h}{\theta_t}u(k-\theta_t)$$

$$\frac{x_2(k+1)-x_2(k)}{T_s} = 0$$

$$x_2(k+1) = x_2(k)$$

And,

$$y(k) = x_1(k)$$

Where Ts represents sampling time.

Therefore, the final discrete time state space representation can be written in the form as in equation (7) and equation (8),

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} \left(1 - \frac{T_s}{\Theta_t}\right) & \frac{T_s}{\Theta_t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} \frac{T_s K_h}{\Theta_t} \\ 0 \end{bmatrix} u(k - \Theta_t) \tag{7}$$

And,

$$y(k) = [1 \;\; 0] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{8}$$

Where,

$$A = \begin{bmatrix} \left(1 - \frac{T_s}{\Theta_t}\right) & \frac{T_s}{\Theta_t} \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{T_s K_h}{\Theta_t} \\ 0 \end{bmatrix}, \quad C == [1 \;\; 0]$$

The obtained discretized model is used as a foundation while implementing MPC.

The detail hardware description of the real physical process of an air heater system is provided in (F. Haugen, Fjelddalen, Dunia, & Edgar, 2007)

3.1.1 Time Delay

As observed in the process model of the air heater system in equation(1), there is a presence of time delay in it. Therefore, it makes sense to implement the time delay in the simulated process to have a simulated process identical to the real physical process. While simulating both the MPC controllers, time delay of 0.2 seconds is included at the control input, as $u(t) = u(t - \Theta_d)$, Θ_d being the time delay.

Time Delay can be formulated in different ways to implement it. Two separate formulations have been implemented while implementing linear and non-linear MPC.

3.1.1.1Modeling of Time Delay

Considering a discrete time state space model as in equation (9) and (10),

$$x_{k+1} = Ax_k + Bu_k \tag{9}$$

$$y_k^- = Dx_k + Eu_k \tag{10}$$

If there is no time delay in the system, the actual output, $y_k = y_k^-$. However, with the presence of time delay(Θ_d) in the system,

$y_{k+\Theta_d} = y_k^-$

The algorithm for implementing time delay of a signal for Linear MPC is illustrated in (Ruscio, 2012a), which explains:

With a vector $y_k^- \in \mathbb{R}^m$, a time delay of the elements in the vector y_k^- of $n\theta_d$ samples may be implemented by using a delay matrix x of size $n\theta_d$ X m. Then, at each sampling instant k,

1.Put y_k^- in the first row (at the top) of the matrix x.

2. Interchange each row (elements) in a matrix one position down in the matrix.

3. The delayed output is taken from the bottom element (last row) in the delayed matrix x.

Different formulation for implementing time delay is used for non-linear MPC, with developing a state space model for time delay. The formulation procedure is illustrated in (Ruscio, 2012a) . According to (Ruscio, 2012a), Considering a state space model where $y_k^- \in \mathbb{R}^m$ is delayed by T time instants (here, T is used instead of θ_d for convenience).

The state space model then can be constructed as,

$$\begin{bmatrix} x^1 \\ x^2 \\ x^3 \\ . \\ . \\ . \\ x^T \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & 0 & 0 & . & . & . & 0 & 0 \\ I & 0 & 0 & . & . & . & 0 & 0 \\ 0 & I & 0 & . & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & I & 0 \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \\ x^3 \\ . \\ . \\ . \\ x^T \end{bmatrix}_k + \begin{bmatrix} I \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} y_k^-$$

$$y_k = [0 \quad 0 \quad 0 \quad . \quad . \quad . \quad I] \begin{bmatrix} x \\ x^1 \\ x^2 \\ . \\ . \\ . \\ x^{T-1} \\ x^T \end{bmatrix}_k$$

The constructed state space model can be written as:

$$x_{k+1}{}^{T} = A^{T}x_{k}{}^{T} + B^{T}y_{k}{}^{-} \tag{11}$$

$$y_{k} = D^{T}x_{k}{}^{T} \tag{12}$$

Where,

$x_{k}{}^{T} \in \mathbb{R}^{Tm}$, and the initial state for the delay state is put to $x_{o}{}^{T} = 0.$

Combining equation (11) and (12) with a discrete time state space model described in equation (9) and (10) represents the system without delay from u_{k} to $y_{k}{}^{-}$, and for the delay from $y_{k}{}^{-}$ to y_{k}(Ruscio, 2012a).

The algorithm suggested for modeling of time delay is used in MATLAB. Any of the suggested formulation of time delay can be used, but to have an understanding of time delay modeling, both algorithms are chosen separately for implementation of linear and non-linear MPC

4 State Estimation

4.1 Introduction

Kalman filter as state estimator is used to estimate the state variables that are subjected to random disturbances and measurement noise. The system, which states are to be estimated, must be Observable for Kalman filter to give good results. The system that has been used in this thesis is found to be Observable, which has been tested using "obsv" function in MATLAB. Kalman filter is also used for optimal filtering of noisy measurements. It is applicable for both continuous and discrete time models/ systems.

The Kalman filter is a state estimator, which produces an optimal estimate ($\hat{x}_k$) in a minimum variance sense, such that the variance of the estimation error $\Delta x_k = x_k - \hat{x}_k$ is minimized, i.e.

$\hat{X} = E((x_k - \hat{x}_k)(x - \hat{x})^T)$ is minimized.

The block diagram of Kalman filter is shown in Figure 2.

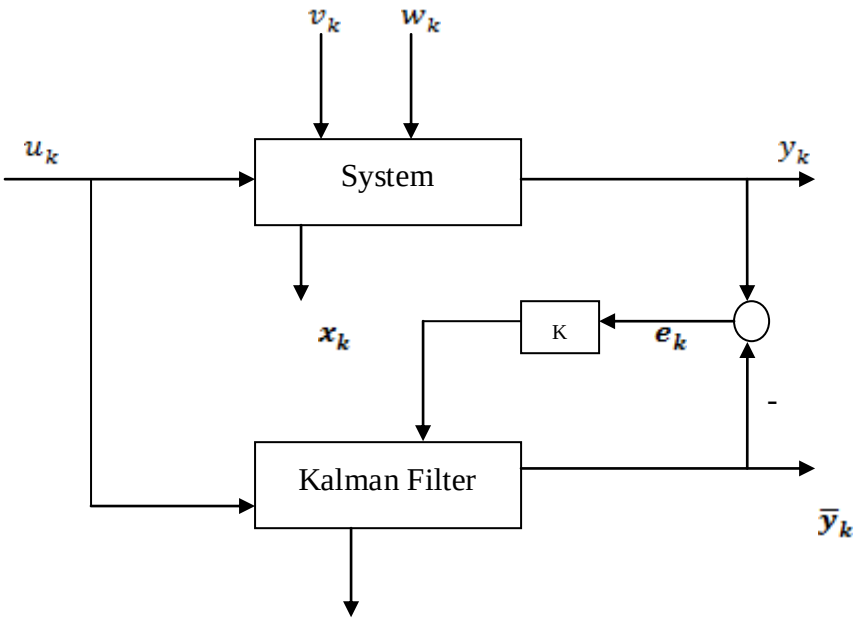

Figure 2:Block Diagram of Kalman Filter

Figure 2 represents the basic block diagram of Kalman filter where u_k represents control input, y_k represents output that are both known. v_k and w_k represent system and measurement noise respectively. x_k represents states that are not measured and needs to be estimated.

4.2 Extended Kalman Filter Algorithm

Extended Kalman filter is employed to estimate the state variables of a dynamic system. The algorithm used for the Extended Kalman filter is described in steps in (F. Haugen, 2012) as:

Step 0: Initial values (at k=0)

Specify the aposteriori state vector ,

$$x_{apost}(t_o) = E(x_o)$$

Specify Covariance of estimation error matrix

$$P_{apost}(t_o) = E\begin{cases}[x(t_o) - x_{apost}(t_o))] \\ [x(t_o) - x_{apost}(t_o)]^T\end{cases}$$ for time steps k=1,2,3,…

Step 1: Time Updates

Apriori estimate(Predicted estimate)

$$x_{apost}(t_k) = f[x_{apost}(t_{k-1}), u(t_{k-1})]$$

$$= x_{apost}(t_{k-1}) + T_s f_c(x_{apost}(t_{k-1}), u(t_{k-1})]$$

Covariance of estimation error

$$P_{apri}(t_k) = AP_{apost}(t_{k-1})A^T + Q$$

Step 2: Measurement Updates

Kalman Filter gain

$$K(t_k) = P_{apri}C'(CP_{apri}(t_k)C' + R)^{-1}$$

Measurement estimate

$$y_{apri}(t_k) = g(x_{apri}(t_k))$$

(c) State estimate (Aposteriori estimate, or corrected estimate), which is applied as the estimate x_e :

$$x_e(t_k) = x_{apost}(t_k) = x_{apri}(t_k) + K(t_k)(y(t_k) - y_{apri}(t_k))$$

(d) Covariance of estimation error

$$P_{apost}(t_k) = [I - K(t_k)C]P_{apri}(t_k)$$

The described algorithm is implemented in MATLAB for state estimation, particularly to estimate ambient temperature.

The response after implementing Kalman filter algorithm in MATLAB is shown in Figure 3.

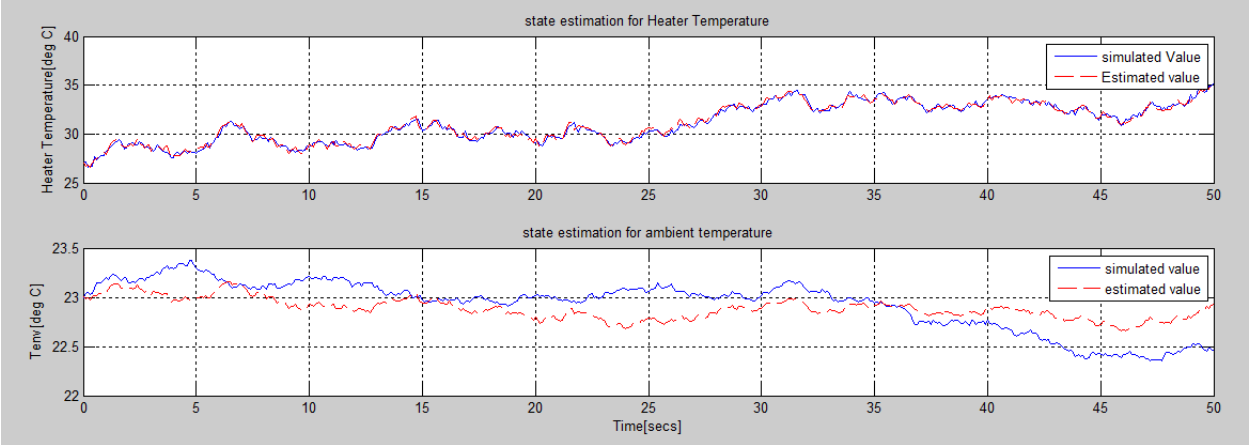

Figure 3: Kalman fitler estimation of the state variables

The estimation error is shown in Figure 4.

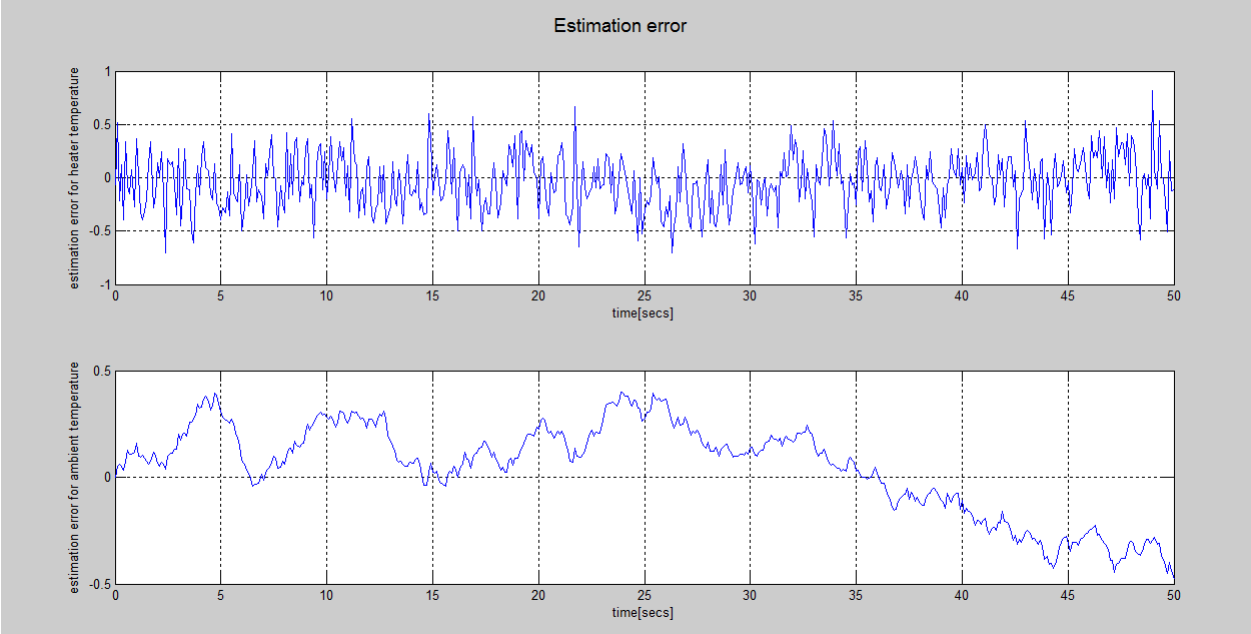

Figure 4: Estimation error

The temperature at the outlet of the air heater system is measurable. However, the ambient temperature is not measurable, for which Kalman filter as an estimator is implemented. The estimation error plot in figure 4 shows that the error in estimation is very less. The Kalman filter estimation is used to calculate the present state while implementing MPC controller.

4.3 Tuning of Kalman Filter

It is essential to fine tune the Kalman filter when it is employed in the real system(Filter, 1960). It is required to tune the process disturbance auto covariance Q, and measurement noise auto covariance R. In addition, it makes sense to select steady state point for $x_{apost}(t=t_o)$ as the initial value for the kalman filter for the tuning purpose. Here, in the proposed system there is only one output; therefore, R has dimension 1 X 1. The value of R is relatively easy to calculate from time series of measurements, but adjusting Q is of more important(Filter, 1960). The value of R is calculated to be 0.09.

The tuning of the parameter Q is of major concern as it has a direct influence on the estimation of states. The estimates are affected by process noise. In order to reduce the noise in estimates, the tuning of Q is essential. Larger the value of Q, large is the Kalman gain and less noisy is the estimates. However, it should be remembered that if Q is increased beyond a certain limit, there's a possibility that the estimates can become more noisy. Therefore, the idea is to chose Q as large as possible without the states being too noisy(Filter, 1960).

Q has dimension 2X2 since there are two states in the system and Q is a diagonal matrix having same dimension as that of the system matrix. The value of Q calculated for the tuning purpose is, Q=diag ([0.0729 0.0004]).

During the simulation, some test is performed with increasing the process noise, and it is observed that it leads to more noisy measurements (graph is not shown here). With increasing value of Q, it is observed that the estimated value is close to the assumed true value at a faster rate but with more noise in the estimates. On the contrary, the estimation is less noisy with decreasing Q but it takes a longer time for estimated value to come close to the assumed true value.

5 Model Predictive Control

5.1 Introduction

As discussed in the section 1.1, Model Predictive control is a class of advanced control strategy that makes the controller take action before any changes occur in the set point. In MPC, Process model is used to predict the future values of output over a defined Prediction Horizon. With constraints on input and output variables, process model and disturbance, current control input and future inputs for remaining sampling instants are calculated by solving the optimization problem over a specified horizon. However, only the first element from the sequence of the control inputs that are calculated is implemented in the process. The same process is repeated for next sampling instant.MPC algorithm consists of:

1. Performance Index(Objective function)
2. Constraints
3. Process Model

5.1.1 Performance Index

Performance Index, or a cost function J_i is the criteria for measuring the performance of control. It is a scalar criterion that is employed to measure, for instance, the difference between future outputs and some future reference, and thus giving the information about the behavior of the control input(Ruscio, 2012a). The idea of using performance index while implementing MPC controller is that the MPC controller computes sequence of future control actions such that the cost function is minimized. Although, a sequence of control actions are computed, only the first control vector, u_k is feed to the process.

The performance Index chosen while implementing the MPC controller in the air heater system is given by equation (13):

$$J_i = \sum_{i=1}^{L} e_i{}^T Q_i e_i + \Delta u_{i-1}{}^T P_{i-1} \Delta u_{i-1} \tag{13}$$

Where,

$\mathbf{e_i} \triangleq \mathbf{y_i} - \mathbf{r_i}$

$\mathbf{y_i}$ is the Output value.

$\mathbf{r_i}$ is a reference value that we want the $\mathbf{y_i}$ to follow.

$\mathbf{L}$ represents Prediction Horizon

Δu represents predicted change in control value

Let us assume at current time, k=0 we desire to compute control action uo with the information of the current state xo. Here, Performance Index can be a criterion to have knowledge of the control action, i.e. to know how good the control action is.

The idea here is to design a controller such that the output value tracks the reference value in an optimal manner, for which we want u_i to be small. Q and P in the defined Performance Index are the weighting function. Q represents output error weight matrix and P represents control weight matrix. It is desired to have $Q_i > 0$, $P_i \geq 0$. The performance Index J is small when e_i is small for which little input is required. It leads to the fact that the performance is good for small value of J. However, for poor control, the control deviation is large and when the control signal is large, the value of J is large thus indicating poor performance.

5.1.2 Constraints

The motivation of using MPC is its ability to handle process constraints as all physical systems have constraints. Common constraints such as input amplitude constraints and input rate of change of constraints can be handled in far more efficient manner. The MPC controller takes account into the process constraints while calculating the future control actions. While implementing MPC controller in the proposed air heater system, input constraint is taken into consideration.

5.1.3 Process Model

As discussed in section 2.1, MPC is dependent on the accuracy of the model, which gives information about input and output behavior of the process. According to (Ruscio, 2012a), simple data driven linear models are used for MPC implementation. (Ruscio, 2012a) has further suggested to use the models identified by subspace Identification method, which leads to fast implementation of MPC.

5.2 Advantages of Using MPC

There are various advantages of using MPC. Some of them can be listed as:

1. It takes account of cross Coupling in MIMO systems in an optimal way, for which it is preferable to use MPC for a process with the large number of manipulated and Controlled Variable(Ruscio, 2012a).

2. MPC offers flexibility of computing control suggestion (not the feedback of the control to the process) to the process operator, which is not applicable in conventional control system (Ruscio, 2012a).

3. MPC takes process inputs and output constraints into consideration.

5.3 MPC Implementation

The block diagram of the Optimization and control system as in Figure 5 illustrates the implementation of MPC during the thesis.

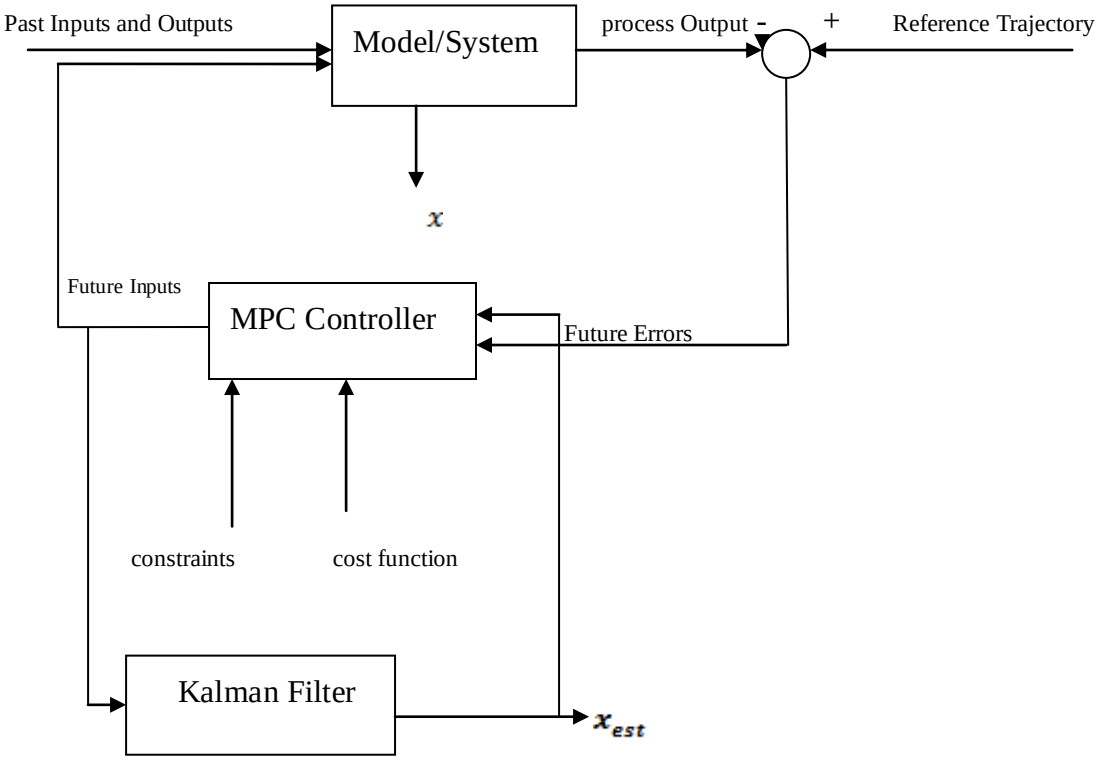

Figure 5: Block Diagram of Optimization and Control system

Figure 5 shows the structure for implementing MPC controller in the process or system model. Two cases for MPC are simulated, one where the states are updated by the simulator, and the other when present states are calculated with the Kalman filter.

5.3.1 Implementation of Linear MPC

To implement linear Model predictive control, a linear Quadratic Optimal controller of velocity form having integral action is presented as in (Ruscio, 2012a).

Given a process model,

$$x_{k+1} = Ax_k + Bu_k + v \tag{14}$$

$$y_k = Dx_k + w \tag{15}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^r$ is the control input vector , $y_k \in \mathbb{R}^m$ is the output measurement vector, and A, B , D represents the system matrices, where

$A = \begin{bmatrix} \left(1 - \frac{T_s}{\theta_t}\right) & \frac{T_s}{\theta_t} \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} \frac{T_s K_h}{\theta_t} \\ 0 \end{bmatrix}$, $D = [1 \;\; 0]$ as obtained in section 3.1.

Here, v and w are both unknowns, where v is an unknown constant or slowly varying process disturbance, and w represents an unknown constant or slowly varying measurements noise vector.

The objective is to study MPC controller subjected to the performance index defined as in equation (13).

From the state equation (14),

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k \tag{16}$$

Where, $\Delta x_{k+1} = x_k - x_{k-1}$

And, $\Delta u_k = u_k - u_{k-1}$

From the measurement equation as in (15),

$$y_k = y_{k-1} + D\,\Delta x_k \tag{17}$$

Augmenting the model equation (16) and equation (17), we get

$$\begin{bmatrix} \Delta x_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} A & 0_{nxm} \\ D & I_{mxm} \end{bmatrix} \begin{bmatrix} \Delta x_k \\ y_{k-1} \end{bmatrix} + \begin{bmatrix} B \\ y_{k-1} \end{bmatrix} \Delta u_k$$

And,

$$y_k = [D \quad I_{mxm}] \begin{bmatrix} \Delta x_k \\ y_{k-1} \end{bmatrix}$$

Therefore, the state space model can now be represented as,

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}\Delta u_k \quad (18)$$

$$y_k = \tilde{D}\tilde{x}_k \quad (19)$$

Where,

$$\tilde{A} = \begin{bmatrix} A & 0_{nxm} \\ D & I_{mxm} \end{bmatrix},\ \tilde{B} = \begin{bmatrix} B \\ y_{k-1} \end{bmatrix} \text{ and } \tilde{D} = [D \quad I_{mxm}]$$

The state space model as in equation (18) and equation (19) can be used to define a prediction model of the form,

$$y_{k+1/L} = p_L + F_L \Delta u_{k/L}$$

Where,

$$p_L = O_L \tilde{A}\tilde{x}_k$$

$$F_L = \left[O_L \tilde{B} \quad H_L{}^d\right]$$

Where,

O_L is the extended observability matrix of the pair $\tilde{A}, \tilde{D}$ and H_L, which represents the Toepliz matrix of impulse response matrices $\tilde{D}\tilde{A}^{i-1}\tilde{B} \in \mathbb{R}^{mL\,x\,(L-1)r}$

The performance Index can thus be written as,

$$J_k = \Delta u_{k/L}{}^T H \Delta u_{k/L} + 2f^T \Delta u_{k/L} + J_o$$

Where,

$$H = F_L{}^T Q F_L + P$$

$$f = f_L{}^T Q(p_L - r_{k+1/L})$$

Then the MPC control is given by,

$$\Delta u_{k/L} = -H^{-1} f_k \quad (20)$$

The MPC control given in equation (20) is for the Unconstrained MPC control. Similar formulation for unconstrained MPC control is also presented in (Ruscio, 2012b).

However, we desire to use a real physical process, and with the input constraint, Constrained MPC control is needed, which can be obtained by using "quadprog" as an optimization solver in MATLAB. The MPC control is achieved in deviation form. However, the absolute value of the control signal is implemented in the process, as $u_k = u_{k-1} + \Delta u_k$.

For the implementation of constrained MPC control, it is desirable to specify input rate of constraints as,

$$\Delta u_{min} \leq \Delta u_{k/L} \leq \Delta u_{max}$$

And, input amplitude constraints,

$$u_{min} \leq u_{k/L} \leq u_{max}$$

Using the Relationship,

$$u_{k/L} = S\Delta u_{k/L} + cu_{k-1}$$ (Ruscio, 2012a)

The constraints can be written as linear matrix inequality,

$$A\Delta u_{k/L} \leq b_k$$

Where,

$$A = \begin{bmatrix} I \\ -I \\ S \\ -S \end{bmatrix}$$

And,

$$b_k = \begin{bmatrix} \Delta u_{max} \\ -\Delta u_{min} \\ u_{max} - cu_{k-1} \\ -u_{min} + cu_{k-1} \end{bmatrix}$$

The constraints are passed through the arguments in the "quadprog" optimization solver to find the optimal MPC control as

duf= quadprog(H,f,A,bk);

And, the absolute value of the control input is calculated as,

u=u_old+duf(1:nu)

The obtained absolute value of the constrained MPC control is feed to the process to maintain the temperature of the air heater system to a desired reference temperature.

The tuning parameters for the MPC, the output error weight function (Q) is chosen as 1.And, the control weight function P=0.01 is found out by trial and error method in the simulator.

5.3.2 Implementation of Non - Linear MPC

The approach to implement non-linear MPC is presented in (F. A. Haugen, n.d.). To implement Non-Linear MPC, "fmincon" as an optimization solver is used in MATLAB. The Objective of the controller is to determine optimal control action value subjected to the Objective function (13), and constraints

$u_{min} \leq u(t) \leq u_{max}$

For the implementation of the controller, the discretized version of the objective function is minimized to get optimal control input ($u_{optimal}$). The first element of the sequence of $u_{optimal}$ $(t = t_o)$ is applied as the control signal at the present time point. The prediction horizon is receding, and the procedure of obtaining ($u_{optimal}$) and $u_{optimal}$ $(t = t_o)$ is repeated. The differential equations of the air heater model as in equation (1) are discretized using explicit Euler method with Ts=0.1 seconds, with Ts being the sampling time. The same sampling time Ts is also used for the discretization of the objective function J.

There can be different methods to solve optimization problems. Here, non-linear optimization function "fmincon" in MATLAB is used to find a local minimum.

The tuning parameters for the non-linear MPC, i.e. the output error weight function (Q) is chosen as 1, and the control weight function P=0.01 is found out by trial and error method in the simulator.

6 Results

6.1 Simulation

MPC controller is tested in a simulated process before implementing it on the physical process. Here, the objective is to keep the temperature at the outlet of the air heater to a reference temperature. Both linear MPC and non-linear MPC are examined to investigate the difference in the performance of the controller.

During the simulation both constrained and unconstrained MPC control are implemented, while only constrained MPC control is applied while implementing the control action value to the real process.

6.1.1 Linear MPC

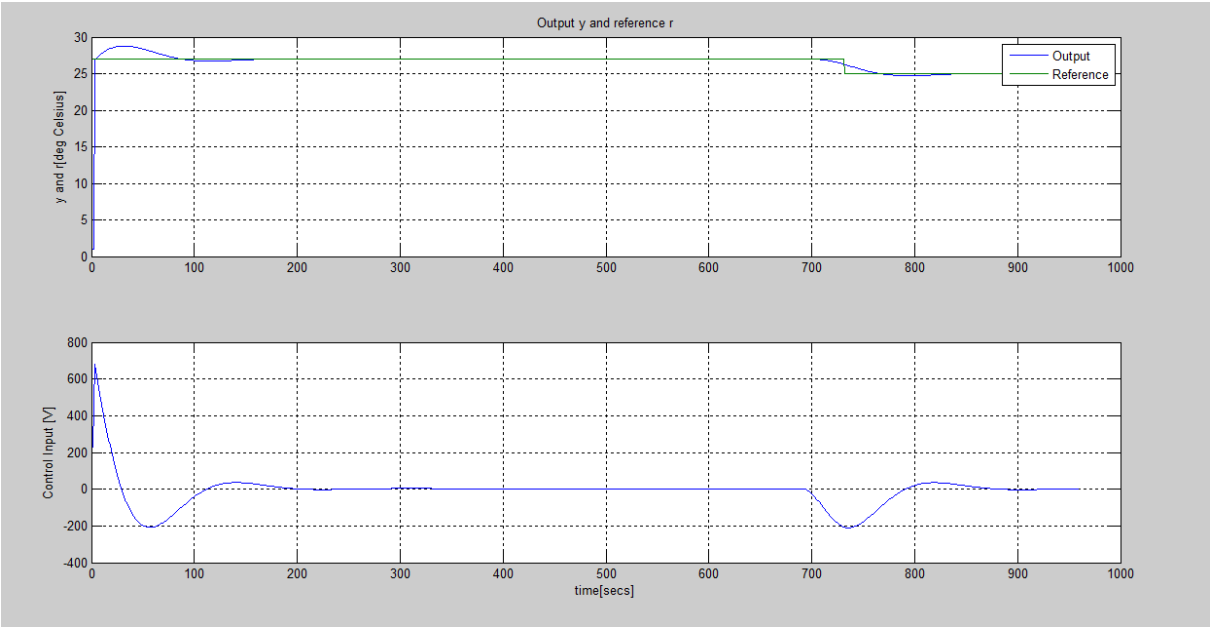

Figure 6: Unconstrained MPC control action for Linear MPC

For simulation purpose in linear MPC, ”prbs” signal is used as a reference signal as shown in Figure 6. Reference temperature signal is varied between 27 deg Celsius and 25 degree Celsius, and the result shows that the output temperature tracks the varying reference temperature, with the integral action.

Figure 6 represents Unconstrained MPC control action. However, the input range is practically infeasible when the control signal has to be implemented in the real process due to the constraints in the input from (0-5) V. Therefore, it makes sense to use constrained MPC control in the physical process. The constrained MPC control is also simulated and the simulated response for constrained MPC is shown in Figure 7.

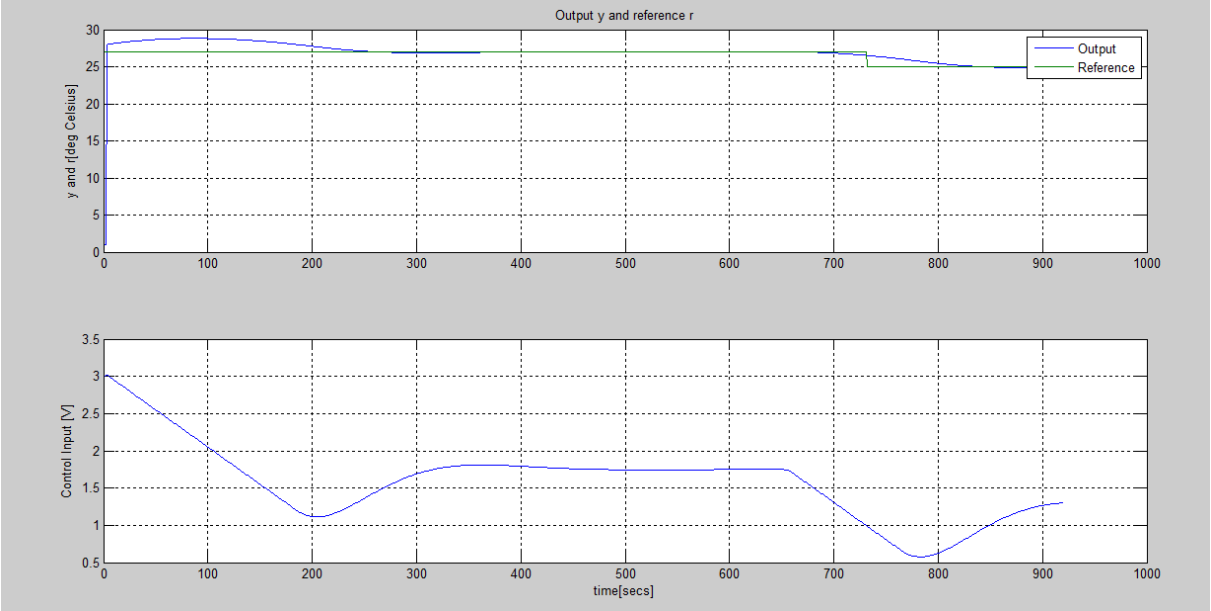

Figure 7:Constrained MPC Control action for Linear MPC

Like in Unconstrained MPC control for linear MPC, the reference temperature is varied between 27 and 25 deg C. The response in Figure 7 shows that the output temperature tracks the varying reference temperature similar with the unconstrained MPC control. The range of MPC control action value is within the range of (0-5) V as shown by the response in Figure 7, indicating that it does not violate the input constraint limit.

MPC control action indicates that when the temperature at the outlet of the air heater goes above the reference temperature, the control input is decreased (the fan makes more airflow into a tube) to take output temperature to the reference temperature level. Also, when the output temperature goes below the reference temperature, the control input rises (the fan makes less air flow in the tube) again to take the output temperature to the reference temperature level. In addition, it is observed that the controller starts to take action before any changes in the reference signal, which is an important feature of MPC controller.

The simulation is also performed for linear MPC where the states are updated by kalman filter. The simulated response for Unconstrained linear MPC control where the states are updated by kalman filter is shown in figure 8.

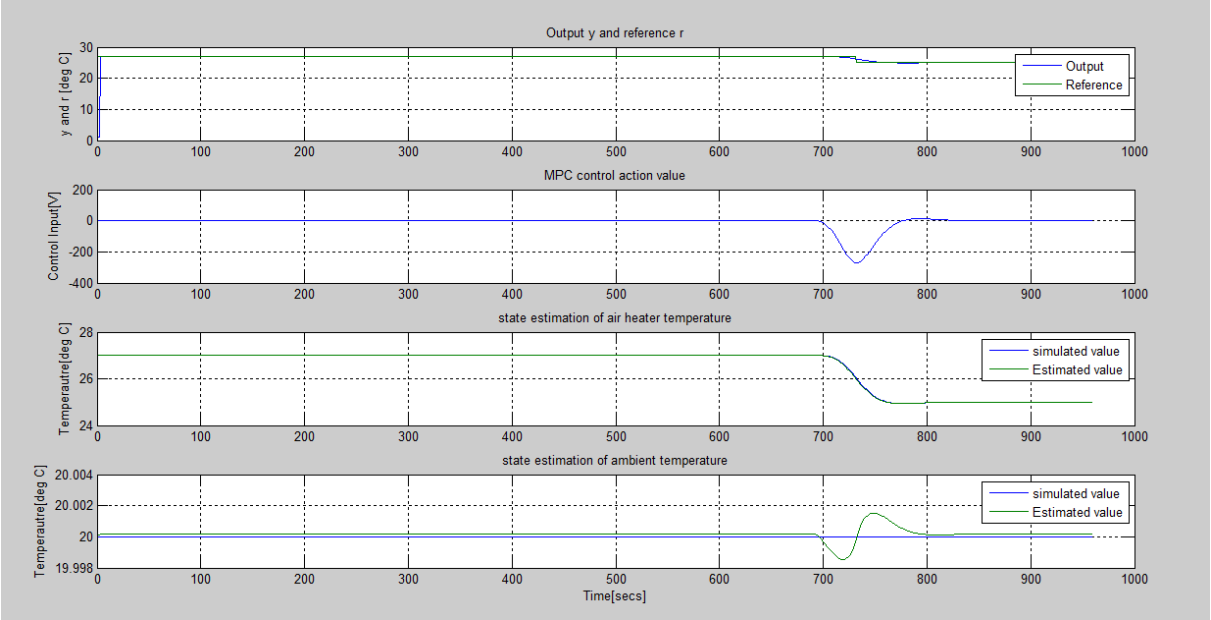

Figure 8:Unconstrained linear MPC control where states are updated by kalman Filter

Figure 8 shows that the estimation error, mainly for ambient temperature is very less, and the controller seems to work fine when the states are updated with the kalman filter, as the output temperature follows the varying reference temperature. The constrained linear MPC control is also simulated to ensure that the control action value lies within the input bounds. The response for constrained linear MPC control where the states are updated by kalman filter is shown in Figure 9.

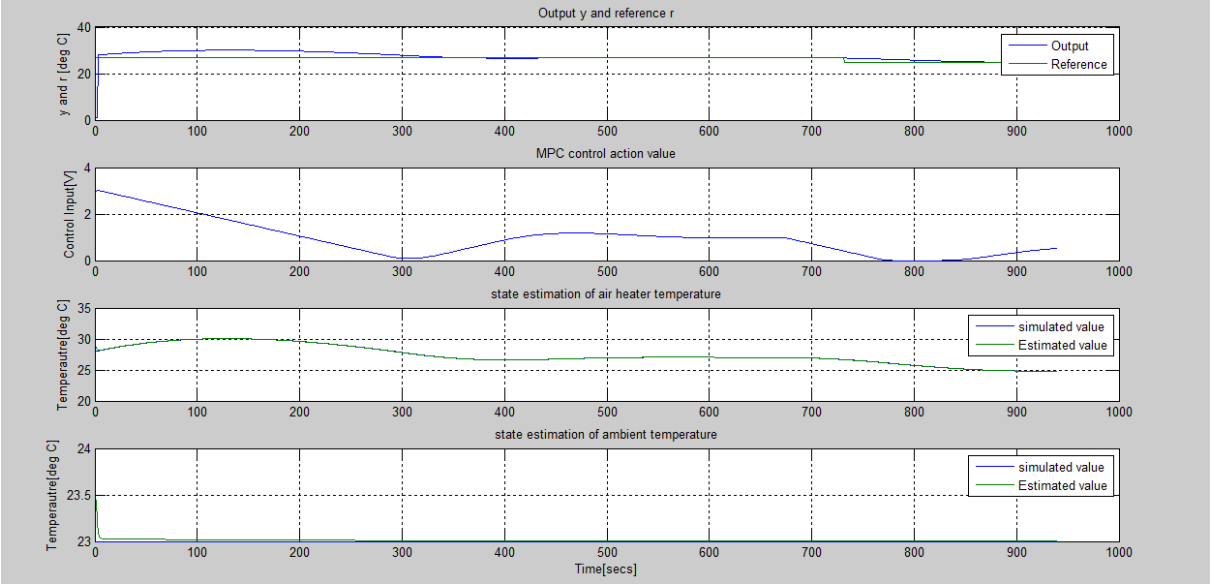

Figure 9:Constrained linear MPC control where states are updated by kalman Filter

6.1.2 Non-Linear MPC

To implement non-linear MPC in the simulated process, "fmincon" as an optimization solver is used in MATLAB. The response is simulated without constraints in the input (Unconstrained MPC control) and with constraints in the input (Constrained MPC control). The simulated response for Unconstrained MPC control action for non-linear MPC is shown in Figure 10.

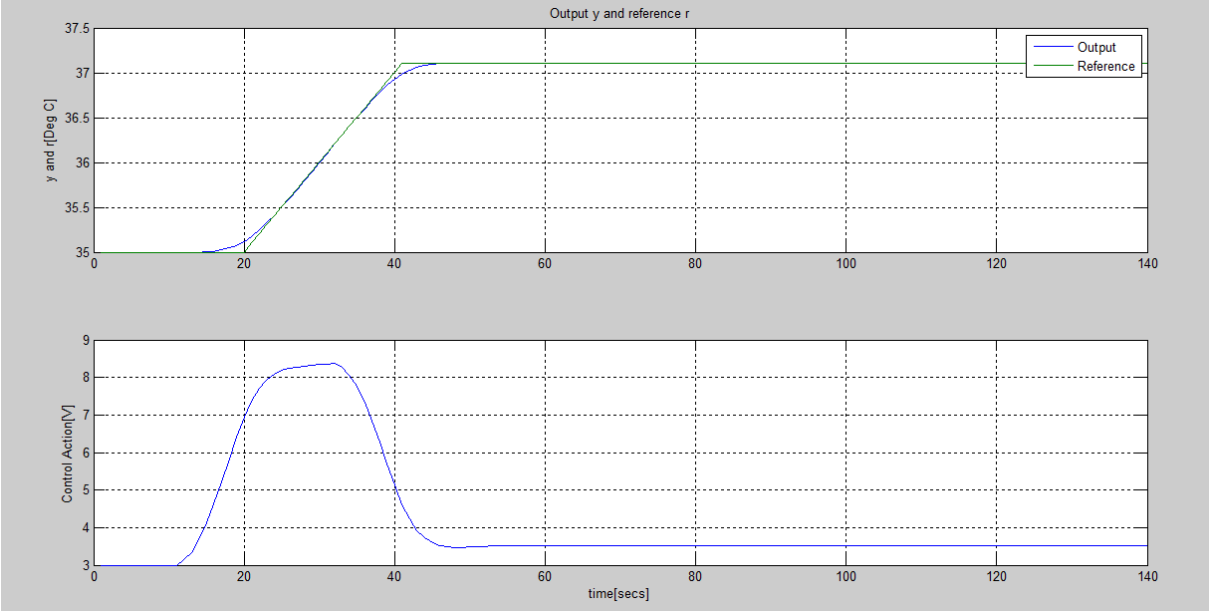

Figure 10:Unconstrained MPC control action for Non-Linear MPC

The reference temperature is set to 35 to 37 degree Celsius. The response as in Figure 10 indicates that the output temperature tracks the reference temperature. Figure 10 also shows the MPC control action, but here also due to unconstrained MPC control, the input range violates the input constraint limit (0-5V).

Therefore, Constrained MPC is designed to ensure the input is within the range of (0-5)V. The response for constrained MPC control is shown in Figure 11.

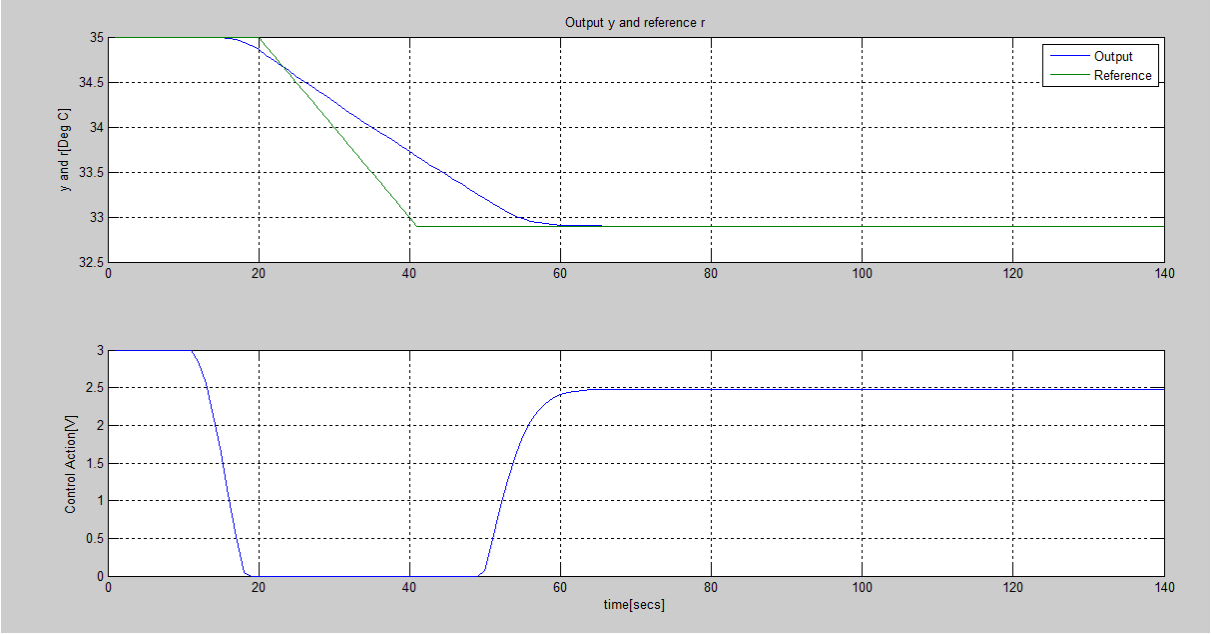

Figure 11:Constrained MPC control for non-linear MPC

Figure 11 shows that the output temperature tracks the desired temperature reference, which is varied between 35 to 33 degree Celsius. The MPC control action in Figure 11 shows that there is no violation of input constraints limit.

The MPC controller response of non-linear MPC also shows that control input has to decrease (that is the fan makes less air to flow into a tube) to bring the temperature down to the reference temperature. With similar behavior, we can assume that if the output temperature has to rise to catch up reference temperature, the control input also increases making fan blow more air into the tube to make the output temperature track the desired reference temperature. In addition, as in the response of linear MPC, the non-linear MPC controller response also indicates the fact that controller starts to take action earlier when there is the change in the reference signal.

The simulaton is also performed where the present state that is needed for the prediction is calculated using Kalman filter. The response of Unconstrained non-linear MPC controller where the states are updated with kalman filter is shown in Figure 12.

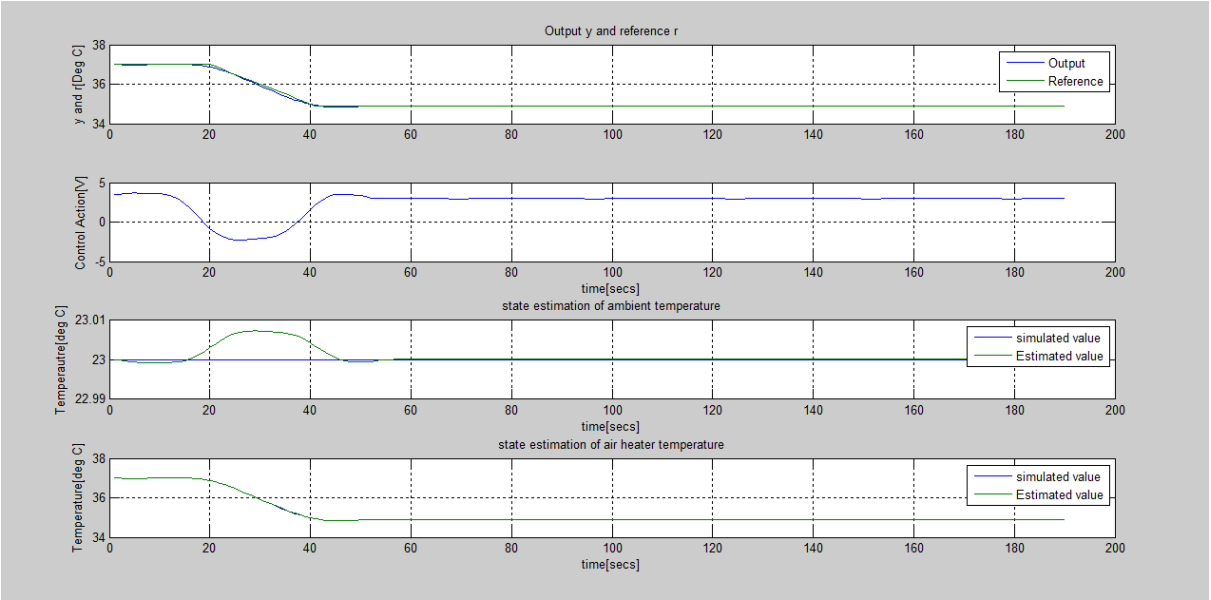

Figure 12: Unconstrained non-linear MPC control where states are updated by kalman Filter

Figure 12 shows the estimation of the air heater and ambient temperature that has been estimated with the use of Kalman filter. The simulated ambient temperature is kept at 23 degree Celsius. The response shows that the kalman filter estimation error is very less, and the controller seems to work fine when the states are updated with the kalman filter. The result shown in Figure 12 is for Unconstrained MPC control, due to which the control signal is not within the input constraint limit.

The response for Constrained MPC control with the use of Kalman filter to update the states is shown in Figure 13, which shows the MPC control action value is within the limit, i.e. (0-5)V.

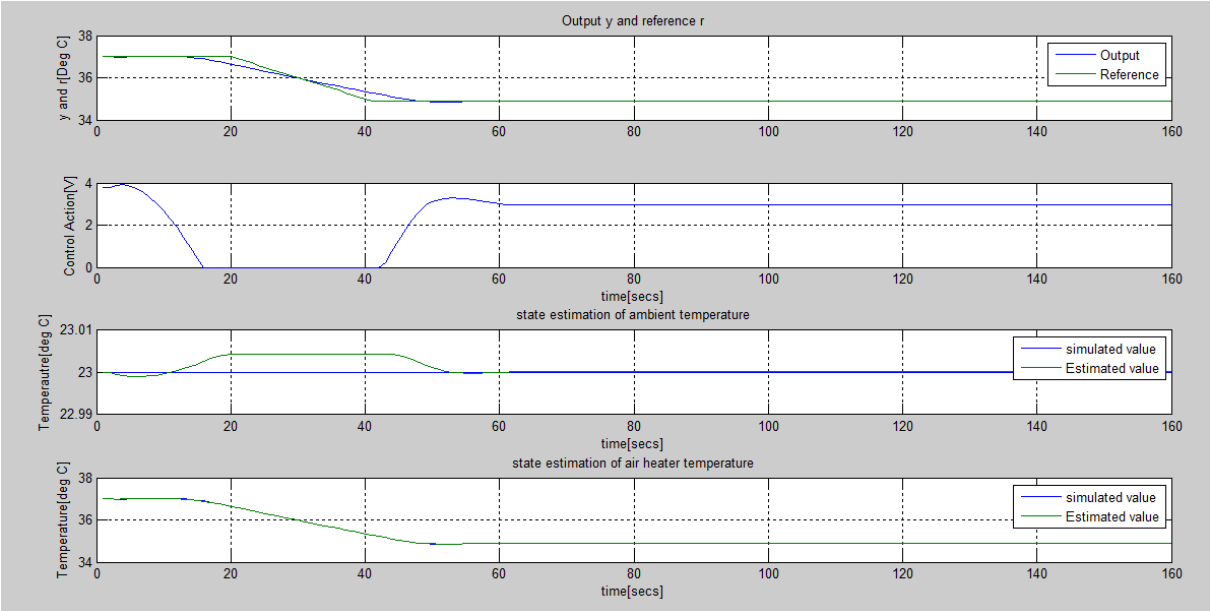

Figure 13:Constrained non-linear MPC control where states are updated by kalman Filter

With the MPC controller response of both the linear and non-linear MPC, it is observed that though the air heater model is non -linear (assumed due to the presence of time delay in it), the behavior shown by it is linear in nature.

7 Discussion

The controller Performance is investigated in terms of stability Margins, Set Point tracking, Horizon length and Computational time. The discussion is based on Unconstrained MPC control.

7.1 Stability Margins

The delay time that has been assigned to the simulated system is 0.2 seconds. One method to observe the Robustness of the controller can be increasing the loop time delay and observe the controller performance against model error. For the analysis purpose, loop time delay is increased and the controller performance of both the linear and non -linear MPC is observed.

The idea of Gain Margin and Phase Margin can be used to measure the relative stability and robustness of the controller(Dale R. Seborg, Thomas F. Edgar, n.d.).

To calculate the Phase Margin, value of additional time delay (T_{add}) that makes the control system to the stability limit needs to be determined. The phase Margin is then given by,

$\mathrm{PM[deg]} = 360° \frac{T_{add}}{P_u}$ (F. A. Haugen, n.d.)

Where,

P_u represents the period of Oscillations.

To find the sustained oscillation for linear MPC, time delay loop has been increased and the corresponding value of T_{add} that makes the control system to the stability limit has been used to calculate the Phase Margin. The control system response for linear MPC after being subjected to sustained oscillations is shown in Figure 14.

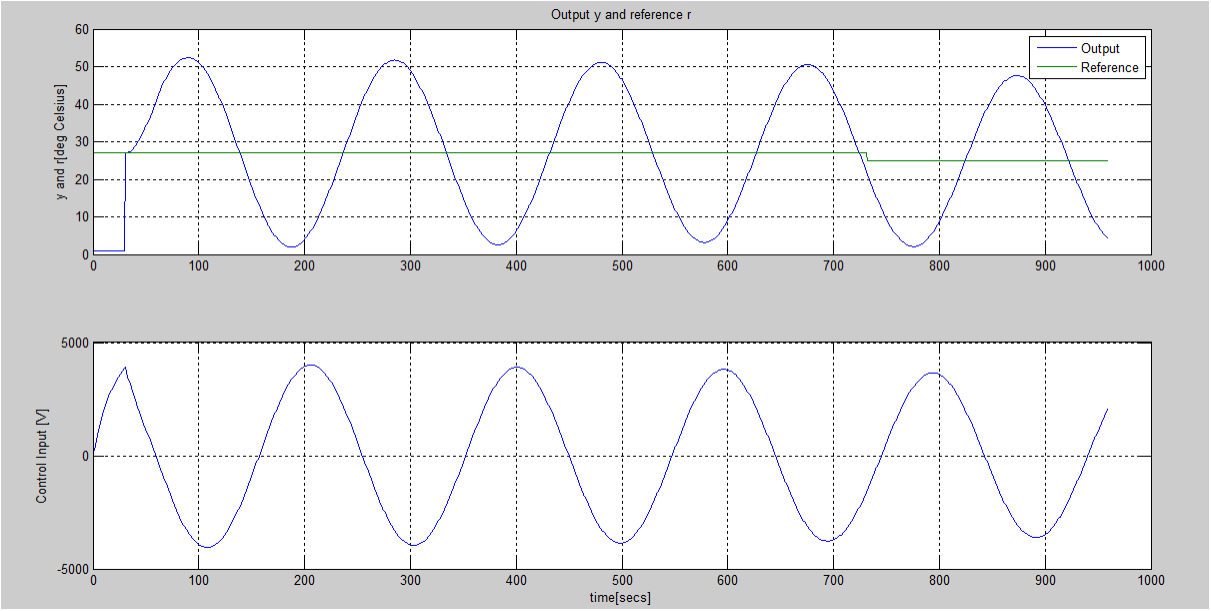

Figure 14:Control system subjected to sustained oscillations after increasing loop time delay for Linear MPC

With the value of T_{add} and period of oscillation P_u, the Phase Margin for the linear MPC is calculated to be 5.48^o.

The similar process of calculating Phase Margin is employed in case of non- linear MPC, and the Phase Margin for non-linear MPC is calculated to be 7.38^o.

The sustained oscillations obtained for the control system while implementing non-linear MPC is shown in Figure 15.

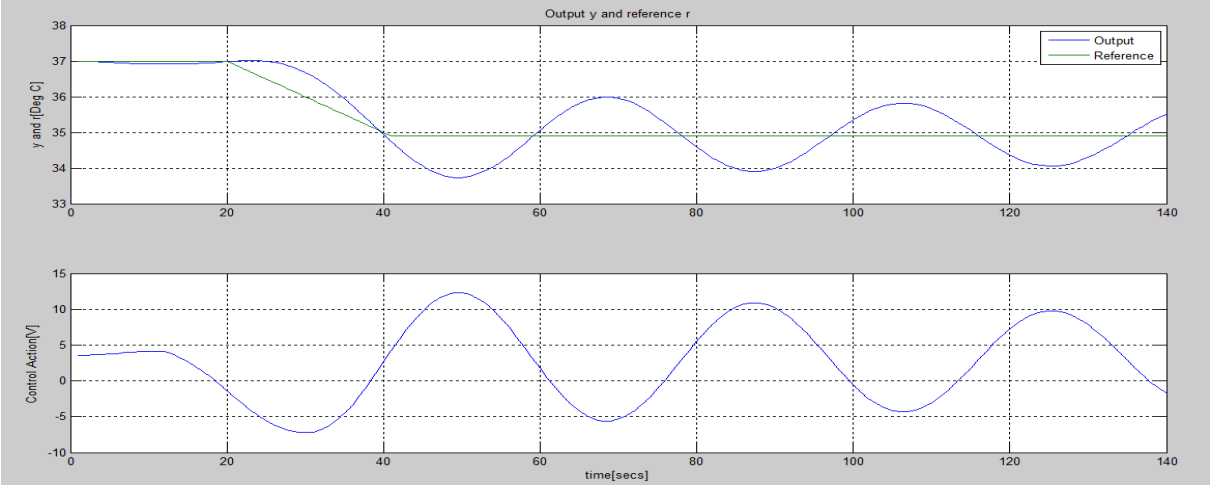

Figure 15:Control system subjected to sustained oscillations after increasing loop time delay for non-linear MPC

Phase Margin is marginally larger for non-linear MPC than that of linear MPC, which shows that the linear MPC is more robust to this model change as compared to non-linear MPC. According to (Ruscio, 2012a) there is no guarantee that a non-linear MPC converges with reasonable computing time. In addition, (Ruscio, 2012a) also explains that the non-linear Optimization problem often has the problem with local minima and convergence problems, which makes it less robust. However, according to (Dale R. Seborg, Thomas F. Edgar, n.d.), a well tuned controller should have a Phase Margin between $30°$and $45°$. Therefore, it can be said neither of the MPC controller is robust against this model change, which also highlights the fact that MPC controllers are model based and need a highly accurate model.

To calculate the Gain margin an adjustable gain ΔK is added between the MPC controller and the process.(F. A. Haugen, n.d.) According to (Dale R. Seborg, Thomas F. Edgar, n.d.), a well-tuned controller should have a Gain Margin between 1.7 and 4.0. The ultimate gain value of ΔK needs to be calculated through trial and error method, for the control system to attain stability limit. However, it is noticed that with the increase in loop gain in either of the MPC (both linear and non-linear), the controller is not subjected to oscillations having constant magnitude. Therefore, it can be assumed that the Gain Margin for the Model predicative controller is infinite.

7.2 Set Point Tracking

Integral of Absolute Error (IAE) is used as a criterion to measure the set point tracking.
Integral of Absolute Error is calculated as:

$$\mathrm{IAE} = \int_{t_i}^{t_f} |e| \, dt \tag{21}$$

Where, $|e|$ represents the absolute value of error

t_i and t_f represents the initial and final time respectively.

For, non-linear MPC, IAE using the relation as in (21) is calculated to be 0.1009, while for linear MPC, IAE computed with relation (21) is 0.42. Therefore, non-linear MPC could be said better in terms of IAE that means response of non-linear MPC tracks the reference in more efficient way as compared to the response of linear MPC.

7.3 Horizon Length

It is observed during simulation that the non-linear MPC can give good, if not better performance with shorter Prediction Horizon length as compared to linear MPC, for which longer horizon length is required.

7.4 Computational time

As it is discussed in section 7.3 that different horizon length is used for linear and non-linear MPC, it is difficult to have a conclusion on computational time for both the MPC. However, just to observe the computational time, same horizon length is used for both the MPC, and it is observed that linear MPC has a faster computational time than that of non-linear MPC. Nevertheless, it should be noted that non-linear MPC gives better results for shorter Horizon length.

Also, during simulations of both the MPC it is noticed that higher value of control weight function(Q) in performance index (13) would result in slow tracking of the reference temperature by the output temperature, while smaller value of Q would result in faster control. In addition, it is observed during the simulation that longer Prediction Horizon gives more stable response.

8 Practical Results

The developed controller for the linear and non-Linear MPC is tested on the physical process. MATLAB is used for the data acquisition purpose. The continuous measurement from the real air heater system is taken, and the MPC control action value is fed to the system for control purpose. The delay that is used in the simulated system is not used in the real process.

As, it has been discussed in the section 6, constrained MPC control is implemented in the real process as input to the DAQ device is limited to 0-5 V.

The response of the controller for the linear MPC is shown in Figure 16.

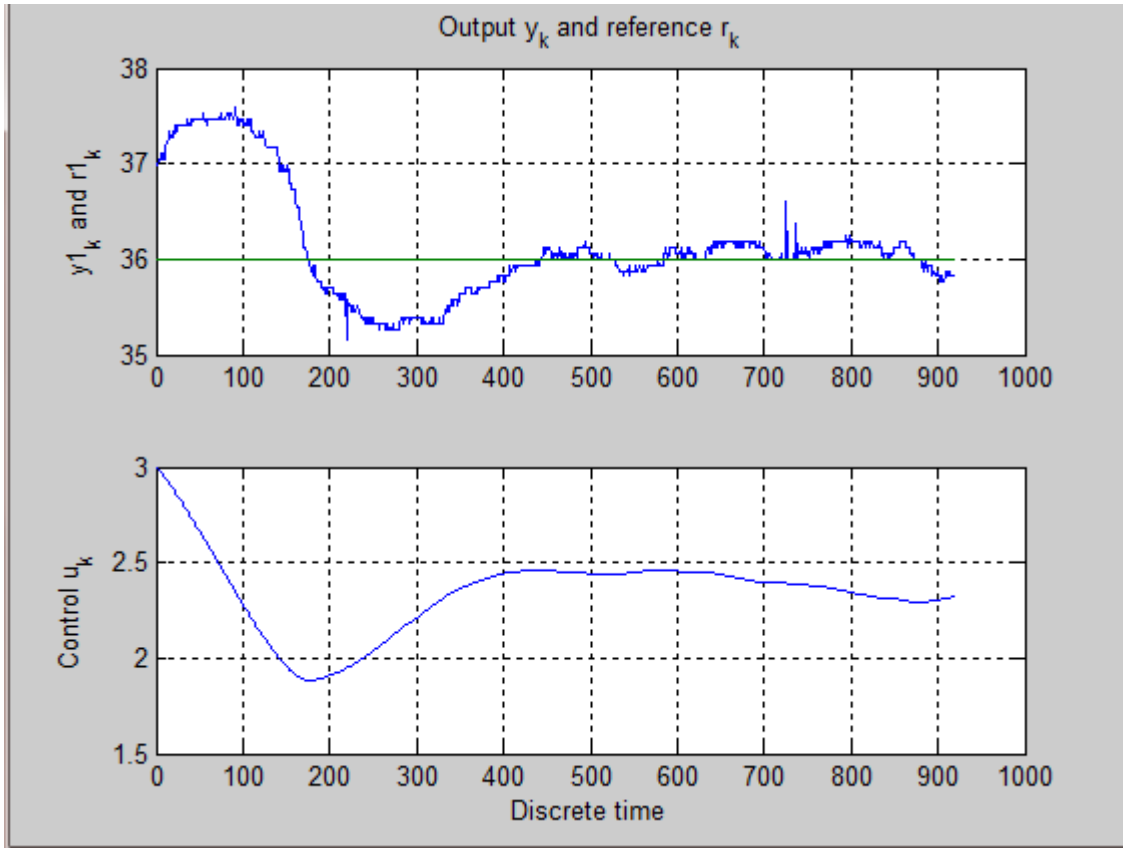

Figure 16: Linear MPC control in phyiscal process

For the implementation of linear MPC in the physical process, the reference is set to be 36 degree Celsius. Figure 14 shows the output temperature of the air heater system tracking the reference temperature. The control action value is also seen in the plot. The control action value is decreasing at the start to take the temperature of the heater system to the reference value. Moreover,, it is also seen that when the temperature drops down below the reference value, the control input increases to take the temperature at the outlet of the air heater to the

reference temperature. After some time, with the optimal input value, the output temperature remains steady following the reference temperature value.

The result tested for non-linear MPC in the physical process was not satisfactory. Therefore only the result for implementation of linear MPC in the real process is presented in the report.

9 Conclusions and Recommendation

9.1 Conclusion

As part of Master thesis, a brief literature on MPC concept and its adoption in process industries is studied. While, MPC may have been popular in process industries, study also suggests that it has encouraged many research enthusiasts to look upon it for their research activities. The main task carried out while thesis is to develop linear and non-linear MPC and investigate the controller performance when applied to a process model/process.

At the outset, the continuous time state space mathematical model of the air heater system is deduced into discrete time state space form. The discretized form of the air heater model is used as a foundation for implementing MPC controller. LQ Optimal control of incremental form is used to implement linear MPC while non-linear optimization function "fmincon" is used for the optimization purpose to implement the non-linear MPC. The controller performance of both linear and non-linear MPC has been compared in terms of stability margins, set point tracking, Horizon length and computational time.

In order to analyze stability margins, Phase Margin and Gain Margin for the controllers are calculated. To calculate Phase margin, an additional loop delay is added between the controller and the process. The loop time delay is increased to the point where the control system attains the stability limit, i.e. the control system is subjected to the sustained oscillations. The Phase Margin of the non-linear MPC is found to be marginally larger than that of linear MPC. However, the result has shown that neither of the MPC controller is that robust against this model change, as the phase Margin calculated for both the controllers are not within the acceptable range. Therefore, a conclusion can be drawn that MPC controllers are not that robust as it is highly model based and depends on the accuracy of the model.

With similar approach, GM for the control system is calculated, where an additional loop gain is added between the controller and the process. In both the controllers, it is noticed that it is highly improbable for control system to attain stability limit with increase in loop gain. Therefore, it can be assumed that the Gain margin for both the controllers is infinite, and the concept of Gain Margin in MPC controllers is uncertain.

IAE is used as a measure for the set point tracking by the controller. The result shows that the non-linear MPC tracks the set point more efficiently than the linear MPC. It is measured by computing IAE, i.e. sum of the absolute value of the error over the entire length of the simulation horizon. The value of IAE computed for the non-linear MPC is found to be less than that of linear MPC, which shows that non-linear MPC is better than linear MPC.

It is also observed that non-linear MPC uses shorter horizon length than linear MPC to give good, if not better results. However, the computational time required by non-linear MPC is found to be higher, provided that same horizon length is used for the control operation in both the MPC.

The ambient temperature influences the temperature at the outlet of the air heater system. This ambient temperature cannot be measured directly. So, Kalman filter has been used to estimate the states. To implement the kalman filter the ambient temperature is augmented with the ordinary state vector,that includes the temperature at the outlet of the air heater system. Two approaches are used to implement the MPC controller, one where the states are updated by the simulator and the other where the present states are calculated with the kalman filter. Finally, the controller developed on the simulated process is implemented in the real process, i.e. the air heater system. While implementing the Controller to the real process, constrained MPC control is used with imposing constraints on the control input from (0-5) V. But, the implementation of non-linear MPC to the real process is not successful. Therefore, only result for linear MPC implementation in the real process is included in the thesis.

9.2 Future Recommendations

There is still a lot of space to exploit in the field of MPC. Some of them can be listed as:

- During this thesis, state space model is used. The MPC algorithms can be tested using different models such as Impulse Response Model, step Response model, ARMAX model, CARIMA model, etc.
- SISO system is used during this thesis. It has been discussed in the section 5.2 that MPC takes account of cross Coupling in MIMO systems in an optimal way. Therefore, it seems interesting to apply MPC control algorithms in the system having multiple manipulated and controlled variables.
- Implementation of non-Linear MPC in the real process.

References

Bemporad, A., & Morari, M. (n.d.). Robust Model Predictive Control : A Survey Optimizer.

Bequette, B. W. (2007). Non-Linear Model Predictive Control : A Personal Retrospective †, *85*(October 2006).

Control, M. P. (2011). Model Predictive Control, 414–438. Retrieved from http://www.nt.ntnu.no/users/skoge/vgprosessregulering/papers-pensum/seborg-c20ModelPredictiveControl.pdf

Dale R. Seborg, Thomas F. Edgar, D. A. M. (n.d.). *Process Dynamics and Control* (Second Edi.).

Filter, K. (1960). State estimation with Kalman Filter, 213–239.

Findeisen, R., & Allg, F. (n.d.). An Introduction to Nonlinear Model Predictive Control 1 Principles , Mathematical Formulation and Properties of, 1–23.

Halvorsen, H. (n.d.). Model Based Control. Retrieved from http://home.hit.no/~hansha/documents/subjects/EE4209.htm

Halvorsen, H. P. (2011). *Model Predictive Control in LabVIEW*. Retrieved from http://home.hit.no/~hansha/documents/labview/training/Model Predictive Control in LabVIEW/Model Predictive Control in LabVIEW.pdf.

Haugen, F. (n.d.). No Title. Retrieved from http://home.hit.no/~finnh/air_heater/

Haugen, F. (2012). State Estimation of a Pilot Anaerobic Digestion Reactor.

Haugen, F. A. (n.d.). *PhD Dissertation : Optimal Design , Operation and Control of an Anaerobic Digestion Reactor*.

Haugen, F., Fjelddalen, E., Dunia, R., & Edgar, T. F. (2007). Demonstrating PID Control Principles using an Air Heater and LabVIEW, 1–38.

Kwee, H. M. E. I., Tan, A. M. Y., & Wern, S. (2006). FORMULATION OF MODEL PREDICTIVE CONTROL ALGORITHM FOR NONLINEAR PROCESSES (FORMULASI ALGORITMA KAWALAN RAMALAN MODEL UNTUK PROSES TIDAK LINEAR) BOO CHIN ENG KHAIRIYAH MOHD YUSOF RESEARCH VOTE NO : Jabatan Kejuruteraan Kimia Fakulti Kejuruteraan Kimia & .

Lee, J. H. (2011). No Title. *Model Predictive Control: Review of the Three Decades of Development*. Retrieved from http://link.springer.com/article/10.1007/s12555-011-0300-6#page-1

Manimaran, M., Arumugam, A., Balasubramanian, G., & Ramkumar, K. (n.d.). Optimization and composition control of Distillation column using MPC, *5*(2), 1224–1230.

Mayne, D. Q., Rawlings, J. B., Rao, C. V, Scokaert, P. O. M., National, C., & Telecom, F. (2000). Constrained model predictive control : Stability and optimality ☪, *36*.

Nagy, Z. K., & Braatz, R. D. (2003). Robust nonlinear model predictive control of batch processes. *AIChE Journal*, *49*(7), 1776–1786. doi:10.1002/aic.690490715

Predictive, M., Impulse, C., Response, S., Identification, M., Models, M., Calculations, C., & Parameters, T. (n.d.). (MPC) MPC - Motivation, 1–49.

Processes, L. (2010). Nonlinear Model Predictive Control and Dynamic Real Time Optimization for. Retrieved from http://numero.cheme.cmu.edu/uploads/Huangthesis.pdf

Qin, S. J., & Badgwell, T. a. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, *11*(7), 733–764. doi:10.1016/S0967-0661(02)00186-7

Ramesh, K., Hisyam, A., Aziz, N., & Shukor, S. R. A. (2012). Nonlinear Model Predictive Control of a Distillation Column Using Wavenet Based Hammerstein Model, (November).

Ruscio, D. Di. (2010). and optimization.

Ruscio, D. Di. (2012a). and optimization.

Ruscio, D. Di. (2012b). iscrete LQ optimal control with integral action: A simple controller on incremental form for MIMO systems. *Modeling, Identification and Control: A Norwegian Research Bulletin*, *33*(2), 35–44. doi:10.4173/mic.2012.2.1

Strand, S., & Sagli, J. R. (n.d.). MPC IN STATOIL – ADVANTAGES WITH IN-HOUSE TECHNOLOGY Stig Strand and Jan Richard Sagli. Retrieved from http://www.nt.ntnu.no/users/skoge/prost/proceedings/adchem03/contents/paperpdf/semiplen/811.pdf

Tricaud, C. (2008). Linear and nonlinear model predictive control using a general purpose optimal control problem solver RIOTS 95. *2008 Chinese Control and Decision Conference*, 1552–1557. doi:10.1109/CCDC.2008.4597578

Appendices

Appendix 1: Abstract

Appendix 2: Task Description

Appendix 3: MATLAB Code

Telemark University College
Faculty of Technology
M.Sc. Programme

MASTER'S THESIS, COURSE CODE FMH606

Student:	**Sharamnsha Bhandari**
Thesis title:	**Evaluation And Comparison of MPC Algorithms applied to a Real Air Heater**
Signature:	. . . . .
Number of pages:	73
Keywords:	Model Predictive Control, Non-linear MPC, Linear MPC, Air heater, fmincon, quadprog

Supervisor:	Finn Aakre Haugen	sign.:	. . . . .
2nd Supervisor:	<name>	sign.:	. . . . .
Censor:	<name>	sign.:	. . . . .
External partner:	<name>	sign.:	. . . . .
Availability:	<Open/Secret>		

Archive approval (supervisor signature)**:** sign.: . . . . . **Date :** . . . . . .

Abstract:

With a remarkable increase in demand for safe, efficient and stable control operation, process control for process plants has become more important. Although Conventional PID controllers have been used in the process industries for the longest time, the need of advanced control strategies such as MPC has been felt in not only the process industries but also in many academic and research activities. This thesis tries to draw attention towards basic idea of MPC theory, the controller performance and its practical implementation in the real physical process. For the implementation purpose, Linear and non-Linear MPC has been developed in the simulated process using the process model before its implementation on a real physical process.

The MPC control algorithm has been tested in the air heater system, where the objective is to keep the temperature of the air heater to the desired reference temperature. The two predictive control algorithms have been compared after applying controller to the process model/process in terms of relative stability margins (Phase Margin and Gain Margin), set point tracking (IAE criterion used as a measure), Horizon length and computational time.

It is noticed that non-linear MPC is better than linear MPC in terms of IAE as the error computed in non-linear MPC is found to be less. In addition, it is observed that non-linear MPC requires shorter horizon length as compared to linear MPC to give equally good, if not better results. However, if same horizon length is used for both linear and non-linear MPC, linear MPC requires less computational time than that of non-linear MPC. Based on the results obtained, it is observed that neither of the predictive control algorithms is that robust against any model change, which is analyzed after calculating Phase Margin for the controllers. In addition, the concept of Gain Margin is not that clear in MPC as it is difficult to calculate Gain Margin, as the controller response is not subjected to the stability limit even with the increase in loop gain that is inserted between controller and the process.

Kalman filter as the state estimator is used especially to estimate the ambient temperature, which cannot be measured directly. It makes sense to use kalman filter to estimate the states as the temperature at the surroundings influences the true air heater temperature.

Telemark University College
Faculty of Technology

FMH606 Master's Thesis

Title: Evaluation and comparison of MPC algorithms applied to a real air heater

TUC supervisor: Associate Professor Finn Haugen

External partner: -

Task description:

The main aim of the thesis is to evaluate and compare a number of model-based predictive control (MPC) algorithms as applied to a real laboratory scale air heater where the temperature is to be controlled.

If possible, the results should be presented in an article to be submitted to a recognized scientific journal. This article, or a draft of an article, will then be a part of the thesis.

Task background:

It is of interest to compare MPC algorithms with respect to implementation and use.

Student category:

SCE student. It is beneficial if the student can cooperate with the student assigned the thesis about comparing MPC algorithms as applied to simulated processes.

Practical arrangements:

The thesis will be accomplished at TUC.

Signatures:

Student (date and signature):

Supervisor (date and signature):

Address: Kjølnes ring 56, NO-3918 Porsgrunn, Norway. **Phone:** 35 57 50 00. **Fax:** 35 55 75 47.

A.1 MATLAB code for implementation of Linear MPC

```
%Script file for Linear MPC Implementation for Air Heater Model
%
% Sharamnsha Bhandari
% Telemark University College
clear all
clc
%continuous time model
T=20;
Kh=4;
%System Matrices
A1 = [-1/T 1/T;0 0];
B1 = [Kh/T;0];
D1 = [1 0];

% Discrete time model
%h=0.1;
%h=0.01;
%h=0.001;
h=0.001;
[A,B,D]=c2dm(A1,B1,D1,zeros(1),h);

% Weighting matrices
 p=0.01;
 q=1;
%p=3; q=1; L=2;
 L=40; %Prediction Horizon

 %Ts=0.001;
%  Ts=0.1;
%  t_start=0;
% t_stop=100;
% N=(t_stop-t_start)/Ts;
% t=[t_start:Ts:t_stop-Ts]';

% Dimensions and extended state space model in terms of deviation variables
nx=size(A,1);
nu=size(B,2);
ny=size(D,1);
%Augmented model matrices(mpc with integral action)
At=[A zeros(nx,ny);D eye(ny,ny)];
Bt=[B;zeros(ny,nu)];
Dt=[D eye(ny,ny)];

%Prediction model
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(ny,nu),L,0);
F_L=[OLB HdL];

Qt=q2qt(q,L);
Pt=q2qt(p,L);
H=F_L'*Qt*F_L + Pt;

% Simulation horizon
 %N=1/h;
 N=1000; %simulation Horizon
%N=5;

% reference signal
 ro=27;
 rand('seed',0); randn('seed',0);
 r=ro;
```

```
 r1=prbs1(N+L,600,1200); r1=ro(1)+1*r1 -1*r1(1);
R=r1;

%Declare some errays to be used
tau=0.2;dt=0.1;
nt=floor(tau/dt);
Delay=ones(nt+1,1);

%initial value for simulation
xo=[27;15];
uo=3;
%yo=xo(1);
yo=27;
x=xo;
u=uo;
u_old=u;
y_old=uo;
x_old=x;
ym=yo;
y=yo;

tic
for k=1:N-L
    %Implementing time delay
    ym=D*x;%y=ym;
    for id=nt+1:-1:2;
        Delay(id)=Delay(id-1);
    end
    Delay(1)=ym;
    y=Delay(nt+1);
    Ym(k,1)=ym;
    r1L=R(k+1:k+L); %future set point profile
    xt=[x-x_old;y_old];
    y_old=y;
    x_old=x;
     e1=R-y;
    pl=OL*At*xt;
    f=F_L'*Qt*(pl-r1L);
    duf=-inv(H)*f; %Unconstrained MPC control action
    u=u_old+duf(1:nu); u_old=u;
    u=u(1);

    %Input feed to the system

   f1=(-1/T)*x(1)+(1/T)*x(2)+(Kh/T)*u;
    f2=0;
    f=[f1;f2];
    %Update the states
    x(1:2)=x(1:2)+h*f;

    %store variables for plotting
    X(k,:)=x';
    Y(k,1)=y;
    U(k,1)=u;
end
too=toc
average_time=toc/k
t=1:N-L;
figure(1)
subplot(211), plot(t,[Y R(1:N-L)]), grid, ylabel('y and r[deg
Celsius]');legend('Output', 'Reference')
```

```
title('Output y and reference r')
subplot(212), plot(t,U), grid, ylabel('Control Input [V]')
xlabel('time[secs]')
```

A.2 MATLAB code for constrained linear MPC control

```
%script file for Constrained Linear MPC control with Integral action
%
% Sharamnsha Bhandari
% Telemark University College
clear all
clc

%continuous time model
T=20; Kh=4;
A1 = [-1/T 1/T;0 0];
B1 = [Kh/T;0];
D1 = [1 0];

% Discrete time model
%h=0.001;
h=0.1;
[A,B,D]=c2dm(A1,B1,D1,zeros(1),h);% Discretization

% Weighting matrices
% p=0.01; q=1;
% L=40;
%q=0.001; p=1;
q=0.001;p=1;
%q=q/p; p=1;
%q=0.01; p=1;
L=80; %Prediction model
%options=optimset('Largescale','off','Display','off');



%constraint matrices
[S,c]=scmat(1,L);
Acal=[eye(L);-eye(L);S;-S];
Du_max=ones(L,1)*0.01;
Du_min=ones(L,1)*(-0.01);
u_max=ones(L,1)*5;
u_min=ones(L,1)*0;

% Dimensions and extended state space model in terms of deviation variables
nx=size(A,1); nu=size(B,2); ny=size(D,1);
At=[A zeros(nx,ny);D eye(ny,ny)];
Bt=[B;zeros(ny,nu)];
Dt=[D eye(ny,ny)];

% Prediction Model
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(ny,nu),L,0);
F_L=[OLB HdL];

Qt=q2qt(q,L);
Pt=q2qt(p,L);
H=F_L'*Qt*F_L + Pt;



% Simulation horizon
```

```
%N=1/h;
%N=1000;
N=1000;
%N=1000;
% references
% R=zeros(N,1);
% rand('seed',0);
 %R=25*ones(N,1)-0.05*prbs1(N,170,320);

%  ro=35;
%     rand('seed',0); randn('seed',0);
%     r=ro;
%     r1=prbs1(N+L,600,1200); r1=ro(1)+1*r1 -1*r1(1);
%    R=r1;

%Declare some errays to be used
%tau=2;
tau=0.2; %delay time
%dt=0.2;
dt=0.1;
nt=floor(tau/dt);
Delay=ones(nt+1,1);

%initial value for simulation
xs=[28;20]; %steady state value for simulation of Unconstrained MPC
ys=27;
us=3;
x=xs;
u=us;
u_old=u;
y_old=ys;
x_old=x;
ym=ys;
y=ym;

%Reference
 ro=27;
  rand('seed',0); randn('seed',0);
  r=ro;
  r1=prbs1(N+L,600,1200); r1=ro(1)+1*r1 -1*r1(1);
 R=r1;
%R=30*ones(N,1);

tic
for k=1:N-L


    ym=D*x; %Measurement
    %y=ym;
    %delay matrix
    for id=nt+1:-1:2;
        Delay(id)=Delay(id-1);
    end
    Delay(1)=ym;
    y=Delay(nt+1);
    Ym(k,1)=ym;
    r1L=R(k+1:k+L); %future set point profile assumed to be known
    xt=[x-x_old;y_old];
    y_old=y;
    x_old=x;
    pl=OL*At*xt;
```

```
    f=F_L'*Qt*(pl-r1L);

    bk=[Du_max;-Du_min;u_max-c*u_old;-u_min+c*u_old];
    duf=quadprog(H,f,Acal,bk); %constrained MPC
    u=u_old+duf(1:nu);
    u_old=u;
    %store variables for plotting
    X(k,:)=x';
    Y(k,1)=y;
    U(k,1)=u;
    %input to the process
    f1=(-1/T)*x(1)+(1/T)*x(2)+(Kh/T)*u;
    f2=0;
    f=[f1;f2];
    %update the states
    x(1:2)=x(1:2)+h*f;

end
too=toc
average_time=toc/k
t=1:N-L;
figure(1)
subplot(211), plot(t,[Y R(1:N-L)]), grid, ylabel('y and r[deg
Celsius]');legend('Output', 'Reference')
title('Output y and reference r')
subplot(212), plot(t,U), grid, ylabel('Control Input [V]')
xlabel('time[secs]')
```

A.3 MATLAB code for Constrained Linear MPC control where the states are updated by Kalman filter

```
%script file for Constrained Linear MPC control with Integral action
%present states needed for the prediction is calculated by Kalman filter
%
% Sharamnsha Bhandari
% Telemark University College
clear all
clc

%continuous time model
T=20; Kh=4;
A1 = [-1/T 1/T;0 0];
B1 = [Kh/T;0];
D1 = [1 0];

% Discrete time model
%h=0.001;
h=0.1;
[A,B,D]=c2dm(A1,B1,D1,zeros(1),h);

% Weighting matrices
% p=0.01; q=1;
% L=40;
%q=0.001; p=1;
q=0.001;p=1;
%q=q/p; p=1;
%q=0.01; p=1;
L=60;
%options=optimset('Largescale','off','Display','off');

%  Ts=0.1;
%  t_start=0;
```

```
% t_stop=100;
% N=(t_stop-t_start)/Ts;
% t=[t_start:Ts:t_stop-Ts]';


%constraint matrices
[S,c]=scmat(1,L);
Acal=[eye(L);-eye(L);S;-S];
Du_max=ones(L,1)*0.01;
Du_min=ones(L,1)*(-0.01);
u_max=ones(L,1)*5;
u_min=ones(L,1)*0;

% Dimensions and extended state space model in terms of deviation variables
nx=size(A,1); nu=size(B,2); ny=size(D,1);
At=[A zeros(nx,ny);D eye(ny,ny)];
Bt=[B;zeros(ny,nu)];
Dt=[D eye(ny,ny)];
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(ny,nu),L,0);
F_L=[OLB HdL];
Qt=q2qt(q,L);
Pt=q2qt(p,L);
H=F_L'*Qt*F_L + Pt;



% Simulation horizon
%N=1/h;
%N=1000;
N=1000;
%N=1000;
% references
% R=zeros(N,1);
% rand('seed',0);
 %R=25*ones(N,1)-0.05*prbs1(N,170,320);

%  ro=35;
%     rand('seed',0); randn('seed',0);
%     r=ro;
%     r1=prbs1(N+L,600,1200); r1=ro(1)+1*r1 -1*r1(1);
%    R=r1;

%Declare some errays to be used
%tau=2;
tau=0.2;
%dt=0.2;
dt=0.1;
nt=floor(tau/dt);
Delay=ones(nt+1,1);

%initial value for simulation
xs=[28;23];
ys=27;
us=3;
x=xs;
u=us;
u_old=u;
y_old=ys;
x_old=x;
ym=ys;
y=ym;
```

```
 ro=27;
  rand('seed',0); randn('seed',0);
  r=ro;
  r1=prbs1(N+L,600,1200); r1=ro(1)+1*r1 -1*r1(1);
 R=r1;
%R=30*ones(N,1);

dx_k=zeros(2,1);
y_k_1=ys;
u_k_1=us;


W=0.09;
Q = diag([0.0729 0.04]);
[K,Xb,Xh]=dlqe(A,eye(2),D,Q,W); %calculate Kalman gain

for k=1:N-L
   ym=D*x;%y=ym;
    for id=nt+1:-1:2;
        Delay(id)=Delay(id-1);
    end
    Delay(1)=ym;
    y=Delay(nt+1);
    Ym(k,1)=ym;
    r1L=R(k+1:k+L); %future set point profile
    dx_k_d=dx_k

     xt=[dx_k_d;y_old];
    y_old=y;
    %x_old=x;
    pl=OL*At*xt;
    f=F_L'*Qt*(pl-r1L);

    bk=[Du_max;-Du_min;u_max-c*u_old;-u_min+c*u_old];
    duf=quadprog(H,f,Acal,bk);
    u=u_old+duf(1:nu);
    u_old=u;
    %store variables for plotting
    X(k,:)=x';
    Y(k,1)=y;
    U(k,1)=u;
    %input to the process
    f1=(-1/T)*x(1)+(1/T)*x(2)+(Kh/T)*u;
    f2=0;
    f=[f1;f2];
    %update the states
    x(1:2)=x(1:2)+h*f;
    du_k=u-u_k_1;
    y_k=x(1);
    dx_k=A*dx_k+B*du_k+K*(y_k-y_k_1-D*dx_k);
    y_k_1=y_k;
    u_k_1=u;
     deviation_k(k,:)=dx_k'
end
q=deviation_k+X

t=1:N-L;
figure(1)
subplot(411), plot(t,[Y R(1:N-L)]), grid, ylabel('y1_k and
r1_k');legend('Output', 'Reference');
```

```
title('Output y_k and reference r_k')
subplot(412), plot(t,U), grid, ylabel('Control u_k');title('MPC control
action value');

subplot(413); plot([X(:,1) q(:,1)]),
grid, ylabel('Temperautre[deg C]');legend('simulated value', 'Estimated
value');
title('state estimation of air heater temperature');
subplot(414); plot([X(:,2) q(:,2)]),
grid, ylabel('Temperautre[deg C]');legend('simulated value', 'Estimated
value');
title('state estimation of ambient temperature');xlabel('Time[secs]')
```

A.4 MATLAB code for Unconstrained linear MPC control where the states are updated by Kalman filter

```
%script file for UnConstrained Linear MPC control with Integral action
%present states needed for the prediction is calculated by Kalman filter
%
% Sharamnsha Bhandari
% Telemark University College
clear all
clc

%continuous time model
T=20; Kh=4;
A1 = [-1/T 1/T;0 0];
B1 = [Kh/T;0];
D1 = [1 0];

% Discrete time model
%h=0.001;
%h=0.001;
%h=0.001;
h=0.001;
[A,B,D]=c2dm(A1,B1,D1,zeros(1),h);

% Weighting matrices
 p=0.001; q=1;
%p=3; q=1; L=2;
 L=40;

 %Ts=0.001;
 Ts=0.1;
 t_start=0;
t_stop=100;
N=(t_stop-t_start)/Ts;
t=[t_start:Ts:t_stop-Ts]';

% Dimensions and extended state space model in terms of deviation variables
nx=size(A,1); nu=size(B,2); ny=size(D,1);
At=[A zeros(nx,ny);D eye(ny,ny)];
Bt=[B;zeros(ny,nu)];
Dt=[D eye(ny,ny)];
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(ny,nu),L,0);
F_L=[OLB HdL];
Qt=q2qt(q,L);
Pt=q2qt(p,L);
H=F_L'*Qt*F_L + Pt;

% Simulation horizon
 %N=1/h;
```

```
N=1000;
%N=5;
% references
 ro=27;
 rand('seed',0); randn('seed',0);
 r=ro;
 r1=prbs1(N+L,600,1200); r1=ro(1)+1*r1 -1*r1(1);
R=r1;
std_y_noise=1.2*0;
%Declare some errays to be used
tau=0.2;dt=0.1;
nt=floor(tau/dt);
Delay=ones(nt+1,1);

%initial value for simulation
xo=[27;20];
uo=3;
%yo=xo(1);
yo=27;
x=xo;
u=uo;
u_old=u;
y_old=yo;
x_old=x;
ym=yo;
y=yo;
%y_init=yo;

y_est=zeros(N-L,1)+y;
x_init=xo;
x_apost_k_minus_1=x_init;
P_init=eye(2,2);%initial covariance of estimation error
P_apost_k_minus_1=P_init;
dx_k=zeros(2,1);
y_k_1=yo;
u_k_1=uo;


W=0.09;
Q = diag([0.0729 0.04]);
[K,Xb,Xh]=dlqe(A,eye(2),D,Q,W); %calculate Kalman gain
tic
for k=1:N-L
    ym=D*x;%y=ym;
    for id=nt+1:-1:2;
        Delay(id)=Delay(id-1);
    end
    Delay(1)=ym;
    y=Delay(nt+1);
    Ym(k,1)=ym;
    r1L=R(k+1:k+L); %future set point profile

   dx_k_d=dx_k;

    xt=[dx_k_d;y_k_1];
    y_old=y_k_1;
    %x_old=x;

    e1=R-y;
    pl=OL*At*xt;
    f=F_L'*Qt*(pl-r1L);
```

```
    duf=-inv(H)*f;
    u=u_old+duf(1:nu); u_old=u;
    u=u(1);

   f1=(-1/T)*x(1)+(1/T)*x(2)+(Kh/T)*u;
    f2=0;
    f=[f1;f2];
    x(1:2)=x(1:2)+h*f;


    X(k,:)=x';
    Y(k,1)=y;
    U(k,1)=u;
   du_k=u-u_k_1;
   y_k=x(1);
    dx_k=A*dx_k+B*du_k+K*(y_k-y_k_1-D*dx_k);
    y_k_1=y_k;
    u_k_1=u;

    deviation_k(k,:)=dx_k'
%     y_k=y+std_y_noise*randn;
%   input_k=u;
%
% [P_apost,x_apost, y_apri,K]=
function_ekf1(y_k,P_apost_k_minus_1,x_apost_k_minus_1,input_k);
%
% y_est=y_apri;
%
% %Time shift:
% x_apost_k_minus_1=x_apost;
% P_apost_k_minus_1=P_apost;
% x_est(1:2)=x_apost(1:2);
% X_est=x_est';
end
too=toc
average_time=toc/k
q=deviation_k+X

t=1:N-L;
figure(1)
subplot(411), plot(t,[Y R(1:N-L)]), grid, ylabel('y1_k and
r1_k');legend('Output', 'Reference');
title('Output y_k and reference r_k')
subplot(412), plot(t,U), grid, ylabel('Control u_k');title('MPC control
action value');

subplot(413); plot([X(:,1) q(:,1)]),
grid, ylabel('Temperautre[deg C]');legend('simulated value', 'Estimated
value');
title('state estimation of air heater temperature');
subplot(414); plot([X(:,2) q(:,2)]),
grid, ylabel('Temperautre[deg C]');legend('simulated value', 'Estimated
value');
title('state estimation of ambient temperature');xlabel('Time[secs]')
```

A.4 MATLAB Function needed to implement Linear MPC

A.4.1 Function to create constraints

```
function [S,c] = scmat(nr,L);

S=zeros(nr*L,nr*L); c=zeros(nr*L,nr);
```

```
for i=1:L
  for j=1:i
    S((i-1)*nr+1:i*nr,(j-1)*nr+1:j*nr)=eye(nr);
  end
end
%
for i=1:L
  c((i-1)*nr+1:i*nr,:)=eye(nr);
end
```

A.4.2 Function to obtain Prediction Model

```
function [H,O,OB]=ss2h(A,B,D,E,L,g);

[ny,nu]=size(E);

O = eobsv(A,D,L);              % The extended observability matrix.
OB= O*B;                       % DB, DAB, and so on.
hi=[E;OB(1:(L-1)*ny,:)];       % The impulse responses which are needed.

                               % The lower block triangular Toeplizt matrix.
for j=1:L+g-1
  H((j-1)*ny+1:L*ny,(j-1)*nu+1:j*nu) =hi(1:(L-j+1)*ny,:);
end
```

A.4.3. Function to make Extended weighted matrix

```
function [qt] = q2qt(q,L);
[nc,nr]=size(q);
qt=zeros(nc*L,nr*L);

for i=1:L
  for j=1:L
    if i==j
     qt((i-1)*nc+1:i*nc,(j-1)*nr+1:j*nr)=q;
    end
  end
end
```

A.4.3 Function to generate pseudorandom Binary signal

```
function [u,t]=prbs1(N,Tmin,Tmax);
-------------

is=1;
Tmin= Tmin-1;
dT  = Tmax-Tmin;

u=zeros(N,1);                  % Make space for input signal.
if is==0
 s=sign(randn);                 % Sign of input (change) at time 0.
else
 s=1;
end
k=1;                           % Initialize integer counter for time to
switch.
it=1;                          % Initialize integer counter for # of
intervals.

while k < N+1
 T=Tmin+dT*rand; T=ceil(T);    % Compute random time horizon T in
 u(k:k+T-1)=s*ones(T,1);       % the interval [Tmin <= T <= Tmax].
```

```
 s=s*(-1);                    % Update sign variable s which is either -1
or 1.
 k=k+T;                       % Update time counter.

 t(it)=T;                     % Save intervals, not necessary.
 it=it+1;
end

u=u(1:N);                     % Last interval T may be shorter than Tmin.
```

A.5 MATLAB code for Non-Linear MPC Implementation

```
%---------------------------------
clear all
close all
%---------------------------------
kh=4; T=20;
Ts=0.1;
Np=10; %prediction horizon
t_start=0;
t_stop=15;
%Nsim=(t_stop-t_start)/Ts;
t=[t_start:Ts:t_stop-Ts]';
Nsim=150;
%---------------------------------------

x_steady=[37   23];

x1_init= x_steady(1);
x2_init= x_steady(2);
y_init = 37;

u_in_const= 3;

%Initial guessed optimal control sequence:
u_const=u_in_const;
u_old=0*zeros(Np,1)+u_const;
%-----------------------------------------------------

%State-space model implementing time-delay in MPC:
Timedelay_mpc=0.2;
nd_mpc=ceil(Timedelay_mpc/0.1);
Ad_mpc=diag([ones(nd_mpc-1,1)],-1);
Bd_mpc=[nd_mpc>=1;zeros(nd_mpc-1,1)];
Cd_mpc=[zeros(1,nd_mpc-1),nd_mpc>=1];
Dd_mpc=[nd_mpc==0];
x_delay_mpc_k=zeros(length(Ad_mpc),1)+u_const;

%-----------------------------------------------------
%Creating future setpoint profile, assumed known for MPC:
t_sp_step_start=0;
t_sp_ramp_start_1=2;
t_sp_const_start_1=4;
t_sp_ramp_start_2=9;
t_sp_const_start_2=10;
y_sp_const_init=y_init;
y_sp_const_1=y_init;
y_sp_slope_1=1;%1;%(L/d)/d;
y_sp_slope_2=0;%(L/d)/d;
```

```
R=[1:length(t)]'*0;%Array init
for k=1:length(t)
    if (t(k)>=t_sp_step_start),
        R(k,1)=y_sp_const_init;
    end
    if (t(k)>=t_sp_ramp_start_1),
        R(k,1)=R(k-1,1)-y_sp_slope_1*Ts;
    end
    if (t(k)>t_sp_const_start_1),
        R(k,1)=R(k-1,1);
    end
    if (t(k)>t_sp_ramp_start_2),
        R(k,1)=R(k-1,1)-y_sp_slope_2*Ts;
    end
    if (t(k)>t_sp_const_start_2),
        R(k,1)=R(k-1,1);
    end
end
%------------------------------------------------------

%------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Preallocation of arrays:
x1=zeros(Nsim-Np,1)+x1_init;
x2=zeros(Nsim-Np,1)+x2_init;

%Matrices defining linear constraints for use in fmincon:
A=[];
B=[];
Aeq=[];
Beq=[];
%-------------------------------------------
%Lower and upper limits of optim variable for use in fmincon:
lb=u_old*0;
ub=u_old*0+5;
%-------------------------------------------
%MPC costs:
c_e=1;
c_u=0;
c_du=0.01;
c_final=0;
weight=[c_e c_u c_du c_final];

%-------------------------------------------
tic
%For-loop for calculating optimal control sequence applied to simulated
process:
for k=1:Nsim-Np


    %Updating future setpoint profile as time elapses:
    RL=R(k:k+Np);

    %Time-shift of state:
    x_mpc_init=[x1(k);x2(k)];

    %Defining fun handle for fmincon:
    c=@(u)objfun(Ad_mpc,Bd_mpc,Cd_mpc,Dd_mpc,u,weight,RL,x_mpc_init,Np,Ts);
```

```
 options=optimset('Algorithm','active-
set','LargeScale','on','MaxIter',10000,'MaxFunEvals',1000*length(u_old),'Di
splay','off');
 [u_optimal,fval,exitflag,output,lambda,grad,hessian] =...
  fmincon(c,u_old,[],[],[],[],lb,ub,@confun,options);
u_old=u_optimal;
    u(k)=u_optimal(1);
    u_store(k)=u(k);

 in_delay_mpc_k=u(k);
  x_delay_mpc_k_plus_1=Ad_mpc*x_delay_mpc_k+Bd_mpc*in_delay_mpc_k;
  out_delay_mpc_k=Cd_mpc*x_delay_mpc_k+Dd_mpc*in_delay_mpc_k;
  inp(k)=out_delay_mpc_k;

 dx1=(-1/T)*x1(k)+(1/T)*x2(k)+ (kh/T)*inp(k);
 x1(k+1)=x1(k)+Ts*dx1;
 x2(k+1)=x2(k);
 x_delay_mpc_k=x_delay_mpc_k_plus_1;

 y(k)=x1(k);
 e(k)=R(k)-y(k);
 y_store(k)=y(k);
%  x1_store=x1(k);
%  x2_store=x2(k);
end
%-------------------------------------------------------
tt=toc
time=toc/k

% end
% S_vfa_mpc=S_vfa;
% %-------------------------------------------------------
t=(1:Nsim-Np)';
figure(1)
subplot(211), plot(t,y_store,t,R(1:Nsim-Np));
grid on;
ylabel('y and r[Deg C]');
legend('Output', 'Reference')
title('Output y and reference r')
subplot(212)
plot(t,u_store);
grid;
ylabel('Control Action[V]')
xlabel('time[secs]')
```

A.6 Non-Linear MPC Implementation where the states are updated by Kalman filter

```
clear all
close all
%---------------------------------
kh=4; T=20;
Ts=0.1;
%t_pred_horizon=1;
t_pred_horizon=4;
Np=t_pred_horizon/Ts;
t_start=0;
t_stop=20;
Nsim=(t_stop-t_start)/Ts;
t=[t_start:Ts:t_stop-Ts]';
%---------------------------------------
x_final=[37   23];
std_y_noise=1.2*0;
```

```
x1_init= x_final(1);
x2_init= x_final(2);

y_init = 37;

u_in_const= 3;%F_feed_end with Fmeth = 150;

%Initial guessed optimal control sequence:
u_const=u_in_const;
u_guess=0*zeros(Np,1)+u_const;

% array Initialization
x1=zeros(Nsim-Np,1)+x1_init;
%x2=zeros(Nsim-Np,1)+x2_init;
y=zeros(Nsim-Np,1)+y_init;
y_est=zeros(Nsim-Np,1)+y_init;
 x1=zeros(Nsim-Np,1)+x1_init;
 x2=zeros(Nsim-Np,1)+x2_init;

%Tuning of Kalman Filter
x_init=[x1_init;x2_init];
x_apost_k_minus_1=x_init;
P_init=eye(2,2);%initial covariance of estimation error
P_apost_k_minus_1=P_init;


%State-space model implementing time-delay in MPC:
Timedelay_mpc=0.2;
nd_mpc=ceil(Timedelay_mpc/Ts);
Ad_mpc=diag([ones(nd_mpc-1,1)],-1);
Bd_mpc=[nd_mpc>=1;zeros(nd_mpc-1,1)];
Cd_mpc=[zeros(1,nd_mpc-1),nd_mpc>=1];
Dd_mpc=[nd_mpc==0];
x_delay_mpc_k=zeros(length(Ad_mpc),1)+u_const;


%-------------------------------------------------------
%Creating future setpoint profile, assumed known for MPC:
t_sp_step_start=0;
t_sp_ramp_start_1=2;
t_sp_const_start_1=4;
t_sp_ramp_start_2=9;
t_sp_const_start_2=10;
y_sp_const_init=y_init;
y_sp_const_1=y_init;
y_sp_slope_1=1;%1;%(L/d)/d;
y_sp_slope_2=0;%(L/d)/d;

y_sp=[1:length(t)]'*0;%Array init
for k=1:length(t)
    if (t(k)>=t_sp_step_start),
        y_sp(k,1)=y_sp_const_init;
    end
    if (t(k)>=t_sp_ramp_start_1),
        y_sp(k,1)=y_sp(k-1,1)-y_sp_slope_1*Ts;
    end
    if (t(k)>t_sp_const_start_1),
        y_sp(k,1)=y_sp(k-1,1);
    end
    if (t(k)>t_sp_ramp_start_2),
        y_sp(k,1)=y_sp(k-1,1)-y_sp_slope_2*Ts;
```

```
    end
    if (t(k)>t_sp_const_start_2),
        y_sp(k,1)=y_sp(k-1,1);
    end
end


%-------------------------------------------
%Matrices defining linear constraints for use in fmincon:
A=[];
B=[];
Aeq=[];
Beq=[];
%-------------------------------------------
%Lower and upper limits of optim variable for use in fmincon:
lb=u_guess*0;
ub=u_guess*0+5;
%-------------------------------------------
%MPC costs:
c_e=1;
c_u=0;
c_du=0.001;
c_final=0;
weight=[c_e c_u c_du c_final];

%-------------------------------------------
tic
%For-loop for calculating optimal control sequence applied to simulated
process:
for k=1:Nsim-Np
    %t(k)

    %Updating future setpoint profile as time elapses:
    y_sp_to_optim=y_sp(k:k+Np);

    %Time-shift of state:
    x_mpc_init=x_apost_k_minus_1;

    %Defining fun handle for fmincon:

c=@(u)objfun(Ad_mpc,Bd_mpc,Cd_mpc,Dd_mpc,u,weight,y_sp_to_optim,x_mpc_init,
Np,Ts);
 options=optimset('Algorithm','active-
set','LargeScale','on','MaxIter',10000,'MaxFunEvals',1000*length(u_guess),'
Display','off');
 [u_optimal,fval,exitflag,output,lambda,grad,hessian] =...
  fmincon(c,u_guess,[],[],[],[],lb,ub,@confun,options);
u_start=u_optimal;
    u(k)=u_optimal(1);
    u_store(k)=u(k);

 in_delay_mpc_k=u(k);
  x_delay_mpc_k_plus_1=Ad_mpc*x_delay_mpc_k+Bd_mpc*in_delay_mpc_k;
  out_delay_mpc_k=Cd_mpc*x_delay_mpc_k+Dd_mpc*in_delay_mpc_k;
  inp(k)=out_delay_mpc_k;

 dx1=(-1/T)*x1(k)+(1/T)*x2(k)+ (kh/T)*inp(k);
 x1(k+1)=x1(k)+Ts*dx1;
 x2(k+1)=x2(k);
 x_delay_mpc_k=x_delay_mpc_k_plus_1;
```

```
 y(k)=x1(k);
 e(k)=y_sp(k)-y(k);
 y_store(k)=y(k);
%  x1_store=x1(k);
%  x2_store=x2(k);
y_k=y(k)+std_y_noise*randn;
input_k=inp(k);
[P_apost,x_apost, y_apri,K]=
function_ekf(y_k,P_apost_k_minus_1,x_apost_k_minus_1,input_k);

y_est(k)=y_apri;

%Time shift:
x_apost_k_minus_1=x_apost;
P_apost_k_minus_1=P_apost;
x1_est(k)=x_apost(1);
x2_est(k)=x_apost(2);
end

t=(1:Nsim-Np)';
figure(1)
subplot(411), plot(t,y_store,t,y_sp(1:Nsim-Np));
grid on;
ylabel('y and r[Deg C]');
legend('Output', 'Reference')
title('Output y and reference r')
subplot(412)
plot(t,u_store);
grid;
ylabel('Control Action[V]')
xlabel('time[secs]')
subplot(413)
 plot([x2(1:Nsim-Np) x2_est']),grid, ylabel('Temperautre[deg
C]');legend('simulated value', 'Estimated value');
 title('state estimation of ambient temperature');xlabel('time[secs]');
 subplot(414)
 plot([x1(1:Nsim-Np) x1_est']),grid, ylabel('Temperature[deg
C]');legend('simulated value', 'Estimated value');
 title('state estimation of air heater temperature');xlabel('time[secs]');
```

A.7 MATLAB functions used for implementation of Non-Linear MPC

A.7.1 Function to compute inequalities in control input

```
function [c,ceq]=confun(u)
c=[];
ceq=[];
end
```

A.7.2 Objective Function MATLAB code

```
function f=objfun(Ad_mpc,Bd_mpc,Cd_mpc,Dd_mpc,u,weight,R,x_mpc_init,Np,Ts)
q=weight(1);
w=weight(2);
p=weight(3);
T=20;
kh=4;
x1=zeros(1,Np)+x_mpc_init(1);
x2=zeros(1,Np)+x_mpc_init(2);
x_delay_mpc_k=zeros(length(Ad_mpc),1)+u(1);
y=zeros(1,Np);
e=zeros(1,Np);
J1=zeros(1,Np);
u_km1=u(1);
```

```
for k=1:Np
 in_delay_mpc_k=u(k);
 x_delay_mpc_k_plus_1=Ad_mpc*x_delay_mpc_k+Bd_mpc*in_delay_mpc_k;
 out_delay_mpc_k=Cd_mpc*x_delay_mpc_k+Dd_mpc*in_delay_mpc_k;
 inp(k)=out_delay_mpc_k;
dx1=(-1/T)*x1(k)+(1/T)*x2(k)+ (kh/T)*u(k);
x1(k+1)=x1(k)+Ts*dx1;
x2(k+1)=x2(k);
y(k)=x1(k);
e(k)=R(k)-y(k);
du_k=(u(k)-u_km1)/Ts;
J1(k+1)=J1(k)+Ts*(q*e(k)*e(k)+w*u(k)*u(k)+p*du_k*du_k);
u_km1=u(k);
x_delay_mpc_k=x_delay_mpc_k_plus_1;
end
J=J1(end);
f=J;
end
```

A.8 MATLAB Code for Kalman filter estimation

```
function [P_apost,x_apost, y_apri,K]=
function_ekf(y_k,P_apost_k_minus_1,x_apost_k_minus_1,input_k)
clc
t_final = 20;
dt=0.1; % sampling time
t=0:dt:t_final;
N=length(t);
R1 = 0.09; % Measurement noise covariance
%Q = diag([0.126025 0.055225]); %process noise covariance
%Q = diag([0.0009 0.0004]);
Q = diag([0.0729 0.0004]);
w = sqrt(Q)*randn(2,N); %process noise
v = sqrt(R1)*randn(1,N);
E=eye(2,2);
%P_apost=eye(2,2);%initial covariance of estimation error
%x_apost=[30.5;15]; %initial state estimate
%x = x_apost;
Ts = 0.1; %Discrete time used when caicaulate matrix A,B
Kh=4;
T=20;
A=[(T-Ts)/T Ts/T; 0 1]; B=[Kh/T;0];
C = [1 0];
%u = 3;%control signal into the air heater /V

% Initialize arrays for plotting at the end of the program
% Y=zeros(N,1);
% X=zeros(N,2);
% X_kalman=zeros(N,2);
 for i=1:N
 %output measurement states(or calculated from model)
    %X(i,:) = x';
 %simulation state variable for next step
    %x = x+dt*(A*x+B*input_k)+ v(:,i);
    %Simulate measurement(or calculated from model)
%     y = x(1)+ v(:,i);
%     Y_m(i,:) = y';

    %%Time update
    %(1) Apriori estimate(predicted estimate)
          x_apri = x_apost_k_minus_1 +
dt*airheater(x_apost_k_minus_1,input_k);
```

```
    %(2) Covariance of estimation error
          P_apri = A*P_apost_k_minus_1*A'+E*Q*E';

    %%Measurement update
    %(1) Kalman filter gain matrix
         K=P_apri*C'*inv(C*P_apri*C'+R1);
         %K_gain(i,:)=K'; % save the value of  Kalman gain
    %(2) Measurement estimate
         y_apri=x_apri(1);
         %Y_p(i,:)=y_apri';
    %(3) State estimate or corrected estimate
         x_apost=x_apri+K*(y_k-y_apri);
         %X_kalman(i,:)=x_apost';
    %(4) Update Covariance of estimation error
         I=eye(2,2);
         P_apost=(I-K*C)*P_apri*(I-K*C)'+K*R1*K';
end


% subplot(311)
% plot(t,X(:,1),'b-',t,X_kalman(:,1),'r--')
% ylabel('Th[deg]')
% title('state estimation for Th')
% legend('simulated Value','Estimated value')
% grid on
% subplot(312)
% plot(t,X(:,2),'b-',t,X_kalman(:,2),'r--')
% ylabel('Tenv[deg]')
% xlabel('Time')
% title('state estimation for ambient temperature')
% legend('simulated value','estimated value')
% grid on
%  subplot(313)
%  plot(t,K_gain(:,1),':k',t,K_gain(:,2),'.-','linewidth',2)
%  ylabel('Kalman gain')
%  title('Kalman Gain Estimation')
%  grid on
```

A.9 MATLAB Code for Air Heater Model

```
function dx =airheater(x,u)
[m,n]= size(x);
x_state=zeros(m,n);
T=20;kh=4;
% A=[(T-dt)/T dt/T; 0 1]; B=[(kh*dt)/T;0]; C=[1 0]; D=0;
A=[-1/T 1/T;0 0]; B=[kh/T;0]; C=[1 0]; D=0;
G=ss(A,B,C,D,'InputDelay',0.2);
A1=G.a;B1=G.b;
x_state=A1*x+B1*u;
dx=x_state;
end
```

A.10 MATLAB code to implement Linear MPC in Real System

```
clear all
clc
%DAQ
mydaq = daq.createSession('ni');
mydaq.addAnalogOutputChannel('dev18', 'ao0', 'Voltage');
mydaq.addAnalogInputChannel('dev18', 'ai0', 'Voltage');

%continuous time model
T=20; Kh=4;
```

```
A1 = [-1/T 1/T;0 0];
B1 = [Kh/T;0];
D1 = [1 0];
% Discrete time model
%h=0.001;
h=0.1;
[A,B,D]=c2dm(A1,B1,D1,zeros(1),h);

% Weighting matrices
% p=0.01; q=1;
% L=40;
q=0.0001;
%q=3;
p=1;
%q=q/p; p=1;
%q=0.01; p=1;
L=60;
%options=optimset('Largescale','off','Display','off');

%constraint matrices
[S,c]=scmat(1,L);
Acal=[eye(L);-eye(L);S;-S];
Du_max=ones(L,1)*0.01;
Du_min=ones(L,1)*(-0.01);
u_max=ones(L,1)*5;
u_min=ones(L,1)*0;

% Dimensions and extended state space model in terms of deviation variables
nx=size(A,1); nu=size(B,2); ny=size(D,1);
At=[A zeros(nx,ny);D eye(ny,ny)];
Bt=[B;zeros(ny,nu)];
Dt=[D eye(ny,ny)];
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(ny,nu),L,0);
F_L=[OLB HdL];
Qt=q2qt(q,L);
Pt=q2qt(p,L);
H=F_L'*Qt*F_L + Pt;



% Simulation horizon
%N=1/h;
%N=1000;
N=1000;
% references
% R=zeros(N,1);
% rand('seed',0);
 %R=25*ones(N,1)-0.05*prbs1(N,170,320);

% ro=25;
%    rand('seed',0); randn('seed',0);
%    r=ro;
%    r1=prbs1(N+L,600,1200); r1=ro(1)+1*r1 -1*r1(1);
%   R=r1;

%Declare some errays to be used
tau=2;dt=0.2;
nt=floor(tau/dt);
Delay=ones(nt+1,1);

%initial value for simulation
```

```
xs=[37;20];
ys=37;
us=3;
x=xs;
u=us;
u_old=u;
y_old=ys;
x_old=x;
 ym=ys;
 y=ym;

%   ro=39;
%   rand('seed',0); randn('seed',0);
%   r=ro;
%   r1=prbs1(N+L,600,1200); r1=ro(1)+1*r1 -1*r1(1);
%  R=r1;
%Reference
R=36*ones(N,1);


for k=1:N-L


     %ym=D*x;%y=ym;
%     for id=nt+1:-1:2;
%         Delay(id)=Delay(id-1);
%     end
%     Delay(1)=ym;
%     y=Delay(nt+1);
%     Ym(k,1)=ym;
    r1L=R(k+1:k+L); %future set point profile
    xt=[x-x_old;y_old];

    y_old=y;
    x_old=x;
    pl=OL*At*xt;
    f=F_L'*Qt*(pl-r1L);

    bk=[Du_max;-Du_min;u_max-c*u_old;-u_min+c*u_old];
    duf=quadprog(H,f,Acal,bk);
    u=u_old+duf(1:nu);
    u_old=u;
    ao_value=u;
    mydaq.outputSingleScan(ao_value);
    y_vol = mydaq.inputSingleScan;



    %y= data_daq(u_old);
    y=6*y_vol+20;

   %x(1)=y12



    %y_scale=7.5*y+12.5;
    %store variables for plotting
    X(k,:)=x';

   Y(k,1)=y;
```

```
    %Y(k,1)=y_scale;
    U(k,1)=u;
    %input to the process
     %f1=(-1/T)*x(1)+(1/T)*x(2)+(Kh/T)*u;

   %f2=0;
     %f=[f1;f2];
%     %update the states
     %x(1:2)=x(1:2)+h*f;

end
t=1:N-L;
figure(1)
subplot(211), plot(t,[Y R(1:N-L)]), grid, ylabel('y1_k and r1_k')
title('Output y_k and reference r_k')
subplot(212), plot(t,U), grid, ylabel('Control u_k')
xlabel('Discrete time')
```

A.11 MATLAB code to implement non-linear MPC in Real system

```
function ai_value= daq_device(inp)
mydaq = daq.createSession('ni');
mydaq.addAnalogOutputChannel('dev1', 'ao0', 'Voltage');
mydaq.addAnalogInputChannel('dev1', 'ai0', 'Voltage');
ao_value = inp;
mydaq.outputSingleScan(ao_value)
ai_value = mydaq.inputSingleScan;
end

clear all
clc
Ts=0.1;
L=4; %prediction Horizon;
Np=L/Ts;
t_start=0;
t_stop=20;
Nsim=(t_stop-t_start)/Ts;
t=[t_start:Ts:t_stop-Ts]';
x1_init=37;
x2_init=15;
y_init=37;
% u_const=2;
% u_start=0*zeros(Np,1)+u_const; %starting guess value for control input

u_input_const=3;

%Initial guessed optimal control sequence:
u_const=u_input_const;
u_start=0*zeros(Np,1)+u_const;

%Time Delay
T=20;
kh=4;
Timedelay_mpc=0.2;
nd_mpc=ceil(Timedelay_mpc/Ts);
Ad_mpc=diag([ones(nd_mpc-1,1)],-1);
Bd_mpc=[nd_mpc>=1;zeros(nd_mpc-1,1)];
Cd_mpc=[zeros(1,nd_mpc-1),nd_mpc>=1];
Dd_mpc=[nd_mpc==0];
x_delay_mpc_k=zeros(length(Ad_mpc),1)+u_const;

x1=zeros(Nsim-Np,1)+x1_init;
```

```
x2=zeros(Nsim-Np,1)+x2_init;
y=zeros(Nsim-Np,1)+y_init;

% N=3000;
% %set point profile
% r0=25;
% rand('seed',0); randn('seed',0);
% r=r0;
% r1=prbs1(Np+L,600,1200); r1=r0(1)+1*r1 -1*r1(1);
% R=r1; %set point


for k=1:length(t)
   R(k,1)=y_init;
    end

%weights
q=1;
w=0;
p=0.01;

weight=[q w p];

% p=0;
% q=1;
% w=0.01;
% weight=[q,p,w];

lb=u_start*0;
ub=u_start*0+5;

for k=1:Nsim-Np
    RL=R(k:k+Np); %future set point profile
    x_mpc_init=[x1(k);x2(k)];
    %y_mpc_init=y(k);
   c=@(u)objfun(Ad_mpc,Bd_mpc,Cd_mpc,Dd_mpc,u,weight,R,x_mpc_init,Np,Ts)
 options=optimset('Algorithm','active-
set','LargeScale','on','MaxIter',10000,'MaxFunEvals',1000*length(u_start),'
Display','off');
 [u_optimal,fval,exitflag,output,lambda,grad,hessian] =...
  fmincon(c,u_start,[],[],[],[],lb,ub,@confun,options);
u_start=u_optimal
    u(k)=u_optimal(1);
    u_store(k)=u(k);

    ao_value = u;
    mydaq.outputSingleScan(ao_value)
y_daq = mydaq.inputSingleScan;
y(k)=6*y_daq+20;
 y_store(k)=y(k);
%  x1_store=x1(k);
%  x2_store=x2(k);
end

 y_store
 t_plot=t(1:Nsim-Np);
t_plot_start=t_plot(1);
 t_plot_stop=t_plot(end);
 i_plot=[1:length(t_plot)];
```

```
plot(t_plot,R(i_plot),'k',t_plot,y_store(i_plot),'r')
grid on;
```