CRANFIELD UNIVERSITY

Rihab Khalid Al Seyab

Nonlinear Model Predictive Control Using Automatic Differentiation

School of Engineering

PhD Thesis

2006

CRANFIELD UNIVERSITY

School of Engineering

Department of Process And System Engineering

PhD Thesis

2006

Rihab Khalid Shakir Al Seyab

Nonlinear Model Predictive Control Using Automatic Differentiation

Supervisor: Dr. Yi Cao

Academic Year: 2005–2006

This thesis is submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Abstract

Although nonlinear model predictive control (NMPC) might be the best choice for a nonlinear plant, it is still not widely used. This is mainly due to the computational burden associated with solving online a set of nonlinear differential equations and a nonlinear dynamic optimization problem in real time. This thesis is concerned with strategies aimed at reducing the computational burden involved in different stages of the NMPC such as optimization problem, state estimation, and nonlinear model identification.

A major part of the computational burden comes from function and derivative evaluations required in different parts of the NMPC algorithm. In this work, the problem is tackled using a recently introduced efficient tool, the automatic differentiation (AD). Using the AD tool, a function is evaluated together with all its partial derivative from the code defining the function with machine accuracy.

A new NMPC algorithm based on nonlinear least square optimization is proposed. In a first–order method, the sensitivity equations are integrated using a linear formula while the AD tool is applied to get their values accurately. For higher order approximations, more terms of the Taylor expansion are used in the integration for which the AD is effectively used. As a result, the gradient of the cost function against control moves is accurately obtained so that the online nonlinear optimization can be efficiently solved.

In many real control cases, the states are not measured and have to be estimated for each instance when a solution of the model equations is needed. A nonlinear extended version of the Kalman filter (EKF) is added to the NMPC algorithm for this purpose. The AD tool is used to calculate the required derivatives in the local linearization step of the filter automatically and accurately.

Offset is another problem faced in NMPC. A new nonlinear integration is devised for this case to eliminate the offset from the output response. In this method, an integrated disturbance model is added to the process model input or output to correct the plant/model mismatch. The time response of the controller is also improved as a by–product.

The proposed NMPC algorithm has been applied to an evaporation process and a two continuous stirred tank reactor (two–CSTR) process with satisfactory results to cope with large setpoint changes, unmeasured severe disturbances, and process/model mismatches.

When the process equations are not known (black–box) or when these are too complicated to be used in the controller, modelling is needed to create an internal model for the controller. In this thesis, a continuous time recurrent neural network (CTRNN) in a state–space form is developed to be used in NMPC context. An efficient training algorithm for the proposed network is developed using AD tool. By automatically generating Taylor coefficients, the algorithm not only solves the differentiation equations of the network but also produces the sensitivity for the training problem. The same approach is also used to solve online the optimization problem of the NMPC. The proposed CTRNN and the predictive controller were tested on an evaporator and two–CSTR case studies. A comparison with other approaches shows that the new algorithm can considerably reduce network training time and improve solution accuracy.

For a third case study, the ALSTOM gasifier, a NMPC via linearization algorithm is implemented to control the system. In this work a nonlinear state–space class Wiener model is used to identify the black–box model of the gasifier. A linear model of the plant at zero–load is adopted as a base model for prediction. Then, a feedforward neural network is created as the static gain for a particular output channel, fuel gas pressure, to compensate its strong nonlinear behavior observed in open–loop simulations. By linearizing the neural network at each sampling time, the static nonlinear gain provides certain adaptation to the linear base model. The AD tool is used here to linearize the neural network efficiently. Noticeable performance improvement is observed when compared with pure linear MPC. The controller was able to pass all tests specified in the benchmark problem at all load conditions.

Keywords:

Continues Time Recurrent Neural Network, Offset Free Control, Nonlinear System Identification, Coal Gasification, Nonlinear Integration.

Acknowledgement

I like to thank my thesis supervisor Dr. Yi Cao for his help and guidance and for suggesting the present project. His encouragement to publish in conferences and scientific journals is particularly acknowledged.

I like to thank Prof. Mike Sanderson (previous head of the PASE department) and Dr. Vivian Morris (responsible for research) for various matters and for waiving a major part of the tuition fees. A lot of thanks to prof. Chris Thompson (the head of this department) for his help and encouragement. The help of the department secretaries Mrs. Janet Caudrey and Mrs. Linda Whitfield are particularly acknowledged. Mr. Mike Devonshire has been really helpful whenever needed. Thanks are also due to Dr. Prabirkumar Saha (visiting staff) for his consultations.

The help and encouragement of Margaret and Lynne of the Community Development Office of Cranfield University are gladly acknowledged.

I like also to thank my family (in Iraq), and friends in the Cranfield University residence for their encouragement and support. My husband Riad and two children Resha and Reshad deserve special thanks for their support, love, patience, and for putting up with me for all these years.

Finally, I like to thank Dr. James Whidborne (the internal member of the examination committee) for his careful reading of this thesis and pointing out few mistakes.

Rihab Khalid Al Seyab

Cranfield University, April 2006

Contents

List of Tables

List of Figures

Nomenclature

In this section the notational conventions used for mathematical symbols, operators and abbreviation are presented.

Lower Case Symbols

b	Bias vector in the Neural Network structure
d	The Taylor series order
$\hat{d}$	Estimated value of the input disturbance for offset free control
d_o	Output integrated disturbance for offset free control
e	Error
e_{ss}	Offset (steady–state) error
f	State and input mapping for the process model
f_{NN}	Nonlinear function to represent the neural network structure
g	Output mapping for the process model
k	Time instant, index
m	The dependent variables number in chapter 3
n	The independent variables number in Chapter 3
n_x	Number of the state variables
n_y	Number of the output variables
n_u	Number of the manipulated variables
r	Reference signal at time t
u	Manipulated variable vector
t	Time vector
x	State variable vector
y	Output variable vector

Upper Case Symbols

A	State dynamic matrix of the linear state–space system
B	Control signal dynamic matrix of the linear state–space system
C	Output–state matrix of the linear state–space system
D	Output–input matrix of the linear state–space system
E	Error vector
F	Nonlinear mapping of the dynamic part in a discrete–time state–space model
G	The linearized model matrix in chapter 5, or the matrix of the steady–state gain in chapter 8
H	Hessian matrix
I	Identity matrix
J	Jacobian matrix
K_1	Gain parameter of the nonlinear integrator function
K_2	Slop parameter of the nonlinear integrator function
M	Control horizon
N	Number of the training (validation) data
P	Prediction horizon
Q	Output weighting matrix of the MPC objective function
R	Control signals weighting matrix of the MPC objective function
S	Control signal change weighting matrix of the MPC algorithm, or a weighting matrix of the QP problem of chapter 8
$\mathbb{R}^n$	Euclidean n–dimensional space
R_{xy}	Cross-correlation function between x and y
U	Manipulated variable vector
T_s	Sampling time
V	Objective function of the NMPC algorithm
W	Weights vector of the neural network
Y	Output variables vector
Y_m	The vector of measured output variables
Y_r	The vector of desired output variables
X	State variables vector

Greek Symbols

α	The state dynamic matrix of the linearized model of the EKF
β	The input dynamic matrix of the linearized model of the EKF
γ	Nonlinear function for the nonlinear integrator
η	The independent variables vector in chapter 7
φ	Regression vector of NARAX–NN model in Chapter 2
ϑ	The actuators noise (white noise) in the EKF method
ω	The measurement noise (white noise) in the EKF method
τ	Normalized time
σ_s	The sigmoid–tanh function of the neural network
Γ	The static matrix of the linearized model of the EKF
Ψ	Constant matrix of the MPC for the gasifier case study
Σ	Covariance matrix in the EKF formulation
$\triangle u$	Changes in manipulated variables
Φ	Performance index
θ	Neural Network parameters vector

Superscripts

T	Transpose
y	Output index

Subscripts

0	Initial value
f	final value
k	k instant
L	Linear
NL	Noninear
NN	Neural network
max	Maximum
min	Minumum

u	Control signal
x	State
y	Output

Abbreviations

AD	Automatic Differentiation
ANN	Artificial Neural Network
CSTR	Continuous stirred tank reactors
CTRNN	Continuous–Time Recurrent Neural Network
DD1	First–order Divided Difference
DD2	Second–order Divided Difference
CFD	Centered Finite Difference
DMC	Dynamic Matrix Control
EKF	Extended Kalman Filter
FD	Finite Difference
FFNN	FeedForward Neural Network
GPC	Generalized Predictive Control
IAE	Integral Absolute Error
LM	Levenberge Marquardt
LMPC	Linear Model Predictive Control
MLP	Multi–Layer Preceptron
MPC	Model Predictive Control
NLP	Nonlinear Programming
NLSQ	Nonlinear Least Square
NMPC	Nonlinear Model Predictive Control
ODEs	Ordinary Differential Equations
PID	Proportional Integral Derivative
QP	Quadratic Programming
RNN	Recurrent Neural Network
SD	Symbolic Difference
SQP	Sequential Quadratic Programming

Chapter 1

Introduction

1.1 Linear Versus Nonlinear MPC

In the last two decades, model predictive control (MPC) has become a first choice for an advanced control strategy in industry, due to its intuitiveness and capability to handle multivariables systems with constraints. MPC can be found now in a wide variety of manufacturing environments including power plants, petroleum refinery applications, chemicals, food processing, automotive, aerospace, metallurgy, and others [QB97].

MPC is a class of computer control algorithms that control the future behavior of a plant through the explicit use of a dynamic model of the process. At each control interval the MPC algorithm computes an open–loop sequence of manipulated variables (*control horizon*) in order to optimize plant behavior in the time ahead (*the prediction horizon*). Process constraints and any changes in the process objectives or operating conditions can be implemented online. This is one of attractions of MPC. This is also a major difference from other control theories. MPC is referred to sometimes as *receding horizon* (or *moving horizon*) control because only the first of the control moves is implemented. A new process measurement is injected in the control loop and the entire optimization is repeated at subsequent control intervals. At the heart of the MPC controller is the model, which is used not only to forecast the effects of future inputs, but also to estimate the current state of the plant from the given history of past measurements and control. The MPC strategy is illustrated in Figure 1.1.

Until recently, industrial applications of MPC have relied on linear dynamic models even though most processes are nonlinear. Linear MPC (LMPC) is acceptable when

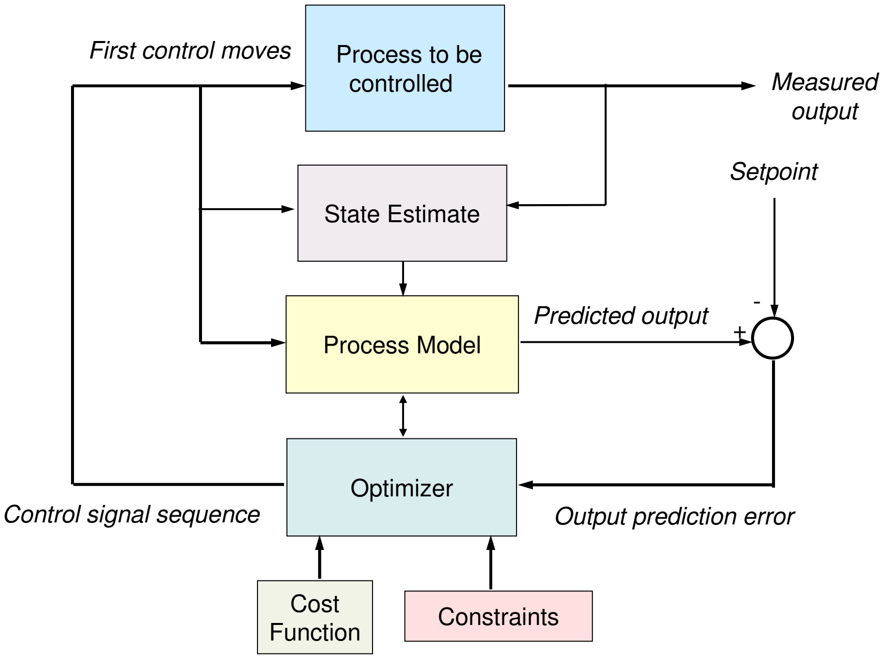

Figure 1.1: The MPC structure.

the process operates at a single operating point and the primary use of the controller is the rejection of small disturbances [PiS00]. The main situations leading to nonlinear MPC (or nonlinear control in general) are; *regulator control* problems where the process is subject to large frequent disturbances hence shows a strong degree of nonlinearity, and *servo control* problems where the operation points change frequently and span a sufficiently wide range of the process dynamics. Both cases are present in most modern chemical processes. However, the extension to nonlinear model based predictive control has not been very successful despite a significant amount of research effort having been put into this area. The main hurdle facing the extension of LMPC to NMPC is the significant computation burden especially in the case of large dimension, fast time response, and highly nonlinear processes. Any strategy that can be devised to alleviate computation burden is therefore desirable. In fact, time saving is needed in all of the different phases of the NMPC problem [BQ01] such as;

1. Rapid, reliable solution of a nonlinear control algorithms in real time.
2. Nonlinear state estimation.
3. Nonlinear model development.

The aim of this thesis is to contribute to reducing the computation burden, and increasing the calculation accuracy in all three phases of the predictive control, thus permitting the NMPC algorithm to be extended far beyond its present limitations.

1.2 The Online Optimization Problem

There exist a number of strategies for tracking the optimal control problem [Bin01]. Some of the common methods are; *successive linearization*, *sequential solution*, *simultaneous method*, and others. All these approaches require intensive computations of partial derivatives such as the sensitivity variables which are defined by the first and second partial derivatives of the process outputs with respect to its inputs or what are known the *Jacobean* and *Hessian* matrices. In a typical situation, calculating dynamic sensitivity could take more than 70% of the total computation time of NLP. Hence, dynamic sensitivity calculations are the computational bottleneck for solving a dynamic optimization problem. In addition, a large computation burden would be inevitable in the solution of a large set nonlinear differential equations representing the nonlinear model itself .

The Automatic differentiation (AD) tool has been recently introduced to handle such tasks [GW04]. AD is more convenient and flexible than manual differentiation and computationally faster than symbolic differentiation and more accurate than numerical differentiation even for complex highly–nonlinear systems. AD find the derivatives of a function given in the form of a computer code using the chain rule. This is done with no truncation error for any selected outputs with respect to selected inputs. So, the resulting derivatives have the same accuracy as the function itself [Gri00]. The AD does the job with much superior speed and accuracy that results in time saving in calculating the sensitivity variables or any order of partial derivative in addition to further time savings in the optimizer resulting from the higher accuracy of calculation.

In the present work an efficient NMPC algorithm is developed using a nonlinear least square optimizer (NLSQ) and the AD tool [CAS03]. The nonlinear least square method that arises in this algorithm is solved using only the residual Jacobian matrix. This is less time consuming compared to other optimizers as the Hessian matrix is estimated from the Jacobean matrix. In one algorithm, a numerical integration method is used to solve the process ordinary differential equations (ODEs), while a shortcut approach is derived to calculate the residual Jacobian matrix directly from the model sensitivity function using one step linear approximation. The dynamic sensitivity equations as a result are simplified and the AD tool is used for fast and accurate differentiation of the state trajectory.

An algorithm to solve ODEs and sensitivity equations using high–order Taylor series and AD for autonomous systems given in [Gri95] is extended to non–autonomous systems by Cao (2005) [Cao05]. In this work, the NMPC problem was recast as a standard nonlinear programming problem and this formulation is extended further to handle the case of black–box processes [ASC05c, ASC06]. A state estimation stage is added to the controller for the case of unmeasurable state variables. Also, a new approach to treat the problem of offset error for persisting disturbances and modelling error is developed and added to the NMPC algorithm.

1.3 The Internal Model

The internal model in NMPC can be constructed from the physical laws governing the behavior of the true system, if these are known. This is often referred to as a *first–principle*, *mechanistic*, or a *white–box* model. An alternative method for constructing a model is based on measurement of input and output data from the real plant. This method belongs to what is known as *system identification* [BM90, DFL94, HS97] and the resulting model is called an *empirical* or a *black–box* model.

In the case of the first–principle model, the sensitivity information can be calculated directly from the models' ODEs using the AD tool. For the second case a model need to be constructed first before the controller is implemented. This is achieved via system identification in state–space form. A continuous–time recurrent neural network (CTRNN) is developed in this work to be used as the process model. The neural network is represented in a general nonlinear state-space form and used to construct the internal model of the NMPC approach. An efficient training algorithm for the proposed network is developed here using AD techniques. By automatically generating Taylor coefficients, the algorithm not only solves the differentiation equations of the network but also produces the sensitivity for the training problem. The same approach is used to solve the online optimization problem in the predictive controller. The proposed neural network and the nonlinear predictive controller were tested on two cases studies; an evaporation system and a two–CSTR process.

For the third application, the ALSTOM gasifier benchmark problem, a nonlinear state–space class Wiener model [CAF03, SnA96] was chosen to identify the black–box model of the process given in the problem package.

1.4 Nonlinear State Estimation

Solving a set of first–order differential equations of the model requires initial conditions (states) to be known. The initial conditions are obtained from the measured information of the process. In model based control algorithm this is repeated in every online optimization step. Further, in many processes the state variables are unmeasurable and requires some form of estimation of their values at every calculation step. The Kalman filter [Kal60] is one of the commonly used tools for this task in the case of linear control. For nonlinear control an extended version (EKF) [LR94] can be devised by linearizing the problem every time step. Extended Kalman filter involves a time–varying linearized model which requires the calculation of Jacobian matrices at each time step. In this work, this is done accurately and automatically via the AD tool. In some cases as when applying the offset free algorithm (using output integrated disturbances) to be described in the next section, simple update of the states from previous values was found to be adequate as the controller steered the plant much nearer to the steady–state conditions all the time. The change in the states becomes small in this case and simple update becomes sufficient.

1.5 Offset–free NMPC

Any real control situation must consider possible unmeasured disturbances that might enter the process while under control. Unmeasured disturbances often produce errors between the model predictions and plant output because the actual value of such disturbance is seen by the plant but not by the controller [QB97, Hen98]. Measured disturbances are less of a problem as their effect can be sensed and the appropriate control action taken. It is more difficult to handle unmeasured disturbances because neither their magnitude nor their points of entrance to the control loop are known. Special methods are usually devised to deal with the problem. Simpler cases like zero mean disturbances may be handled using filters like that of the extended Kalman filter. However for persisting disturbances an offset (steady–state) error could be produced in the final output response of the plant. Therefore a different procedure of disturbances estimation is needed [RMM94, QB97, KC01].

In this thesis two methods based on adaptive nonlinear integration are devised and shown to work very well in all tested control situations. In the first method, one or more of the unmeasured disturbances are chosen as adaptation parameters. The estimated value of these disturbances are then calculated by integrating the current output tracking error along one sample time. The estimated values of the unknown disturbance are also applied to the nonlinear model in the EKF stage to correct the model/plant mismatch.

In the second variation to the above method, an output virtual disturbance is created from the current measurement using the same adaptive algorithm of the method above. These disturbances are then added to the model outputs to compensate its steady–state shifting from the real steady–state due to model/plant mismatch. The EKF block can as a result, be removed from the control loop and replaced by a simple state estimation method using state updates. Both input and output disturbance modelling methods above can work together to produce offset–free NMPC performance in some difficult cases.

The proposed offset removal using the output disturbance model is simple in design and can efficiently eliminate offset errors produced from different sources (unmeasured disturbances, modelling error, or both) which enhances the NMPC robustness and stability.

1.6 Applications

1.6.1 Evaporator Process

In this thesis, an evaporator process is chosen to test various proposed controllers. The first–principle equations of the process are known, and are given in a state–space formulation comprising three inputs and three outputs. All the states are measurable in this case. The same plant was then treated as black–box with input–output data collected from simulation and then a nonlinear model was constructed for the process. The proposed CTRNN is used in this case. The results are tested for robustness against large and varied known and unknown disturbances, modelling error, and large changes in the set points. Performance was also compared with results from other published control algorithms.

1.6.2 Two CSTR Process

A process consisting of two–CSTR was chosen as the second application . This problem has two control configurations S1, and S2. The first–principle model equations are also available. The process is nonlinear process with two inputs, two outputs, and six states. A state estimator is needed in this case as four state variables are not measurable. The internal model for the controller as in the previous case was constructed using the model equations or using a black–box CTRNN model derived from input–output data collected from the simulation of the first–principle model. Results are then tested for robustness and response with other controllers such as PID, linear optimal control, and LMPC. Excellent results were obtained using the proposed NMPC algorithm with the help of the extended Kalman filter and the offset removal algorithms.

1.6.3 The ALSTOM Gasifier

This is a challenge case, issued by the ALSTOM company [Dix99]. The first–principle equations are not available for the process. The process is simulated by a MATLAB/SIMULINK model. The model is nonlinear with five inputs, four outputs, and 25 states. A few published works have addressed this problem because of its importance for a clean and environmentally friendly way to burn coal. The process has very stringent quality and safety constraints that makes it an excellent candidate for model predictive control. The process is also characterized by widely varied response time constants between the output and the various inputs. Furthermore, the operation points for the plant can be varied between zero and full load conditions. The controller for this case should be able to take all these facts into consideration. Different control strategies for different input/output routes are attempted in this case. LMPC and NMPC approaches were mixed to reach the best control strategy [ACY04, ACY06, ASC05a, ASC05a]. A system identification step is performed before hand to specify the internal model in the controller using a state–space class Wiener model. A feedforward neural network is used here for modelling the static part for one of the output variables the gas pressure. Linearization of the neural network is performed at every sample time and the AD tool is used to calculate the derivatives required for this local linearization. Linear quadratic programming (QP) is used in the optimization section of the controller. All the challenge requirements are satisfied using the designed LMPC and NMPC algorithms.

1.7 Thesis Organization

The thesis is organized as follows. After an introduction in **chapter 1**, background material and survey of the relevant literature in the area of MPC and mainly NMPC, are given in **chapter 2**.

In **chapter 3** the AD tool is introduced in its various version and examples are given on how to use it for calculating partial derivatives from the a code based formulation of a multivariable function.

In **chapter 4**, the algorithm for a new NMPC strategy is developed. The procedure to include the AD in the algorithm is also given. The model equations are integrated numerically and the sensitivity equations required to solve the optimization problem are calculated using a linear (first–order Taylor) approximation. The nonlinear least squares optimizer is used to solve the online nonlinear programming problem of the predictive controller. An evaporator process is chosen to demonstrate the proposed NMPC algorithm in different operating tests.

In **chapter 5** the proposed NMPC algorithm is extended to include a state estimate stage. The algorithm to implement the state estimation in the form of EKF using the AD tool is given in this chapter. The two–CSTR process with two control configurations S1 and S2 are used as case studies to examine the developed NMPC with the state estimator.

Chapter 6 includes the development of two alternative parameter adaptation techniques to remove the offset error observed in the NMPC performance as a result of unmeasured disturbances or/and modelling error. The two technique include an integrated disturbance which can be used in the model input or output to shift the model steady–state targets to the correct values. The nonlinear integration is used here for fast and smooth responses during the offset error rejection time. The evaporator and the two–CSTR processes are used as examples in this chapter.

In **chapter 7**, the case of black–box model is considered. The continues–time recurrent neural network is introduced as a system identification method for the internal model of the proposed NMPC algorithm. The development of a new efficient training algorithm for the CTRNN model via the AD tool is presented. In this algorithm, the AD tool is used to calculate both the Taylor series coefficients and the sensitivity equations required to solve the NLP problem of the network training. A similar approach is then used to solve the online optimization problem of the predictive controller. The evaporator process and the two–CSTR process are used as case studies with satisfactory results.

In **chapter 8** the benchmark ALSTOM gasifier plant case is studied. Using the black–box model provided with the problem, a linear state–space model is derived first. Different to previous attempts to control this problem, the linear model is correctly selected based on the zero–load operating point. A LMPC algorithm based on a state–space formulation is derived to control the plant as a first trial. Then, the same controller is extended to NMPC approach via model linearization. Further improvement to the controller was achieved using a nonlinear model for one of the plant outputs, the gas pressure. A Wiener type model is developed for this case. Linearization at every time step is needed and the AD is employed in this case. The controllers are tested in different operating conditions and the results are discussed.

Further discussion of the performance results, conclusions, and recommendations for further work are given in **chapter 9**.

Chapter 2

Background Material and Literature Survey

2.1 Introduction

The field of control is truly vast whether on the theoretical side or application side. This is rightly so for a field that forms a back–bone of the present civilization and beyond, MPC is only a recent addition to the field appearing in the literature of the process industry around 1978 [RiR78] where it was then called dynamic matrix control (DMC) [CuR80]. There were no major theories backing the early algorithms, as the structure of this type of control is intuitive. A good example of MPC is a car driver who sees a difficult situation ahead, uses his knowledge of the car (the internal model) to decide on some optimized moves (control horizon) to keep to a set of responses (prediction horizon) and achieve a certain goal (the objective function) to overcome the road difficulty. After every move the whole situation is reassessed and a new set of moves (receding horizon) are decided, taking into account the new position of the car together with any changes in the objectives or the problem constraints.

Literature describing theoretical and practical issues associated with MPC technology are summarized in several recent review articles. Qin and Badgwell present a brief history of MPC technology and a survey of industrial applications in [QB97]. Meadows and Rawlings summarize theoretical properties of MPC algorithms in [MR97]. Morari and Lee discuss the past, present, and future of MPC technology in [ML99].

What is needed to perform the above control task is clearly a huge computing capability and speed. This is why MPC started making fast progress only in parallel with available computing power. The computing problem involves the solution of the internal model equations, which are in general nonlinear with many variables. It also involves the solution of an additional and probably larger set of nonlinear differential equations of the sensitivity needed in the optimization step. This has to be done online before the process gets out of control and probably in a MIMO context with a large number of inputs and outputs and a long prediction horizon. It is clear therefore that a considerable effort is needed to tackle and reduce this computational burden. In addition to computation efficiency, studies in the literature have addressed robustness, stability and other issues related to the solution of a large number of nonlinear differential equations.

The next sections will introduce briefly the literature and background material on linear model predictive control first and pass–on to NMPC. Literature on ways to construct the internal model, whether using the process equations or the process input–output data, is also discussed. The issues of controller algorithm, the solution of the nonlinear model equations and sensitivity equations are discussed next. This is followed by discussion of literature on other important issues like the structure of the objective function, robustness and stability against unpredictable and unmeasured factors. The final part is devoted to the important tool of AD and the algorithms to implement it in the control and in the solution of the nonlinear model equations.

2.2 MPC Framework and LMPC

Model predictive control is an attractive feedback strategy, especially for linear and nonlinear systems subject to input and output constraints. The MPC algorithm in general, consists of the following three steps:

1. Explicit use of a model to predict the process output along a future time horizon (prediction horizon).
2. Calculation of a control sequence along a future time horizon (control horizon), to optimize a performance index.
3. A receding horizon strategy, so that at each instant the horizon is moved towards the future, which involves the application of the first control signal of the sequence calculated at each step.

The first predictive control methods were based on linear dynamic models. With a few exceptions, linear MPC methods mostly employ a *discrete–time* model of the process.

The major cause of this is that discrete–time models are known to be practical for implementation on a digital computer. However, in NMPC there is a significant group of methods based on continues–time models, because such models result from first–principles modelling procedures. Discrete–time models, are also obtained as a result of black–box or gray–box system identification. One major advantage of continuous–time models however is that it allows a free choice of the time step without degrading the model performance especially for the black box–model class.

The term LMPC refers to a family of MPC schemes in which linear models are used to predict the system dynamics, even for the case when the dynamics of the closed loop system is nonlinear due to the presence of constraints for example.

LMPC approaches are successful for many applications in a very wide range of processes [QB97, RiR78, CuR80, GM82, CH88] and including the initial versions of MPC e.g. DMC by [CuR80] or Generalized Predictive Control (GPC) by [CMT87].

LMPC is probably acceptable and sometimes desirable when the process operates at a single setpoint and the primary use of the controller is the rejection of small disturbances [PiS00]. Many systems are, however, inherently nonlinear. This, together with higher product quality specifications, increasing productivity demands, tighter environmental regulations and demanding economical considerations in the process industry require operating systems that can operate closer to the boundaries of the admissible operating region [ML99]. Such conditions are also met in many cases including product change over in continuous processes, tracking problems in startup and batch processes and the control of nonlinear reactors [Bie98, PiS00]. Because these processes make transition over the nonlinear range of the system, LMPC often results in poor control performance [Mac02, RAB03].

2.3 Nonlinear MPC

To properly control nonlinear processes, a nonlinear dynamic process model must be used. Recognizing this need, a number of NMPC algorithms incorporating nonlinear prediction models have appeared in the literature; e.g. Nonlinear Quadratic Dynamic Matrix Control (NLQDMC) by Garcia [Gar84], and other types described in [CW91, BS89, LGM92, SaA01, DiB02].

The constrained optimization problem can be expressed in the form:

$$\min_{u} V(k) \tag{2.1}$$

subject to process dynamics

$$\begin{aligned} x_{k+1} &= F(x_k, u_k) \\ y_k &= g(x_k, u_k) \end{aligned} \tag{2.2}$$

and constraints:

$$\begin{aligned} u_{min} &\leq u_k \leq u_{max} \\ \Delta u_{min} &\leq \Delta u_k \leq \Delta u_{max} \qquad (2.3) \\ y_{min} &\leq y_k \leq y_{max} \qquad (2.4) \end{aligned}$$

where x is the process state vector, u is the manipulated variable vector, y is the process output vector, $f(\cdot)$ and $g(\cdot)$ are nonlinear functions, $\Delta u := u_{k+1} - u_k$ is the manipulated variables increments, and V is the objective function.

Typically, the criterion objective function $V(k)$ is a quadratic function of the error between the predicted output and the desired trajectory (reference) over the prediction horizon, and usually includes terms which penalize the control effort and the rate of change of the control variable. Following the notation in [Mac02], a typical criterion function can be written as:

$$V(k) = \sum_{i=i_d}^{P} ||\hat{y}_{k+i|k} - r_{k+i}||_Q^2 + \sum_{i=i_d}^{M} ||u_{k+i|k}||_R^2 + \sum_{i=i_d}^{M-1} ||\Delta u_{k+i|k}||_S^2 \tag{2.5}$$

where i_d is a number of time steps representing the process delay time, $\hat{y}_{k+i|k}$ denotes the output prediction until time $(k+i)$, made at time k, r_{k+i} denotes the value (or an estimate) of the reference at time $(k+i)$, u and Δu denote the manipulated variable and the change of the manipulated variable. Subscripts Q, R, and S refer to the matrices which are positive semi–definite diagonal weighting matrices, and $||x||_W \equiv \sqrt{x^T W x}$ denotes the weighted 2–norm of vector x. The weighing matrices Q, R, and S, as well as the prediction horizon P and the control horizon M are design parameters that must be tuned to reach a satisfactory performance in the controller.

2.4 Computational Approaches in NMPC

In NMPC context, there exist a number of strategies for tracking the optimal control problem through nonlinear programming (NLP) [Hen98]:

1. Successive linearization or Instantaneous (Jacobian) linearization method, [GZ92].

2. Multiple model MPC approach.
3. Sequential method [Mor86, EMP86, JJM87, Beq91a, Beq91b, SB91].
4. Simultaneous method, [PRE90, ER90, PE93, LP94].

2.4.1 Successive Linearization Method

A number of researchers have developed MPC approaches based on a linearization of the plant model for the prediction phase. In this solution, the model equations are linearized around the operating point and then apply one of the many linear optimization methods available in the literature such as QP routine to obtain the optimum control step. The linearization step is performed over all the prediction horizon [Mac02, KR03], or at a number of time steps in the prediction horizon [BS89], [GZ92, ZB02]. This method is very useful when the nonlinear model is extremely difficult to identify. This strategy is highly successful in controlling mildly nonlinear processes [NPR98].

2.4.2 Multiple Model Schemes

Model varying predictive control (MVPC) or multiple model predictive control (MMPC) are approaches that are also based on the model linearization principle [ZhX03, RAB03, AB03]. In addition, it is possible to sub–divide the operating range into sub–ranges and use a single model within each sub–range as the prediction model in MPC. This results in a family (or bank) of linear models obtained by linearizing the nonlinear equations in several operating points. Based on these local multiple models, a controller design is carried out at the different operation regions. This allows the use of simple linear models to represent a nonlinear system and then design systematic controllers. There are many advantages to use multiple models, such as their flexibility in selecting the modelling methods (*e.g.* transfer function and state–space), different presentation (*e.g.* continuous–time and discrete–time) [JF95, Joh95] and other cases such as noise and disturbance reduction. Moreover, this method can be used for online control of systems with high speed and accuracy [GL00, NBC95, PSR97]. The computation speed gain comes from the fact that the linearized model is ready off–line thus eliminating the need of linearization calculations at every time step.

An extension of DMC to multiple models approach was proposed by Zhao et al. [ZhX03] and further developed to handle different operating regions and input disturbances by Aufderheide et al. [AB03]. This is done by using a multiple model framework of step response models. Two different model banks were tested in [AB03]; one uses actual step response for the different operating conditions and the other is a minimal knowledge based approach using only first–order plus dead time models (FOPDT). Different operating regions and disturbances are handled by the overall model bank switching to a more appropriate model(s) using a switching control algorithm [ZhX03], or Recursive Bayesian algorithm which assigns weights to each model [AB03].

2.4.3 Sequential Method

To solve the MPC problem, it is necessary to both solve an optimization problem and solve the system model equations. These two procedures may be implemented either *sequentially* or *simultaneously.*

For sequential solutions, improved closed–loop performance is achieved because the nonlinear dynamical model is directly used in the NMPC calculations. Standard NLP however is not designed to handle dynamic constraints. This limitation is overcome by using a two-stage approach wherein an optimization routine serves as the outer loop which iteratively select new sets of manipulated variables moves, while a differential equation solver is used to integrate the dynamic equations at each optimization step [MLL86, Beq91a, Hen98, Beq91b]. In this solution only the control signals are the decision variables. It is sequential because the optimization and integration problems are solved iteratively to obtain a certain accuracy.

Most efficient methods for solving nonlinear programs require the evaluation of partial derivatives of the objective function with respect to the decision variables (sensitivity equations). In order to take advantage of the efficiency of these methods, it is important to be able to obtain gradient information efficiently and accurately.

Traditionally, there are three ways to calculate the sensitivity information of a dynamic system [StH99]: *perturbations*, *sensitivity equations* and *adjoint equations.* In a perturbation approach finite differences are used to approximate derivatives. Hence it need at least applying N perturbations of the dynamic system to get the solution of a N–parameter sensitivity problem. Also, derivatives accuracy is difficult to determine and numerical errors can grow excessively with problem complexity. Using numerical derivatives obtained through finite differences in the sequential approach has been reported to be with strongly negative results [HS97]. Differencing the output of an integration routine with adaptive step size, although tightly bounded by the integration algorithm, the integration error is unpredictable. Alternatively, sensitivity variables can also be obtained by simultaneously solving the original ODEs together with nN sensitivity equations, where n is the number of states [JJM87, CJ87, JJM89, ScM04]. Finally, sensitivity can be calculated by solving n adjoint equations (in reverse direction). A number of efficient solvers have been developed to tackle the dynamic sensitivity problem, for example DASAC in Fortran language [CS85], and CVODES package in C language [SH03].

Joseph et al. integrate the model ODE's and sensitivity equations for the optimal solution [JJM89]. The prediction and control horizons are the same length. They used a first–order filter for the setpoints, which eliminates the deadbeat type of control that normally results when the prediction horizon and control horizon are the same.

Morshedi (1986) presents an extension of QDMC called universal dynamic matrix control (UDMC), [Mor86]. The modelling equations are integrated using a nonlinear ODE solver, while the mean–value theorem is used to develop a linear analytical solution to the sensitivity equations in order to reducing the computational burden significantly. In the starting of this thesis a first–order approximation was derived using AD to simplify the dynamic sensitivity equations associated with a NMPC problem so that computation efficiency was improved [CAS03]. It is similar to the Morshedi approach above but using AD.

Recently, the AD techniques have been applied to tackle the dynamic optimization problem [GW04, Cao05]. In these approaches, AD tool have been used to solve ODEs and sensitivity equations using high–order Taylor series in a NLP problems. In the second part of this work, the approach of [Cao05] is extended to solve both the nonlinear model identification problem (training a dynamic recurrent neural network) and NMPC control problem to speed up calculations and to increase efficiency [ASC05c, ASC06].

2.4.4 Simultaneous Method

In the simultaneous solution [ER90, PE93, SK99] the differential equations are transformed to algebraic equations which are solved in addition to the nonlinear equality constraints in the optimization. The decision variables includes both the model states and control signals and the model equations are appended to the optimization problem as equality constraints. This can greatly increase the size of the optimization problem, leading to a trade–off between this approach and the sequential one. A major advantage of the simultaneous approach is that the state and output variables constraints are easily handled. However, the major disadvantage to an infeasible path approach, such as sequential quadratic programming (SQP) is the termination may occur at an infeasible point. This may be all right in off–line strategies, where a new initial guess could be made and the solution performed again. In online strategies, some outer layer type of approach must be used to detect such a failure [Beq91b]. The simultaneous approach is well suited for large NLP problems with state/output constraints [ER90, PRE90].

In general, the excessive computational burden appears to be the main disadvantage of the NMPC strategies that resort to the direct use of an ODEs solver and nonlinear optimization. The requirement of computing the gradient information at each iteration requires repeated integration of ODEs which can be considerably demanding. In fact, the calculation time of a large number of dynamic sensitivity equations required to solve any NLP problem in the NMPC could take more than 70 percent of the total computation time of the optimization problem. Hence, dynamic sensitivity calculation is the bottleneck of solving any dynamic optimization problem.

2.5 Internal Model Formulation

The basic control strategy in MPC is the selection of a set of future control moves (control horizon) and minimize a cost function based on the desired output trajectory over a prediction horizon with a chosen length. This requires a reasonably accurate internal model, that captures the essential nonlinearities of the process under control, to predict the dynamic behavior multi–step ahead [PeR03]. That makes the heart of MPC.

The NMPC schemes proposed in literature use models developed from first–principles [BS89, PE93] or models identified from input–output data (black–box model or empirical model) [DFL94, SMc97, ZGS98]. Both approaches have been used successfully in MPC applications. A third type of model which is a mixture of the two types of models above and it is called *hybrid models* which can be also found in a number of applications.

The various model forms used in this thesis are derived as a special cases from a general continuous–time nonlinear state–space model:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= g(x(t), u(t)) \end{aligned} \tag{2.6}$$

where, $u \in \mathbb{R}^{n_u}$ is a vector of manipulated (control) variables, $y \in \mathbb{R}^{n_y}$, is a vector of an output variables, and $x \in \mathbb{R}^{n_x}$ is a vector of state variables.

This general form has been extensively used for control system analysis and design for the cases of both white–box and black–box situations. It has many attractive characteristics [ZGS98]:

1. It is a compact model consisting of fewer parameters than a corresponding Finite Impulse Response (FIR) or Finite Step Response (FSR) models.
2. It provides a standard basis on which the MPC analysis and design can be performed in a framework parallel to the linear quadratic optimal control.
3. It can capture full nonlinear dynamics.
4. Since many excellent theoretical results for stability properties of NMPC and robust design are based on state–space model, their implementation will be straightforward if the state–space models are available.

In the following sections, few model types commonly used in NMPC are discussed in more details.

2.5.1 Linear Internal Models

In almost any control application, linear design techniques are usually the first to be attempted and are completely satisfactory for many engineering applications, especially those involving regulation about a steady–state operating point.

Linear models have been used in the majority of MPC applications to date. A wide variety of model forms are used, but they can all be derived from system (2.6) by linearizing about an operating point and discretized in a sampling time T_s to get:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{2.7}$$

An equivalent discrete–time, transfer function model, autoregressive model with exogenous inputs (ARX), FIR model, and FSR model can be also derived from (2.6) (see [QB03]).

MPC based on linear models often results in poor control performance for highly nonlinear processes because of the inadequateness of a linear model to predict dynamic behavior of a nonlinear process. There is therefore, a strong requirement of a good fitting model for NMPC applications.

2.5.2 First–Principle (White–box) Nonlinear Models

In many practical applications, a restricted mathematical model based on physical principles is developed based on process knowledge, usually constructed from mass and energy balances, and written to take the form of (2.6). This type of model is often referred to as a *first–principle model*, or *white–box model*, or *mechanistic model* in some publications.

A general discrete–time first principle model that can be derived from system (2.6) by integrating across the sample time can be written as:

$$\begin{aligned} x_{k+1} &= F(x_k, u_k) \\ y_k &= g(x_k, u_k) \end{aligned} \tag{2.8}$$

A major advantage of first–principle models is that they can be used to predict over a wide range of conditions, even without prior operating experience, provided that the basic assumptions of the model remain valid. On the other hand, a model based on (for example) artificial neural networks has almost no predictive value outside the range of operating conditions where data has been collected. However, the development of these type of models is usually time consuming and effort demanding, especially for complex processes. For some poorly understood processes, it is even impossible to build first–principle models or the resulting model is too complicated to be used for control [ZM99].

2.5.3 Empirical (Black–box) Nonlinear Models

This type of models is based on using measurements of input and output signals from the true system using *nonlinear system identification* and the resulting model is called an *empirical model* or *black–box model*.

For the nonlinear case, the decision to use first–principle or empirical models is less clear. For small systems with well–understood physical phenomena, fundamental modelling is preferable, because of the ability of the model to predict beyond the range of existing operating data. On the other hand, model identification is easier to use for larger systems using black–box models [HS97].

In many cases, nonlinear system identification is an inevitable step in a NMPC project. Possibly, it is also the most costly and time consuming part of the project [ZGS98]. Therefore, an efficient and affective approach of nonlinear system identification is critical to the success of NMPC.

A fundamental difficulty associated with the nonlinear input–output data (or nonlinear empirical) modelling approach is the selection of a suitable model form. Discrete–time models are most appropriate because plant data is available at discrete–time instants and NMPC is most naturally formulated in discrete–time.

Unlike linear system identification, there is no uniform way to parameterize general nonlinear dynamic systems. Some of the more classical identification method are given bellow. The types of nonlinear models utilized for NMPC include;

1. **Volterra models** [MaD96].
2. Polynomial auto regressive moving average model with exogenus inputs (polynomial **ARMAX**) [SrA97].
3. **Hammerstein** and **Wiener models** [CXS94, DFL94].
4. **Artificial Neural Network (ANN) models** [SMc97].

Voltera models can be described by the following input/output relationship [PeR94, MaD96];

$$\begin{aligned} y(k) \quad = \quad & y_0 + \sum_{j=0}^{\infty} a_j u(k-j) + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} b_{ij} u(k-i)u(k-j) + \\ & \sum_{l=0}^{\infty}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} c_{lij} u(k-l)u(k-i)u(k-j) + \cdots \end{aligned} \tag{2.9}$$

where, a, b, and c are the model parameters. Volterra series models can be used to model a wide class of nonlinear systems however, these models are non–parsimonious in parameters and in turn, difficult to use for modelling MIMO systems [Sah04].

NARMAX model provides a description of the systems in terms of a nonlinear function of delayed input, output, and prediction error. Many MPC based on NARMAX models were proposed in the literature [KP91, Mat91]. The NARMAX form for a single–input single–output (SISO) system is:

$$y(k) = F\left(y(k-1), \cdots, y(k-n), u(k-1), \cdots, u(k-m), e(k), \cdots, e(k-q+1)\right) \tag{2.10}$$

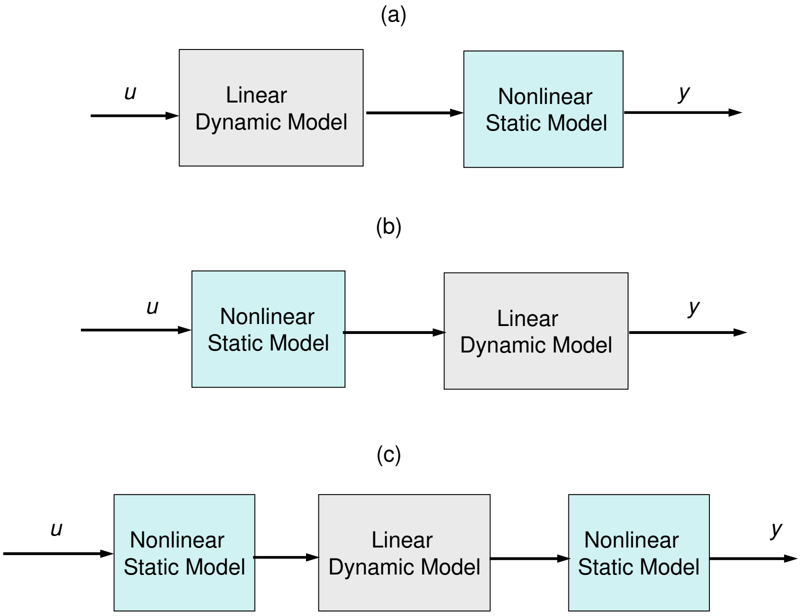

Figure 2.1: Basic structure of nonlinear models, (a) Wiener model, (b) Hammerstein model, (c) Hammerstein–Wiener model.

where, F is a nonlinear mapping which can be a polynomial nonlinearity, neural network structure, or others, k represents the time instant $k\Delta t$, where Δt is the sampling time, y is the controlled output, u is the manipulated input, e is the noise input, n is the number of past outputs used, m is the number of past inputs used, and q is the number of current and past noise inputs used.

The main difficulty with NARMAX or its other variants even for SISO systems, is the determination of an appropriate model structure ($i.e.$ n, m, and q) that best represents the process dynamics.

Two basic types of nonlinear empirical models have been used in this thesis to identify the black–box process. These are a discrete–time state–space class Wiener model, and a continues–time state–space recurrent neural network model. More details about each type of these models are given in the following subsections.

2.5.4 Modelling with Wiener Models

Hammerstein and Wiener models have a special structure that facilitate their application to NMPC [DFL94, CAF03, SnA96]. *Wiener models* are particularly useful in representing the nonlinearities of process without introducing the complications associated with general nonlinear operators. These models consist of a linear dynamic element followed in series by a static nonlinear element. *Hammerstein model* contain the same elements in the reverse order [NPR98]. Also, the *simultaneous model* contains a linear dynamic model sandwiched between two memoryless nonlinear gain (see Figure 2.1). These models correspond to processes with linear dynamics but nonlinear gain, and can adequately represent many of the nonlinearities commonly encountered in industrial processes such as distillation and pH neutralization [ZGS98]. Due to the static nature of their nonlinearities, they can be effectively removed from the control problem. This fact generalizes the well–known gain–scheduling concept for nonlinear control [CAF03].

The Hammerstein and Wiener type models are relatively simple models requiring little effort in development than a standard linear model, yet offer superior characterization of systems with highly nonlinear gains. Wiener models may be incorporated into MPC schemes in a unique way which effectively removes the nonlinearity from the control problem, preserving many of the favorable properties of linear MPC [NPR98]. Norquay et al. examined the characteristics of Wiener models as well as various methods of system identification (ARX and step–response models) and final model validation [NPR98]. They have been presented as an effective way of introducing nonlinearity to a control problem without the significant increase in complexity usually associated with nonlinear MPC. They reported that, NMPC based on Wiener model was shown to effectively control the highly nonlinear pH neutralisation process with excellent setpoint tracking and disturbance rejection capabilities as compared to simple LMPC and PID control.

Zhao et al. [ZGS98] presented an approach to identifying linear/nonlinear state–space class Wiener models for MIMO MPC. A hybrid linear and neural network model structure was used in the static part of the model. It is shown that this type of nonlinear models can approximate any discrete–time nonlinear processes with fading memory [SBG98].

A problem of identification and control using a Wiener model was proposed by [AKC96]. They proposed a hybrid model consisting of an ARMA model as a linear dynamic model in cascade with a multilayer feedforward neural network. They also suggested control using the Wiener model by inserting the inverse of the static nonlinearity in the appropriate loop locations. They demonstrated the effectiveness of their proposed identification and control algorithm using simulation results.

Wiener–Laguerre nonlinear model is a special case of a Wiener model, where the linear dynamic part is represented by a series of orthonormal *Laguerre* filters followed by a memoryless nonlinear mapping. Dumont et al. have used a model of this type for developing an adaptive MPC scheme for controlling SISO nonlinear systems [DFL94]. Recently, multivariable extension of the Wiener–Laguerre model have been proposed by Sentoni et al. [SBG98] and Saha [Sah99]. In both cases, the MIMO systems were modelled by several multi-input single–output (MISO) models. Sentoni et al. [SBG98] used artificial neural networks (ANNs) for constructing a nonlinear state output map while Saha [Sah99] used quadratic polynomials as well as ANNs for constructing a nonlinear output map and used these models in NMPC formulation.

Wiener–Laguerre model can be used to approximate the dynamic response of the process only for open–loop stable systems. To handle open–loop unstable systems Wiener–Kautz model was proposed by Wahlberg [Wah94]. This is done by replacing Laguerre filters with *Kautz* orthonormal filters [LW93]. Recently, Abrahantes et al. [AVA01] developed a nonlinear MIMO state–space model which is similar to Sentoni et al. [SBG98] but they used Kautz filters instead of Laguerre filters in order to approximate open–loop unstable systems.

The modelling scheme, based on a nonlinear extension of Laguerre filter modelling, appears promising as the nonlinear model inherit all the merits of its linear counterpart i.e. it requires no prior knowledge of plant order and time-delay and also has a flexible structure so that the model complexity can easily be changed online [DFL94, SJ97, GeJ00].

Recently, Huzmezan et al. (2002) designed an adaptive MPC approach based on Wiener–Laguerre model to handle integrating type processes with long dead times and long time constants [HuG02] . A system identification approach and NMPC algorithm based on Wiener model is developed to control a pH neutralistion process by Gomez et al., [GJB04]. The performance of the proposed Wiener model predictive control (WMPC) was compared with that of LMPC and PID controllers and the results show that the WMPC outperforms the other two controllers.

2.5.5 Modelling with Artificial Neural Networks

Neural networks hold great promise for solving problems that have proven to be extremely difficult for standard digital methods [BM90]. The universal approximation properties of the neural networks makes them a powerful tool for modelling nonlinear systems [BM90, FN93]. There are two key advantages of using the neural networks;

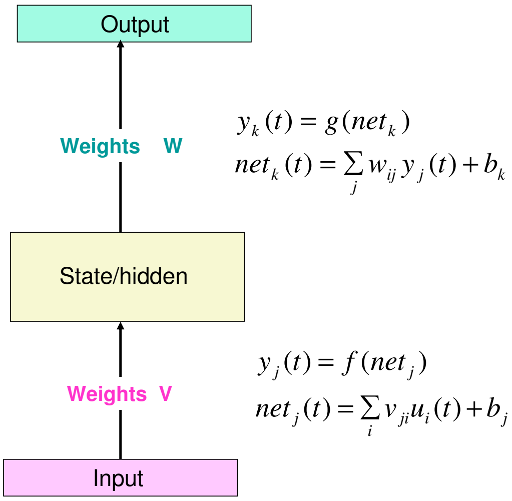

Figure 2.2: Simple Feed Forward Neural Network

1. Neural networks are inherently parallel machines and as a result they can solve problems much faster than a serial machine.

2. Many neural networks have the ability to *learn*.

Neural networks generally are composed of neurons processing elements, termed *nodes*, which are arranged together to form a network. The most commonly used processing element is one which weights the input signals and then sums them together with a bias term. The neuron output is then obtained by passing the summed, weighted inputs a nonlinear activation function such as the hyperbolic *tanh*. In this case, the network will be called feedforward neural network (FFNN) (see Figure 2.2). The incorporation of a dynamic element into the network is important for the modelling of dynamic data, so that some of the outputs are fed back to the network input as the network dynamic states. In this case, the network will be called a dynamic or recurrent neural network (RNN) type (see Figure 2.3).

Different neural networks models embedded in MPC systems are reviewed by [Hen98]. An advantage of the use of ANNs over a polynomial NARMAX models in MPC is that the structure of the ANNs models is easier to develop, which is particularly true when applied to the modelling of MIMO processes [YG02]. The application of ANNs based model predictive control scheme is given in [DER95].

It is has been shown [BM90] that conventional FFNN can be used to approximate any spatially finite function given a set of hidden nodes. That is, for functions which have a fixed input space there is always a way of encoding these functions as a neural networks. For two–layered network, the mapping consists of two steps;

$$y(t) = G(F(u(t))) \tag{2.11}$$

It can use automatic learning techniques such as backpropagation to find the weights of the network (*i.e.* G and F) if sufficient samples from the function is available. Figure 2.2 show a simple diagram of a two layers FFNN. The first layer of nods is the input layer (*hidden* or *state* layer), and the second layer is the *output* layer respectively. Each layer will have its own index: k for output nodes, j (and h) for hidden, and i for input nodes. In FFNN, the input vector u is propagated through a weight layer V,

$$\begin{aligned} y_j(t) &= f(net_j(t)) && (2.12)\\ net_j(t) &= \sum_i^{n_u} u_i v_{ji} + b_j && (2.13) \end{aligned}$$

where, n_u is the number of inputs, b_j is a bias, v_{ij} is the ij element of the weight matrix V, and f is an output function (of any differentiable type).

The output of the network is in both cases determined by the state (the hidden layer outputs) and a set of output weights W;

$$\begin{aligned} y_k(t) &= f(net_k(t)) && (2.14)\\ net_k(t) &= \sum_j^{n_x} y_j w_{kj} + b_k && (2.15) \end{aligned}$$

where, g is an output function (possibly *linear* or the same as f), and w_{ij} is the ij element of the weight matrix W.

Most of the publications in nonlinear system identification use FFNNs with backpropagation or its other variations for training, for example [TSM95, TC96]. Successful applications of FFNNs to model chemical processes were reported by [BM90].

The static FFNNs together with tapped-delay lines provide a means to model nonlinear dynamic systems in discrete–time [MSW90, NP90, LzP02]. In general, this type of model (which is also referred to the nonlinear autoregressive with exogenous inputs (NARX) model) can be represented by the following mathematical form;

$$y(t) = f_{NN}(\varphi(t), \theta) + e(t) \tag{2.16}$$

where y is the output of the model, f_{NN} is a nonlinear function, θ is the network parameters, e is known as the model residual, and φ is known as the regression vector (which depends on past input and output information) and given as;

$$\varphi(t, \theta) = \Big[y(t-1), \cdots, y(t-n), u(t-m), \cdots, u(t-m+1)\Big]^T \tag{2.17}$$

where, n and m are the tapped–delay–line memory (order) of the output and input respectively. The main drawback of this approach is the input and output orders n and m are unknown in real applications. In addition, this type of model can only provide predictions for a predetermined finite number of steps, in most cases, only one step. This drawback makes such models not well suited for predictive control, where variable multi–step predictions are desired.

Recently, a NMPC based on multiple FFNNs has been proposed [Jaz04]. In this approach, a combination of multiple FFNNs with one hidden layer are used to model an m–input n–output nonlinear dynamic system. This system consists of a two–dimensional array of FFNNs blocks and each block consists of a one–step–ahead predictive neural model, which is identified to represent each output of the MIMO system. These models are employed to predict the future outputs over the prediction horizon of P time steps. This approach might solve the multi–steps ahead prediction problem of the FFNN but it needs the training of a new FFNN for every extension to the prediction horizon, or a large number of networks when a long prediction horizon is needed.

Recurrent neural networks on the other hand are capable of providing long range predictions even in the presence of measurements noise [SMc97]. Also, RNNs are more efficient than FFNNs and can do an equivalent job using fewer neurons [DKW95]. In some cases a small feedback system is equivalent to a large and possibly infinitely large feed–forward system [HH93]. Therefore, RNN models are better suited for NMPC.

RNNs with internal dynamics are adopted in several recent works. Models with such networks are shown [FN93, JNG95], to have the capability of capturing various plant nonlinearities. They have been shown to be more efficient than FFNNs in terms of the number of neurons required to model a dynamic system [DKW95, HH93]. In addition, they are more suitable to be represented in state-space format, which is quite commonly used in most control algorithms [ZV98].

RNNs are fundamentally different from FFNN architectures in the sense that they not only operate on an input space but also on an internal *state* space, a trace of what already has been processed by the network. The two–layered discrete–time RNN can

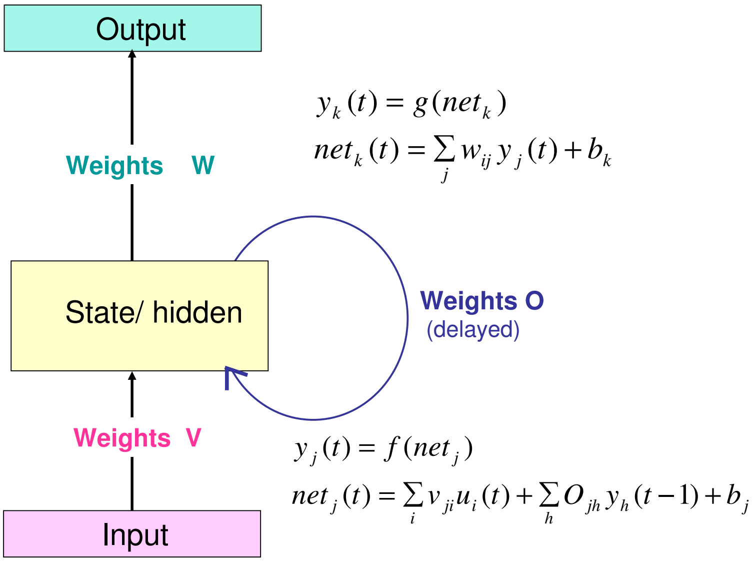
Output
Weights W
$y_k(t) = g(net_k)$
$net_k(t) = \sum_j w_{ij} y_j(t) + b_k$
State/ hidden
Weights O
(delayed)
Weights V
$y_j(t) = f(net_j)$
$net_j(t) = \sum_i v_{ji} u_i(t) + \sum_h O_{jh} y_h(t-1) + b_j$
Input

Figure 2.3: Simple Recurrent Neural Network.

be represented as;

$$x(t) = F(x(t-1), u(t); \theta) \tag{2.18}$$

$$y(t) = G(x(t)) \tag{2.19}$$

where, x is the internal state vector of the network, and θ is the vector of the network parameters (*i.e.* weight and bias elements).

In a simple RNN (see Figure 2.3), the input vector is similarly propagated through a weight layer, but also combined with the previous state activation through an additional *recurrent* weight layer, O,

$$y_j(t) = f(net_j(t)) \tag{2.20}$$

$$net_j(t) = \sum_i^{n_u} u_i v_{ji} + \sum_h^{n_x} y_h(t-1) x_{jh} + b_j \tag{2.21}$$

where, n_x is the number of *state* nodes.

RNN can be discrete–time neural networks, [ZV98], or continuous–time neural network (CTRNN), [FN93, KGW00]. In this case the difference equation (2.18) can be replaced by the following differential equation;

$$\dot{x}(t) = f(x(t), u(t); \theta) \tag{2.22}$$

$$y(t) = g(x(t)) \tag{2.23}$$

The CTRNN have some advantages over and are computationally more efficient than the discrete formulation even if in–the–end both are represented on the computer using only discrete values [PeB95]. In addition, the sampling period used with CTRNN can be varied without the need for re-training [FN93, KaC00, KGW00]. This is not possible in the case of discrete–time RNNs.

Although a continuous-time model has clear advantages compared with discrete–time RNN, it has rarely been used in NMPC. The main reason is due to the difficulty to solve the dynamic optimization problem associated with the continuous–time nonlinear model identification problem. So, the training of this type of neural network is difficult as reported by [PeB95, KaMC96, PBV03]. To solve the nonlinear optimization problem associated with CTRNN training, the calculation of a large number of dynamic sensitivity equations is required.

In this study, the CTRNN is chosen for the nonlinear process model approximation to be used in NMPC context. The network training difficulty is reduced by developing a new training algorithm using Taylor series expansion and AD tool.

2.5.6 NMPC based on Hybrid Models

Hybrid nonlinear models are developed by combining the first–principle knowledge with empirical modelling approaches. This allows the advantages of each modelling approach to be exploited. A common method for developing hybrid models is to use empirical (black–box) models to estimate unknown functions in the first–principle model [Hen98]. Another possible approach is to utilize a first–principle model as a nonlinear function of physical variables that generates other physically meaningful variables to capture the basic process characteristics, and then to describe the residual between the plant and the model using a nonlinear empirical model. Both techniques allow the nonlinear model to be constrained by the underlying physics, but they do not require a complete rigorous model of the plant. While hybrid models hold great promise, their use for NMPC design has not been explored [Hen98].

2.6 Tuning Parameters

The most significant tuning parameters of the NMPC that must be selected are the sampling period Δt (or T_s), the control horizon M, prediction horizon P, and penalty weight matrices. In fact, the choice of these parameters together with the feedback method if used, have a profound effect on NMPC nominal stability and robustness [HS97]. A limitation of NMPC is that the effect of these parameters on closed–loop performance is difficult to predict a priori. To date, parameter values which ensure nominal closed–loop stability have been determined only if the prediction horizon is infinite or a terminal state constraint is imposed. Because these conditions are rarely satisfied in practice, it is important to develop heuristic guidelines. The following results are summarized from [MR97] and [HS97].

For stable, and minimum phase systems, stability does not depend on the sampling period (time). However to ensure good closed–loop performance, the sampling period should be small enough to capture adequately the dynamics of the process. Small sampling period generally improve performance but require a longer prediction horizon to adequately capture the process dynamics which means an increase in the online computation time [Hen98]. On the other hand, a large sampling period reduces online computation, but it can result in poor performance such as *ringing* between sample points [GM82]. For unstable systems, robustness depends critically on the sampling period [HS97, Hen98]. There is an inverse relationship between T_s and the allowable modelling error. As modelling error increase, more frequent feedbacks of process measurements (i.e. small T_s) is required to indicate the onset of unstable behavior [Hen98].

Linear systems results [MuR93] indicate that shortening the control horizon M relative to the prediction horizon P tends to produce less aggressive controllers, slower response and less sensitivity to disturbance. For NMPC the effect of the control horizon on the closed–loop performance is similar. For fixed prediction horizon, smaller control horizons yield more sluggish output responses and more conservative input moves. Large control horizons have the opposite effect on performance. In addition, large values of M lead to increase the online computation as M is linearly related to the number of decision variables in the NLP problem. In practice, M often must be chosen to provide a balance between performance and computation.

The prediction horizon P has similar effects as the control horizon M. In fact, nominal stability is strongly affected by the prediction horizon length. However, the advantages of longer P are outweighed by the increase in computation time and result in more aggressive control [Hen98].

The weighting matrices Q, R, and S in equation (2.1), can be the most difficult tuning parameters to select because their values depend on the scaling of the problem. Typically, they are chosen to be diagonal matrices with positive elements [Hen98].

The magnitude of the diagonal elements depend both on the scaling and the relative importance of the variables. For problems in which all the variables are scaled similarly, it is suggested in Meadows and Rawlings [MR97] that the output penalties Q be chosen in the range 1–100 and the input penalties R and S be chosen in the range 1–10. The final parameter values can be obtained by fine tuning via simulation study. In this thesis these tuning ideas examined in the account in the NMPC design for a number of case studies.

2.7 Stability Issues in NMPC

The first property of a control system that should be satisfied is nominal stability, i.e. stability for systems free from modelling errors or disturbances. For linear systems without constraints, stability can be verified by checking eigenvalues of the closed–loop system. For nonlinear systems, there is no simple equivalent criterion.

Bitmead et al. [BGW90] showed that the general form of MPC does not guarantee closed–loop stability, because a *finite horizon* criterion is not designed to deliver an asymptotic property such as stability and closed–loop stability can only be archived by a suitable tuning of design parameters such as prediction horizon, control horizon, and weighting matrices. In the LMPC case, the infinite horizon controller can be reformulated as a finite horizon controller with a terminal state penalty [MuR93]. For NMPC case, Bitmead et al. [BGW90] suggested an *infinite horizon* method (*i.e.* $P \to \infty$ (closely related to linear quadratic (LQ) control), which however, results in an optimization problem that can generally be solved only for unconstrained linear systems. However in practice the solution of the optimization problem with $P = \infty$ can not be obtained or is computationally demanding [**?**]

Rawlings and Muske [RM93] propose a receding horizon control scheme with *infinite* prediction horizon and finite control horizon. Mayne and Michalska [MM90] showed that the finite horizon constrained optimal control problem can be posed as minimizing a standard quadratic objective function as the following form;

$$V(k) = ||x_{k+P|k}||_W^2 + \sum_{i=i_d}^{P} ||x_{k+i|k}||_Q^2 + \sum_{i=i_d}^{M} ||u_{k+i|k}||_R^2 \qquad (2.24)$$

subject to an additional *terminal state equality constraint* requiring the states to be zero at the end of the finite prediction horizon *i.e.* $x(t+P) = 0$. Note that the above objective function is differs from the prototypical formulation of (2.5) in that;

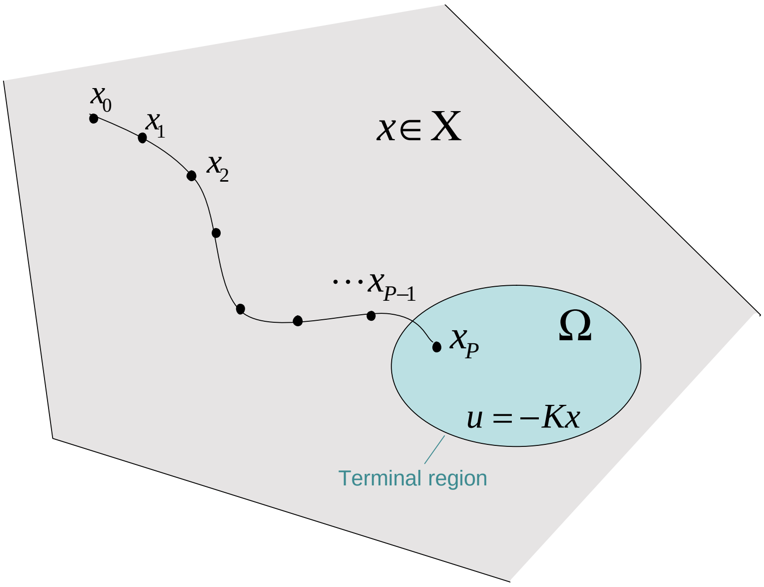

Figure 2.4: Suboptimal NMPC control.

(i) the control horizon is equal to the prediction horizon; (ii) the state variables x are penalized rather than the output variables y; (iii) a penalty on the rate of change of the input is not included; and (iv) the steady–state target values are zero [Hen98]. However, from a computational point of view, an exact satisfaction of the terminal equality constraints requires an infinite number of iterations in the nonlinear case [ChJ98]. In order to avoid this drawback, they extend their work in [MM93] with a *terminal inequality constraints* such that the states are on the boundary of a *terminal region* at the end of a variable prediction horizon. They suggested a *dual–mode* receding horizon control scheme with a local linear state feedback controller inside the terminal region and a receding horizon controller outside the terminal region. Closed–loop control with this scheme is implemented by switching between the two controllers, depending on the states being inside or outside the terminal region.

Chen and Allgower [ChJ98] proposed a *quasi–infinite horizon NMPC* (QIH–NMPC) scheme that optimizes online an objective functional consisting of a *finite horizon cost* and a *terminal cost* ($E_s(x(t+P))$) subject to system dynamics, input constraints and a *terminal region constraints* $\Omega(x(t+P)) \geq 0$. Roughly speaking the terminal state penalty term approximates the infinite horizon cost. The terminal region is calculated around the origin that can be stabilized by a linear control law (*i.e.* $u = -Kx$). In this case, instead of requiring the final states to be at the origin, the final states will be in this region, Ω as shown in Figure 2.4.

In fact, the primary tool for NMPC stability analysis above is Lyapunov theory [KGE88]. For all algorithms in this theory, the Lyapunov function becomes the cost or objective function of the optimization.

The major differences in the existing algorithms are the length of the horizon and the consideration of an optimal or suboptimal solution;

1. Infinite horizon MPC
2. Finite horizon MPC without terminal constraints
3. Finite horizon with equality terminal constraints
4. Finite horizon with terminal state penalty and terminal region constraints (suboptimal MPC)

Other different schemes about guaranteeing stable NMPC can be found in the literature (see [MaR00, RO00]) for recent reviews. Recently, Imsland et al. [ImF03] reported that; if an approach of state feedback NMPC controller in combination with a state observer is used, in general little can be said about the stability of the closed–loop, since no universal separation principle for nonlinear systems exists.

2.8 Nonlinear State Estimation

Recent trends in MPC favor the closed–loop approach, where the measurements are incorporated into the prediction. This feature necessitates an estimator to recover the states from noisy measurements and a knowledge of a process model with uncertainty. The prediction is repeated at every time instant using the recovered states as initial conditions. Since closed–loop MPC requires the solution of the estimation and regulation problems online at each step, the computation time is limited between two successive measurements. For a linear system the estimation problem can be easily solved by the Kalman filter [Kal60], and the regulation problem can be solved very fast by linear optimizer. A number of closed–loop estimation techniques using Kalman filter for MPC has been proposed by many researchers [Ric90, LGM92, LY94]. However, for nonlinear systems, the computation cost becomes the biggest challenge for MPC applications. Here, both the regulation and estimation problems are generally nonlinear optimization problems, which are time–consuming to solve even for simple nonlinearities.

The most commonly used state estimation methods in NMPC are the *extended Kalman filter* (EKF) [LR94], and *moving horizon state estimation* (MHSE) [TRW04]. EKF is a nonlinear extension of the Kalman filter and it basically linearizes a nonlinear model repeatedly and applies a Kalman filter on the resulting time–varying system. The estimation component of NMPC using the EKF has negligible online computational load when compared to the control signal optimization since it actually deals with a linear problem. However, the EKF needs statistical knowledge (covariance matrices) of the noises acting on the states and on the output, which can be difficult to obtain in nonlinear cases.

A few more state estimation techniques are also available. There are some refined methods other than the celebrated Kalman filter or its extended nonlinear counter part. The refinements include; re-iteration [GeK82], higher–order filtering [May82], and statistical linearization [Lew86]. The more advanced techniques generally improve estimation accuracy but it happens at the expense of a further complication in implementation and much increased computational burden.

Norgaard et al, (2000) developed a new state estimator for nonlinear systems based on polynomial approximations rather than Taylor series approximations of the nonlinear transformations obtained with a particular multi-dimensional interpolation formula [NPR00]. The new estimators are named, a first-order and second-order Divided Difference filters (DD1) and (DD2) respectively. DD1 filter is based on first-order polynomial approximation and DD2 filter is based on second-order polynomial approximation. The authors claimed that the implementation of these types of filters is significantly simpler than estimators based on Taylor approximations used in the EKF estimator as no derivatives are required.

An interesting alternative is to use an optimization based method such as MHSE, which consists of minimizing an output criterion on a time horizon. It treats the estimation as a least square optimization problem in a moving window. However, the cost of extracting better performance is longer computation times for MHSE, which is comparable to the optimization cost of regulation. Both EKF and MHSE are function based state estimators in the sense that they pose optimization problems with the nonlinear (or linearized) functions of the models as constraints.

Recently, several probability density function based state estimators have been proposed. The Sequential Monte Carlo (SMC) [ChW04] or particle filters and Markov chain based Cell Filter (CF) [UC03] are typical examples. While these methods are computationally more expensive than EKF, they are shown to be less demanding and easier to tune than MHSE [ChW04].

2.9 Automatic Differentiation

The evaluation of derivatives of mathematical functions is a crucial ingredient in a variety of computational techniques in numerical simulations. Gradients, Jacobians, or higher–order derivatives are needed, for instance, in the solution of nonlinear systems of equations, differential and differential–algebraic equations, or optimization problems. Derivatives also play a key role in sensitivity analysis, model validation, inverse problems, and simulation problems [Ver00, ECF02], to name a few.

In this situation, numerical differentiation based on divided (finite) differencing (FD) is by far the most widely used approach. However, the main disadvantage of numerical differentiation is that any derivative value obtained from FD involves truncation error. It is sometimes difficult or even impossible to find a suitable step size [ECF02]. In addition, the run time requirements of finite difference approach are often unacceptably high, particularly for problems with large number (thousands) of independent variables.

Symbolic differentiation (SD) method is often used in computer algebra packages. Symbolic languages are increasely being used in analysis and implementation of control algorithms [Com94]. However, SD is known to be slow and to often produce large expressions which can become unmanageable [Com94]. It usually generates too long formulas for practical use, and, moreover, it is weak on differentiation of a function defined by a program containing conditional branches [Iri97].

Automatic differentiation (AD) [Gri89], which has developed rapidly during the last 20 years, is being recognized as the most promising among the differentiation algorithms. Like symbolic differentiation AD obtains exact derivatives without truncation error and preforms more efficiently than FD and SD in many cases [XZJ04].

AD is a simple and efficient technique for computing the derivative of a function represented by means of a program written in a higher level language such as FORTRAN, C, MATLAB, or others. It can be used as a pre-compiler that can take as input a FORTRAN/C subroutine that computes a function of several independent variables and write as output a program that computes not only the function, but also the gradient of the function with respect to the independent variables [ChS94, Ver00].

AD tool was introduced by Wengert in 1964, [Wen64], (see also [Wil64]) and further developed by Rall in 1980-1983 [Ral81]. The recent development of general purpose AD codes combined with the increasing interest in larger and more sophisticated control problems makes a consideration of the use of AD tool in control appropriate [GDJ96, Com94].

Several AD software tools are being developed by the applied mathematics and computing research community, all having both common and distinct features, regarding languages supported (FORTRAN 77, FORTRAN 90, C, C++). These tools include ADIFOR [BGH92], AUTO–DERIV [SPF00], ADIC [Bis96], ADOL-C [GDJ96], Odysee [FP98], and TAMAC [Gie97]. In the MATLAB environment, there is ADMIT-1 [CV97], ADMAT [CV98], MAD [AMR00], and ADiMAT [Veh04].

Rich and Hill (1992) presented a C language implementation of AD that can be called from MATLAB [RH92]. The C language GRAD [Rich89], automatically computes the value of multivariate function and its gradient vector of first partial derivatives at a given domain point. This function is called from MATLAB with a character string representing a function f of n variables and a point X in the n–space at which f and its first partial derivatives are to be evaluated.

A comparison between AD code in C language, (ADOL–C) [GDJ96], and symbolic approach on the problem of integration of nonlinear prescribed path control problem is given in [Com94]. The results were, for small problems either approach is suitable but for larger problems or moderate sizes involving a moderate number of derivative evaluations, the AD code is superior. Also, AD techniques are used to train large-scale artificial feedforward neural network by [ErG97]. The algorithm used AD for calculating derivatives and a conjugate gradients to approximatively solve a quadratic linear programming problem in each iteration of an optimization routine.

The application of AD to numerical integration algorithms for ODEs is discussed in [EB99]. An algorithm and software for sensitivity analysis of large-scale differential algebraic equation (DAE) systems are given in [ShP00].

For nonlinear systems there are some observer design methods which are based on differential geometric or differential algebraic concepts. The application to non-trivial systems of these methods and others are limited due to the burden of symbolic computations involved. To tackle this problem, a method for observer design using AD is developed by [RbR03].

A software COOPT package for optimal control of large-scale DAE systems is proposed in [SP01]. Gradient and Jacobian matrix that are the derivatives of the objective function and constraints, required in SQP method were computed via DAE sensitivity software DASPK3.0 [LP99]. The sensitivity equations to be solved are generated via AD techniques.

An extended automatic differentiation (XAD) algorithm was developed to generate the derivative and was coded into subroutines to replace the first derivative evaluation module of the SQP algorithm [XiW01]. The numerical results obtained using this algorithm showed that the derivative evaluation by expressions, produced by XAD, are far faster than that by finite difference approximation and plain AD. A significant enhancement in optimization efficiency was also obtained.

The advantages of using AD in the engineering applications was demonstrated in [BLR02]. They implemented the sensitivity analysis of electrostatic potential problem via AD. The derivatives produced by this approach shown to be more efficient than derivatives based on finite differences. They claimed that, their study demonstrated that the technology of AD is not only applicable to small codes but scales up to computer models consisting of hundred of thousands of lines of code. Also Dojouad et al. used AD tool for sensitivity analysis in the multiphase chemical mechanism application by [DAS02].

A methodology for implementing AD techniques into the TOUGH/ECO2 multiphase flow simulator was described in [KF03]. AD was used to provide accurate analytical derivatives for the Jacobian matrix, which is calculated to handle numerically the non-linear behavior inherent in non–isothermal, multiphase flow problems. They claimed that, the automatically generated AD code provides a faster derivative computation, with faster convergence in the subsequent linear solution steps, compared to the traditional finite difference method. Therefore a reduction in computational running time using AD was about 28%. Furthermore, the AD approach enhances the efficiency of the linear solution step, which resulted in a total computational time improvement of up to 60%.

AD has been applied to optimal control problems by [Ino03]. The sensitivity of the system was computed implicity by AD and the performance function deduced using a gradient method. Martinsen et al., (2004) investigated the application of four different SQP optimization algorithms to NMPC [MBF04]. The comparison results are collected through a CSTR case study to reached the suitable choice between these strategies. The Jacobian matrices associated with each discretization method of the CSTR model are calculated either analytically by AD tool, or approximated by FD method. The final results led to the following conclusion [MBF04];

- FD approximations of the full Jacobian in all the SQP strategies should be avoided.
- All solvers benefit from analytic derivatives and AD is a cheap way of achieving

this.

Recently, a module-oriented AD approach based on AD algorithms is presented in [XZJ04]. This approach can exploit the sparsity of a controller model by partitioning it into a series of sequential modules and choosing the best differentiation algorithm for each module accordingly. Moreover, external Jacobian evaluation codes for specific modules are claimed to be easily incorporated into this approach.

Mathematical exposition of the AD tool together with some examples of the use of the tool will be given in Chapter 3.

2.10 Unmeasured Disturbances and Their Modelling

Closed–loop performance of MPC algorithms is directly related to model accuracy. In practice, modelling error and unmeasured disturbances can lead to *steady–state offset* unless precautions are taken in the control design. Intuitively, one expects that measuring all state variables (usually impossible) would provide the best initial conditions. Sistue and Bequette (1991), [SB91] have shown that this is not the case for systems with parameter or model structure uncertainty. Poor dynamic response and steady–state offset can occur even if all state variables are measured and used as initial conditions with model/plant mismatch [Beq91b]. Also, offset could be caused by the finite prediction horizon [RST02]. Therefore, some form of feedback is required to remove this steady–state offset.

The first method involves modifying the control objective to include integration of the tracking error. This method, employed by the PID control algorithm, can also be used in the MPC framework [MuB02]. In this method, the integral action is incorporated into MPC algorithms by augmenting the process model to include a constant step disturbance. This disturbance is generated by comparing the measured and predicted process output at time k as;

$$\hat{d}(k) = y_m(k) - y(k) \tag{2.25}$$

where $y_m(k)$ and $y(k)$ represent the process measurement and model prediction at the current time, respectively. This disturbance is generally assumed to remain constant in the future and its effect on the controlled variables is removed by shifting the steady–state target for the controller. The simple formula of this disturbance can be given as;

$$y^*(k+i) = y(k+i) + \hat{d}(k), \quad \text{for} \quad i \in [1, P] \tag{2.26}$$

where y^* is the corrected prediction output. This is equivalent to assuming that a step disturbance enters the output of the process [MuR93]. For the case of a linear model and no active constraints, Rawlings et al. have shown that this form of feedback leads to offset–free control [RMM94]. However, when the process has a pure integrator, the constant output disturbance assumption will no longer lead to offset–free control [KC01]. For this case it is common to assume that an integrating disturbance with a constant ramp rate has entered at the output [QB97].

Most current NMPC implementations use the same method above . However, this approach suffers from several known limitations even for linear systems [KC01]. Also, this method is acceptable for stable plants. It cannot be used if the plant is unstable [Pan03].

The other approach to eliminating steady–state offset is that, the integral term is incorporated by augmenting the process model with tracking error states. However, for large–scale systems, this augmentation can significantly increase the computational cost of the dynamic optimization which grows in proportion to the cube of the state dimension [RWR98]

Muske and Rawlings [MuR93] show that a wider class of disturbance models can be implemented to the linear MPC using a standard Kalman filter [Kal60]. A disturbance model that adds step disturbances either to the state or the process output is used by Muske and Badgwell [MuB02]. This method is proven to remove offset when all the measured variables are controlled at a given setpoint. A general disturbance model for the case in which some of the measured variables are controlled at a given setpoint is proposed by Pannocchia and Rawlings [PR03].

The choice of the disturbance model may have a strong influence on the performance of MPC regulators. In fact, for nonlinear systems it seems reasonable to expect that similar benefits can be achieved by implementing an explicit disturbance model using EKF. Guidelines for disturbance model design are needed, however, to ensure that the resulting augmented system is detectable and that it allows for offset–free control. According to [Beq91b];

“the approach used for the solution of the ODEs is not nearly as important as the other issues involved, such as selection of the initial conditions for the model at each time step and the adjustment of the tuning parameters”.

In the context of linearizable systems, Sastry and Isidori [SI89] used parameter adaptation approach for improving the control of nonlinear processes. This approach was used by Iyer and Farell [IF95] to improve the performance of input–output control of one reactor. In a parallel study, Huberman and Lumer [HL90] introduced a simple adaptive control mechanism into nonlinear systems where a parameter in the system is updated using the difference between setpoint and output, and its derivatives. This approach is similar to the parameter adaptation of Sastry and Isidori [SI89], and was used in the internal model control of nonlinear systems by Narayanan et al., [NKR97], and Shukla et al., [ShD93]. Hu and Rangaiah [HR99] proposed a parameter adaptation law for internal model control of nonlinear processes and studied its performance theoretically as well as via simulation on typical processes.

Recently, Rangaiah et al. [RST02] proposed a simple adaptation technique for offset–free NMPC approach used to control an industrial four–stage evaporator system. They were considering that frequent disturbances are mainly in one of the plant unmeasured input variables (the flow rate of feed), and this quantity has been chosen as the model parameter to be updated irrespective of the actual disturbance in the real process. The adaptation gain is selected to eliminate the offset error (the difference between the plant outputs and setpoints) by a combination of physical insight and heuristics. They claimed that the main purpose of using the parameter adaptation is to improve the performance of NMPC in the presence of unmeasured disturbance.

In this thesis, two new offset removal techniques using a parameter adaptation techniques are developed for the NMPC context. These techniques included an integrated disturbances modelling via nonlinear integration, which can be then used as input or output disturbances to process model in order to correct its steady–state shifting from that of the real plant. Using nonlinear integration here is necessary to improve the transient response of the system during the offset error rejection time.

2.10.1 Nonlinear Integration of the Output Error

In the late 1950's, some researchers started to pay attention to the fact that a nonlinear regulator might improve the performance of a control system. A representative such result was the *nonlinear integrator* with 38^o phase shift presented by [Clg58]. Karybakas presented a nonlinear integrator with zero phase shift [Kar77]. In recent years, research work on nonlinear regulators emphasized by several types of nonlinear integrators named *intelligent integrators* have been proposed [GsA88, YZ90].

In a closed–loop control system, the role of the linear integrator is usually to eliminate the steady–state error. However, the phase shift of the *linear integrator* is a constant of -90^o and results in the reduction of phase margin of a closed–loop control system, and the possibility of instability in the system [SS98].

The Proportional Integral Derivative (PID) controller has been around for more than 50 years and it still widely used due to its structural simplicity and field–proven reliability. Linear and nonlinear PID controller can be found to work successfully in many industrial applications in the literature. A conventional PID (or specifically linear PID) controller can be described as:

$$u = K_P \left(e + \frac{1}{T_I} \int edt + T_D \dot{e} \right) \tag{2.27}$$

where, e, $\int edt$, and $\dot{e}$ represent the error, the integration of the error, and the derivative of the error, respectively, K_P is the proportional gain, T_I is the integral time constant, T_D is the derivative time constant, and u is the controller output.

A nonlinear PID (NPID) controller can be described as [JG01]:

$$u = K_{NP} \left[f(e, \alpha_P, \delta_P) + \frac{1}{T_{NI}} f \left(\int edt, \alpha_I, \delta_I) + T_{ND} f(\dot{e}, \alpha_D, \delta_D \right) \right] \tag{2.28}$$

where, e, $\int edt$, and e are the same as in the linear PID controller. The parameters K_{NP}, T_{NI}, T_{ND} hold the same meaning to K_P, T_I, and T_D in the linear PID controller, and $f(\cdot)$ is a nonlinear function. The idea of the NPID controller is to use a nonlinear combination of e, $\int e$ and $\dot{e}$ in place of the linear values in the linear PID controller. The $f(x, \alpha, \delta)$ function could be an exponential function with α and δ parameters (gain and exponent), tanh function or other types are used for the nonlinear mapping between x and y. Compared with the linear function $y = x$, the exponential nonlinear function $f(\cdot)$ for example gives high gain for small x and small gain for large x. This is equivalent to using a linear PID controller with varying parameters which are modified online based on the magnitude of the error. This is a form of *gain–scheduling* which gives high gain for small errors and small gain for large errors. The NPID controller uses an exponential function to implement this idea simply and systematically. The tuning of a NPID controller is similar to tuning a PID controller based on test results. Jiang [JG01] proposed a NPID control algorithm based on the idea above which is applied to a class of truck Anti–lock Brake System (ABS) problems. He showed that the proposed NPID for ABS archived better performance than the linear PID controller, and combines the advantages of robust control and easy tuning.

A link between PID controller and generalized predictive control was performed by Tan et al [THL00]. They proposed a PID controller with time–scheduled gains to unstable systems with deadtime. The controller gains are designed based on a GPC control approach. Also, a class of nonlinear PID predictive controllers are derived using nonlinear GPC approach by Chen et al. [CBG]. It is pointed out that this composite controller is equivalent to a nonlinear controller with integral action. The proposed controller is illustrated by an example nonlinear mechanical system.

Nonlinear PID control has a long history [Kri80, Rug87, Ser98, ANK01] and has found two broad classes of applications:

1. Nonlinear systems, where NPID control is used to accommodate the nonlinearity, often to achieve consistent response across a range of conditions [Rug87].
2. Linear systems, where NPID control is used to achieve performance not achievable by linear compensation [ANK01].

It has been proven that for linear systems NPID control can provide [Ser98, JG01]:

- Increased damping
- Reduced rise time for step or rapid inputs
- Improve tracking accuracy
- Friction compensation
- Increase the controller robustness

The requirements for high performance control with changes in operating conditions or environmental parameters and safety consideration are often beyond the capabilities of simple PID (fixed–gain) controllers [Ser98]. A nonlinear integrator with positive phase shift was proposed by Sheng [SS98]. In comparison with the linear PID controller, the proposed nonlinear one has a better performance in the control system.

Seraji [Ser98] introduced a NPID controllers and provided a formal treatment of their stability. Three simple nonlinear gains are used for the proposed controllers: the *sigmoidal–tanh* function, the *hyperbolic* function, and the *piecewise linear* function. The systems to be controlled are assumed to be modelled or approximated by second–order transfer functions. The proposed nonlinear PI controller were implemented as a force controller on a robotic arm and experimental results are presented. The results demonstrated the superior performance of the nonlinear PI controller relative to a fixed–gain PI controller.

The nonlinear integration method has been used in the PID controller context only and no application of it in MPC context was found. In this thesis a novel method is proposed to ensure free–offset NMPC algorithm using a nonlinear integration action to eliminate the steady–state error due to unmeasured disturbances or modelling error with a satisfactory results.

2.11 Summary

The presented literature and background theory can be summarized as follows:

- Model predictive control works in real time in a feed forward mode with a feed back of the process output after every calculation cycle is completed and a control step is affected. It uses an internal model of the process to be controlled and can take account of any changes in the objective function or process constraints
- The process and constraints can be linear resulting in LMPC or nonlinear resulting in NMPC. Linear MPC results in a convex objective function with guaranteed optimal solution. Nonlinear MPC on the other hand does not in general produce convex objective functions and other methods need to be pursued in order to achieve convergence in the optimizer.
- The computation burden for NMPC is prohibitive and leaves plenty of scope for improvement in this area especially for highly nonlinear processes or those with large number of input and input variables or long prediction horizons. A major part of this computation burden is in function and partial derivative evaluations.
- The automatic differentiation tool is a new and efficient tool for evaluating functions and its partial derivatives. It has already been used successfully for the solution of differential equations and optimization, but not in the NMPC field. The use of AD can therefore bring a great reduction in computation time. Further, as the AD works on the computer code that defines a problem, it can also reduce modelling time in addition to computing time.

Chapter 3

Automatic Differentiation

3.1 Introduction

There are many ways to obtain the derivatives of a mathematical function. Straight forward hand calculations is the first to come in mind. This course is usually taken in problems of small size. If the function is not simple, the number of variables is large, or the input and output range of values is large, hand calculation becomes nearly impossible and prone to errors that are difficult to debug. In general, there are three important methods for finding the derivatives. These are, the numerical differentiation method, the symbolic method, and the recently developed method of automatic differentiation. The three methods are introduced in the next sections with special emphasis in the AD tool.

3.2 Numerical Differentiation

The most common alternative to hand coding is the numerical approximation of derivatives by *Finite Difference* (or FD) formula. A simple formula is constructed from the expansion of $f(x)$ in Taylor series truncated after the first order term:

$$f(x) = f(x_k) + \Delta x \left.\frac{\partial f}{\partial x}\right|_{x=x_k} + \mathcal{O}(\Delta x^2) \tag{3.1}$$

where $\Delta x = x - x_k$ and it is some very small positive number. Evaluated at $x = x_k - \Delta x$, then a good approximation of the derivative is computed easily as;

$$\left.\frac{\partial f}{\partial x}\right|_{x=x_k} = \frac{f(x_k) - f(x_k - \Delta x)}{\Delta x} + \mathcal{O}(\Delta x^2) \tag{3.2}$$

The order of the approximation is controlled by the term at which the series is truncated that is the last term in (3.2). Note that only function evaluations are needed to calculate the derivative. Thus, the coding of the algorithm is very simple and existing codes can be used.

More accurate derivative can be computed using *Centered Finite Differences*(CFD) method as:

$$\left.\frac{\partial f}{\partial x}\right|_{x=x_k} = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + \mathcal{O}(\Delta x^3) \tag{3.3}$$

which usually gives better approximation, but cost an additional function evaluation. The main disadvantage of FD and CFD approaches lies with the tradeoff between truncation error and roundoff error. As the truncated Taylor series expansion is only valid in the neighborhood of the expansion point x_k, small values of Δx tend to reduce the truncation error. Ideally, the exact derivative is the limit of these formula above, when Δx tends to zero. However, very small values of Δx increase the roundoff error. Finding the best Δx requires numerous executions of the program, and even then the computed derivatives are just approximations. Other disadvantages of this approach are the instability of higher order differentiation formula and the computational cost of the techniques, approximately $n+1$ times the computational effort associated with evaluation of the function itself.

3.3 Symbolic Differentiation

Symbolic differentiation (or SD) is a computer aided analog to analytical or hand differentiation employing a graph theoretical approach. The formula representation of a function is transformed into a formula representation for its derivative, that is either interpreted or further transformed into a program in a common programming language. In principle, evaluation of these formula gives exact values of the derivatives of the function. Symbolic differentiation only incurs roundoff error resulting from the individual floating–point operations.

Symbolic differentiation, usually performed in computer algebra packages like Maple [ChG88] and Mathematica, is unable to deal with branches, loops and subroutines intrinsic in computer codes. For every binary operator (except + or -), the derivative expression is likely to double in size, leading to a combinatorial explosion effect. Due to this effect, the resulting derivative code is difficult to manipulate and to be used for practical software applications. The computational cost of SD is almost impossible to predict. It generally grows enormously with function complexity. Instead, Automatic

Differentiation is bounded in terms of the number of independent and dependent variables.

3.4 Automatic Differentiation

Automatic Differentiation (or AD), which is developed for the automatic computation of derivatives, is a new approach to obtain analytical derivatives of programs (possible containing conditional statement, loops etc.) [TB98]). AD is the numerical computation of exact values of the derivative of a function at a given argument value. It just like FD, requires only the original program f. But instead of executing f on different sets of inputs, it builds a new, augmented, program f', that computes the analytical derivatives along with the original program. This new program is called the *differentiated* program. Precisely, each time the original program holds some value v, the differentiated program holds an additional value dv, the differential of v. Moreover, each time the original program performs some operations, the differentiated program performs additional operations dealing with the differential values. So, it decomposes the model into a series of elementary functions ($\times, /, sin(\cdot)$, etc.), applies the simple rules of differentiation (product rule, quotient rule, etc.) to evaluate the partial derivatives of the elementary functions, and then accumulates them with the chain rule to obtain the derivatives of the program. The resulted derivative values are obtained without generating a formula for the derivatives, thus avoiding the unnecessary overhead of symbolic differentiation and the truncation error inherent in FD formulas [Gri89].

To explain further, consider the function $f(x, y)$ represented below:

$$f(x, y) = 0.5xy^3 - \exp(\sin(x + y)) \tag{3.4}$$

The partial derivatives of this functions are easily obtained and are equal to:

$$\frac{\partial f}{\partial x} = 0.5y^3 - \cos(x + y)\exp(sin(x + y)) \tag{3.5}$$

$$\frac{\partial f}{\partial y} = 1.5xy^2 - \cos(x + y)\exp(sin(x + y)) \tag{3.6}$$

Using only binary operations, this function would be represented as shown in Table 3.1. Differentiation each line of the code, one would get the code to generate the derivative without the formula of the derivative, shown as (3.5) and (3.6).

In order to calculate the partial derivative in respect to x the vector $[dx, dy]^T$ is set to $[1, 0]^T$, meaning that $\partial y/\partial x = 0$ and $\partial x/\partial x = 1$. Analogously, to calculate the partial

Table 3.1: Evaluation of f and its derivative

Evaluation of $f(x,y)$	Evaluation of $f'(x,y)$
$v_1 = y$	$dv_1 = dy$
$v_2 = v_1^3$	$dv_2 = 3v_1^2 dv_1$
$v_3 = xv_2$	$dv_3 = xdv_2 + dxv_2$
$v_4 = 0.5v_3$	$dv_4 = 0.5dv_3$
$v_5 = x + y$	$dv_5 = dx + dy$
$v_6 = \sin(v_5)$	$dv_6 = \cos(v_5)dv_5$
$v_7 = \exp(v_6)$	$dv_7 = \exp(v_6)dv_6$
$f(x,y) = v_4 - v_7$	$df = dv_4 - dv_7$

Table 3.2: Evaluation of the partial derivative of f

Evaluation of $\partial f/\partial x$	Evaluation of $\partial f/\partial y$
$dx = 1, \quad dy = 0$	$dx = 0, \quad dy = 1$
$dv_1 = dy = 0$	$dv_1 = dy = 1$
$dv_2 = 3(v_1)^2 dv_1 = 0$	$dv_2 = 3(v_1)^2 dv_1 = 3y^2$
$dv_3 = dxv_2 = v_2 = y^3$	$dv_3 = xdv_2 + 0 = 3xy^2$
$dv_4 = 0.5dv_2 = 0.5y^3$	$dv_4 = 0.5dv_3 = 0.5(3xy^2) = 1.5xy^2$
$dv_5 = dx + dy = 1 + 0 = 1$	$dv_5 = dx + dy = 0 + 1 = 1$
$dv_6 = cos(v_5)dv_5 = cos(x+y)$	$dv_6 = cos(v_5)(dv_5) = cos(x+y)$
$dv_7 = \exp(\sin(x+y))(\cos(x+y))$	$dv_7 = \exp(\sin(x+y))(\cos(x+y))$
$df = 0.5y^3 - \cos(x+y)\exp(\sin(x+y))$	$df = 1.5xy^2 - \cos(x+y)\exp(sin(x+y))$

derivative in respect to y, the vector $[dx, dy]^T$ is set to $[0,1]^T$. In Table 3.2 it is shown that the evaluation of the formulas on Table 3.1 would lead to the same expressions of (3.5) and (3.6).

Although great advances have been made in symbolic differentiation of formulas, AD generally requires less memory and CPU time, and also applies to functions defined by computer programs or subroutines for which no formula may be available [CVB00].

3.5 Different AD Modes

Automatic differentiation has two basic modes of operation, the *forward mode* and the *reverse mode*.

In the forward mode, the derivatives are propagated through the computation using the chain rule. This is the most classical approach, the differentiation machinery behaves as a human who would augment the code by additional instructions computing the derivatives (and re–using the shared expressions assigned to temporary variables). Consider a function, $y = g(f(x))$ consisting of two operations: $v = f(x)$ and $y = g(v)$. In forward mode, by applying the chain rule, $\dot{y} = dy/dx$ can be evaluated in the sequence: $\dot{x} = 1$, $\dot{v} = f'(x)\dot{x}$ and $\dot{y} = g'(v)\dot{v}$. In forward mode, a function and its derivatives can be evaluated in parallel. Note that the same method is used in Table 3.1. This mode is easy to understand and implement, and requires computational effort proportional to $n \times m$, where n is the number of independent variables and m the dimension of the function component.

In the reverse mode, the intermediate derivatives are computed in the reverse order, from the final results down to the independent variables. The reverse mode evaluation is based on the definition of adjoint, $\overline{v} = dy/dv$. After evaluating the sequence, $v = f(x)$ and $y = g(v)$ with all intermediate results recorded, adjoints are evaluated in a reverse sequence: $\overline{y} = 1$, $\overline{v} = \overline{y}g'(v)$ and finally, $dy/dx = \overline{x} = \overline{v}f'(x)$.

The reverse mode requires saving the entire computation trace, since the propagation is done backwards through the computation, and hence, the partial derivatives need to be stored for derivative computation. Hence the reverse mode can be prohibitive due to memory requirements [Ver00]. However, the reverse mode is better for computing multi–dimensional gradients of a function, because the computational effort for it is proportional with m, the length of the code list, and independent of n, the number of independent variables. In fact, when the number of m is much less than n such as the objective function of an optimization problem, evaluation in reverse mode is vastly more efficient than in forward mode. This can result in significant saving in computational time [Ral81].

There are aspects to be considered other than merely the computational cost when discussing AD modes. Reverse mode implementation is quite more sophisticated and may employ complex structures of indirect addressing. That may prevent vectorization of the final code [CVB00]. Hence, available AD codes employ a combination of both strategies in order to balance complexity and computational cost.

In the thesis the MAD/ MATLAB Toolbox is used for derivative calculation, where only first-order derivative is required such as in the first NMPC algorithm where the sensitivity equations are solved using a first–order approximation. Also it is used in the local linearization step required in the EKF stage, and in the NMPC of the ALSTOM gasifier case study. While ADOL–C tool is used in algorithms that needed higher order derivatives such as the sensitivity equations calculations using Taylor series expansion and in the training algorithm of the continuous–time recurrent neural network and also in the predictive controller for this type of model.

3.6 Implementation of AD using MAD/MATLAB Toolbox

MAD (forward mode AD) can easily compute first derivatives of functions defined via expressions built–up of the arithmetic operations and intrinsic function of MATLAB using forward mode AD. It evaluates a function $f(x)$ at $x = x_0$ and in the course of this also evaluates the directional derivatives $f'(x_0)v$. This is done by first initializing x as an **fmad** object using (in MATLAB syntax) $x = \text{fmad}(x_0, v)$ to specify the point x of evaluation and the directional derivative v. Then evaluation of $y = f(x)$ propagates **fmad** object which, through overloading of the elementary operations and functions of MATLAB ensures that directional derivatives and values of all successive quantities and in particular y are calculated. By specifying $v = V$, a matrix forming several directional derivatives may be propagated simultaneously. In particular, if $V = I$ the identity matrix, then the Jacobian $f'(x)$ at $x = x_0$ is evaluated. If the result of a calculation is a N–dimensional array, then directional derivatives are N–dimensional arrays, and an $N+1$ dimensional array may be propagated to handle multiple directional derivatives [AMR00].

In this thesis MAD is used as AD tool to calculate the first–order partial derivatives required to solve the online optimization problem of the first proposed NMPC approach, and also in the model linearization step for different cases. For calculating high order derivatives (as the case of Taylor series expansion method required in another parts of the thesis), the C language package ADOL–C is used.

3.7 Implementation of AD using ADOL-C Software

The **C++** package ADOL–C (**A**utomatic **D**ifferentiation by **O**ver**L**oading in **C++**) proposed by Griewank et al. [GDJ96], facilitates the evaluation of first and higher derivatives of vector functions that are defined by computer programs written in **C** or **C++**. The resulting derivative evaluation routines may be called from C/C++, Fortran, or any other language that can be linked with C. In the second part of the thesis (where high order derivatives will required) ADOL–C is linked with MATLAB via *mex* warp for derivatives evaluation.

ADOL–C facilitates the simultaneous evaluation of arbitrarily high directional derivatives and the gradients of these Taylor coefficients with respect to all independent variables. Relative to the cost of evaluating the underlying function, the cost for evaluating any such scalar–vector pair grows as the square of the degree of the derivative but is still completely independent of n, the number of vector functions component, and m, the number of independent variables.

For the reverse propagation of derivatives, the whole execution trace of the original evaluation program must be recorded, unless it is recalculated in piece as advocated in [Gri92]. In ADOL–C, this potentially very large data set is written first into a buffer array and later into a file if the buffer is full or if the user wishes a permanent record of the execution trace. In either case, it refer to the recorded data as the *tape*. The user may generate several tapes in several named arrays or files. During subsequent derivative evaluations, tapes are always accessed strictly sequentially, so that they can be paged in and out to disk without significant runtime penalties. If written into a file, the tapes are self–contained and can be used by other Fortran, C or C++ programs [Gri00].

3.8 Taylor Series Function Expansion using AD

Consider a d–time continuously differentiable function, $f \; : \; \mathbb{R}^n \rightarrow \mathbb{R}^m$. Let $x(t) \in \mathbb{R}^n$ denote any vector polynomial in the scalar variable $t \in \mathbb{R}$ which can be given by the truncated Taylor series:

$$x(t) = \sum_{j=0}^{d} x_{[j]} t^j \tag{3.7}$$

with Taylor coefficient vectors:

$$x_{[j]} = \frac{1}{j!} \left. \frac{\partial^j x(t)}{\partial t^j} \right|_{t=0} \tag{3.8}$$

are simply the scaled derivatives of $x(t)$ at the parameter origin $t = 0$. The first two vectors $x_{[1]}, x_{[2]} \in \mathbb{R}^n$ can be visualized as tangent and curvature at the base point $x_{[0]}$, respectively. Then, $z(t) = f(x(t)) \in \mathbb{R}^m$ can be expressed by a Taylor expansion [GDJ96]:

$$z(t) = \sum_{j=0}^{d} z_{[j]} t^j + \mathcal{O}(t^{d+1}) \tag{3.9}$$

where $z_{[j]} \in \mathbb{R}^m$ coefficient is given by

$$z_{[j]} = \frac{1}{j!} \frac{\partial^j z(t)}{\partial t^j}\bigg|_{t=0} \tag{3.10}$$

From the chain rule, $z_{[j]}$ is uniquely and smoothly determined by the coefficient vectors, $x_{[i]}$ with $i \leq j$, *i.e.*

$$z_{[0]} = f(x_{[0]}), \quad z_{[1]} = f'(x_{[0]})x_{[1]} \tag{3.11}$$

and

$$z_{[2]} = f'(x_{[0]})x_{[2]} + \frac{1}{2}f''(x_{[0]})x_{[1]}x_{[1]} \tag{3.12}$$

It is well known that the number of terms that occur in these “symbolic” expressions for the $z_{[j]}$ (in terms of the first j derivative tensors of f and the “input” coefficients $x_{[i]}$ with $i \leq j$) grows very rapidly with j. Fortunately, this exponential growth does not occur in AD, where the many terms are somehow implicity combined so that storage and operating count grow only quadratically in the bound d on j.

Provided f is analytic, this property is inherited by the functions;

$$z_{[j]} \equiv z_{[j]}(x_{[0]}, x_{[1]}, \cdots, x_{[j]}) \in \mathbb{R}^m \tag{3.13}$$

Inherently, functions $z_{[j]}$ are also d-time continuously differentiable and their derivatives satisfy the identity [Chr92]:

$$\frac{\partial z_{[j]}}{\partial x_{[i]}} = \frac{\partial z_{[j-i]}}{\partial x_{[0]}} := A_{[j-i]} \equiv A_{[j-i]}(x_{[0]}, x_{[1]}, \cdots, x_{[j-i]}) \tag{3.14}$$

where, $A_{[j]} \in \mathbb{R}^{n\times n}, j = 0, \cdots, d$ are also the Taylor coefficients of the Jacobian path, *i.e.*;

$$\frac{\partial f}{\partial x} = A_{[0]} + A_{[1]}t + \cdots + A_{[d]}t^d + \mathcal{O}(t^{d+1}) \tag{3.15}$$

The AD software package ADOL–C provide an efficient way to calculate these coefficients vectors, $z_{[j]}$ and matrices $A_{[i]}$ [Gri00]. For example, using the forward mode of AD all Taylor coefficient vectors for a given degree, d can be calculated simultaneously, whilst the matrices, $A_{[i]}$ can be obtained using the reverse mode of AD. The run time and memory requirement associated with these calculations grow only as d^2.

3.8.1 Derivatives for Ordinary Differential Equations

When the above approach is applied to an *autonomous* ordinary differential equation, *i.e.*;

$$\dot{x} = f(x(t)) \tag{3.16}$$

since

$$x_{[k+1]} = \frac{z_{[k]}}{k+1} \tag{3.17}$$

all Taylor coefficients of $x(t)$ up to any order can be iteratively obtained from $x_{[0]} = x(0)$ using (3.13) by the forward mode of AD (**forward** routine in ADOL–C package) [Gri95]. Moreover, the sensitivity of Taylor coefficients against the initial value $x_{[0]}$ can also be efficiently obtained by matrix accumulation from (3.14):

$$B_{[k]} := \frac{dx_{[k]}}{dx_{[0]}} = \frac{1}{k}\frac{dz_{[k-1]}}{dx_{[0]}} = \frac{1}{k}\sum_{j=0}^{k-1}\frac{dz_{[k-1]}}{dx_{[j]}}\frac{dx_{[j]}}{dx_{[0]}} = \frac{1}{k}\sum_{j=0}^{k-1}A_{[k-j-1]}B_{[j]} \tag{3.18}$$

where, $B_{[k]} \in \mathbb{R}^{n\times n}, k = 0,\cdots,d$ are the Taylor coefficients of the solution to the sensitivity equations, $\dot{B} = f'(x)B$, $B_{[0]} = B(0) = I$.

The above algorithm was extended to solving dynamic sensitivity of the following *non–autonomous* state–space systems where an input signal is present as follows:

$$\begin{aligned}\dot{x}(t) &= f(x(t), u(t)), \quad x(0) = x_0 \\ y(t) &= g(x(t), u(t)), \quad 0 \le t \le T_s\end{aligned} \tag{3.19}$$

where, $u(t) \in \mathbb{R}^{n_u}$ is the control input, $y(t) \in \mathbb{R}^{n_y}$ is the output, and T_s is the sampling time. System (3.19) can be converted to an autonomous system by augmenting it with $\dot{u} = 0$ [RV04], so that the results described in above can be directly used. However, the augmented system has n_u extra differential equations, hence the algorithm is not–efficient particularly when n_u is large. An efficient approach was proposed by [Cao05] to tackle this problem which is used in this thesis as will be shown in Chapter 7.

Chapter 4

NMPC using Dynamic Sensitivity Approximation with AD

.

4.1 Introduction

Although NMPC might be the best choice for a nonlinear plant, it is still not widely used. This is mainly due to the computational burden associated with solving a set of nonlinear differential equations and a nonlinear dynamic optimization problem. In this chapter, a new NMPC algorithm based on nonlinear least square (NLSQ) optimization is proposed. In the new algorithm, the residual Jacobian matrix is efficiently calculated from the model sensitivity function without extra integrations. The sensitivity functions are accurately and efficiently obtained from the state trajectory using AD tool. These three features make the new algorithm computationally efficient.[1]

4.2 Principle of NMPC

Model Predictive control solves an online finite horizon open–loop optimal control problem to select a set of future control moves (control horizon) based on the desired output trajectory and constraints involving states and controls. Figure 4.1 shows the general principle.

[1] Original paper has been published in Ref. [CAS03]

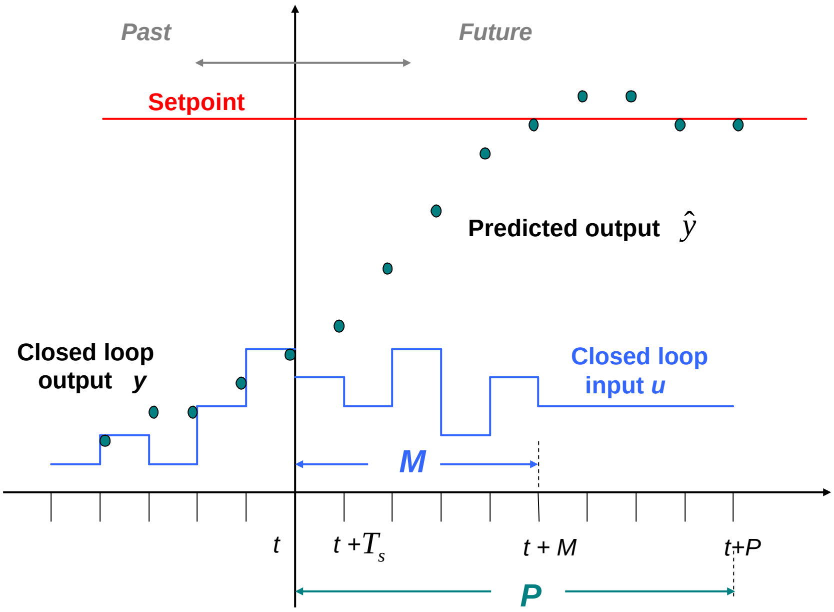

Figure 4.1: The principle of the Model Predictive Control

Based on measurements obtained at time t, the controller predicts the future dynamic behavior of the system over a prediction horizon P and determines (over a control horizon $M \leq P$) the input such that a predetermined open-loop performance objective functional is optimized. If there were no disturbances and no model/plant mismatch, and if the optimization problem could be solved for infinite horizons, then one could apply the input function found at time $t = 0$ to the system for all times $t \geq 0$. However, this is not possible in general. Due to disturbances and model/plant mismatch, the true system behavior is different from the predicted behavior. In order to incorporate some feedback mechanism, the open–loop manipulated input function obtained will be implemented only until the next measurement becomes available. The time difference between two measurements can vary if necessary, however often it is assumed to be fixed, i.e the measurement will take place every T_s sampling time units. Using the new measurement at time $t + T_s$ the whole procedure prediction and optimization is repeated to find a new input function with the control and prediction horizons moving forward (for this reason, MPC is also referred to as moving horizon control) [FnI03].

If the output/input relation of the plant is linear, the problem is LMPC and the optimization problem is convex and relatively easy to solve using existing tools. If the output/input relation is nonlinear it becomes NMPC and a nonlinear optimization problem is at hand. A new efficient NMPC algorithm based on nonlinear least square optimizer is developed in the next sections.

4.3 Nonlinear Model Predictive Control with AD

4.3.1 Nonlinear Least Square Problem

The nonlinear model predictive control considered is to solve the following nonlinear optimization problem at each sampling time

$$\min_{\substack{\underline{u} \leq u \leq \overline{u}, \\ k=0,...,M-1}} \quad V = \frac{1}{2}\sum_{i=1}^{P} ||e^y_{k+i}||^2_{Q_i} + \frac{1}{2}\sum_{j=1}^{M} ||\Delta u_{k+j}||^2_{S_j} \tag{4.1}$$

subject to

$$\begin{aligned} \dot{x} &= f(x,u), \quad t \in [t_0, t_P] & (4.2)\\ y &= g(x,u) & \\ x(t_0) &= x_0, \quad x_k := x(t_0 + kT_s) & \\ e^y_k &:= y_k - r_k, \quad k \in [1,P] & (4.3)\\ \Delta u_k &= u_{k+1} - u_k \quad k \in [1,M] & (4.4)\\ u_k &= u(t_k) = u(t), \quad t \in [t_k, t_{k+1}] & (4.5)\\ u_k &= u_{M-1}, \quad k \in [M, P-1] & (4.6) \end{aligned}$$

f and g are a nonlinear functions, T_s is the sampling period, the control horizon $[t_0, t_M]$ is divided into M intervals, and the prediction horizon $[t_0, t_P]$ is divided into P intervals, $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, $y \in \mathbb{R}^{n_y}$, are state, input, and output variables in n_x, n_u, and n_y dimensions respectively, r_k the reference vector at t_k. $\overline{u}$ and $\underline{u}$ are constant vectors determining the input upper and lower constraints, and in the form of element-by-element inequalities. Note that, the term $||e||_Q \equiv \sqrt{e^TQe}$ denotes the weighted 2–norm of a vector e, the weighting matrices $Q_k \in \mathbb{R}^{n_y \times n_y}$ and $S_k \in \mathbb{R}^{n_u \times n_u}$ are chosen to be positive definite and both have a Cholesky factorization *i.e.*:

$$Q = Q_1^T Q_1, \; S = S_1^T S_1 \tag{4.7}$$

where, Q_1 and S_1 are an upper triangular matrices.

Taking

$$e_k^Q := Q_{1,k} e_k^y \tag{4.8}$$

$$\Delta u_k^S := S_{1,k} \Delta u_k \tag{4.9}$$

$$E = \begin{bmatrix} e_1^{QT} & \cdots & e_P^{QT}, \ \Delta u_1^{ST} \cdots \Delta u_M^{ST} \end{bmatrix}^T \in \mathbb{R}^{n_y P + n_u M} \tag{4.10}$$

and

$$U = \begin{bmatrix} u_0^T & \cdots & u_{M-1}^T \end{bmatrix}^T \tag{4.11}$$

$$\overline{U} = \begin{bmatrix} \overline{u}^T & \cdots & \overline{u}^T \end{bmatrix}^T \in \mathbb{R}^{n_u M} \tag{4.12}$$

$$\underline{U} = \begin{bmatrix} \underline{u}^T & \cdots & \underline{u}^T \end{bmatrix}^T \in \mathbb{R}^{n_u M} \tag{4.13}$$

Then the optimization problem of (4.1) can be restated as a standard NLSQ problem in the form:

$$\min_{\underline{U} \leq U \leq \overline{U}} \quad V(U) = \frac{1}{2} E(U)^T E(U) \tag{4.14}$$

The gradient matrix $G(U)$ and Hessian matrix, $H(U)$ of (4.14) have the special structure:

$$G(U) \;=\; J^T(U)E(U) \tag{4.15}$$

$$H(U) \;=\; J^T(U)J(U) + W(U) \tag{4.16}$$

where, $J(U) \in \mathbb{R}^{(n_y P + n_u M) \times n_u M}$ is the Jacobian matrix of residuals vector, $W(U)$ is defined as:

$$W(U) = \sum_{i=1}^{n_y P + n_u M} E_i(U) H_i(U) \tag{4.17}$$

where E_i, is the ith element of E, and H_i, is the Hessian matrix of E_i. The matrix $W(U)$ has the property that when U is approaching optimal solution and residual $\|E(U)\|$ tends to zero then also $W(U)$ tends to zero. This allows some efficient algorithms to be applied to solve the problem [Mar63].

4.3.2 Sensitivity Calculation using First–order Approximation and AD

To solve the nonlinear least square problem, the Jacobian matrix for E is needed as derived in this section. The $(n_y P + n_u M) \times n_u M$ Jacobian matrix is defined as:

$$J(U) = \left[\frac{\partial E_i}{\partial U_j} \right], i \in [1, n_y P + n_u M], \ j \in [1, n_u M] \tag{4.18}$$

which can be re-written as;

$$J(U) = \begin{bmatrix} J^y(U) & J^u(U) \end{bmatrix} \tag{4.19}$$

where the Jacobian matrix J^y can be partitioned into $P \times M$ blocks as;

$$J^y(U) = [J^y_{i,j}], \quad i \in [1, P], \quad j \in [1, M] \tag{4.20}$$

and each block is an $n_y \times n_u$ matrix defined as;

$$J^y_{i,j} = \frac{\partial e^Q_i}{\partial u_{j-1}} = Q_{1,i}\frac{\partial e^y_i}{\partial u_{j-1}} = Q_{1,i}\frac{\partial y_i}{\partial u_{j-1}} \tag{4.21}$$

and the Jacobian matrix J^u can be portioned into $M \times M$ blocks as

$$J^u(U) = [J^u_{i,j}], \quad i \in [1, M], \quad j \in [1, M] \tag{4.22}$$

where each block ia an $n_u \times n_u$ matrix defined as

$$J^u_{i,j} = \frac{\partial \Delta u^S_i}{\partial u_{j-1}} = S_{1,i}\frac{\partial \Delta u_i}{\partial u_{j-1}} \tag{4.23}$$

The value of the Jacobian matrix $J^y(U) \in \mathbb{R}^{n_y P \times n_u M}$ in equation (4.20) is calculated using the fact that future input cannot have an effect on a past state, $J^y_{i,j} = 0$ for $i < j$, i.e. the Jacobian matrix is low block-triangular.

Taking partial derivative of both sides with respect to u_{j-1} of equation (4.2) gives:

$$\frac{\partial}{\partial u_{j-1}}\left(\frac{dx_i}{dt}\right) = \frac{d}{dt}\left(\frac{\partial x_i}{\partial u_{j-1}}\right) = f_x\frac{\partial x_i}{\partial u_{j-1}} + f_u\frac{\partial u_i}{\partial u_{j-1}} \tag{4.24}$$

$$\frac{\partial y_i}{\partial u_{j-1}} = g_x\frac{\partial x_i}{\partial u_{j-1}} + g_u\frac{\partial u_i}{\partial u_{j-1}} \tag{4.25}$$

where, $f_x := \partial f/\partial x$, $f_u := \partial f/\partial u$, $g_x := \partial g/\partial x$, and $g_u := \partial g/\partial u$ respectively. Note that, the order of differentiation in the l.h.s. term of equation (4.24) is reversed because u is a continuous function during the sampling time.

Equation (4.24) is a linear time-varying system with initial condition; $\partial x(t_0)/\partial u_{j-1} = 0$. For $j < M$, the input, $\partial u(t)/\partial u_{j-1}$ is an impulse function:

$$\frac{\partial u}{\partial u_{j-1}} = \begin{cases} I, & t \in [t_{j-1}, t_j] \\ 0, & \text{otherwise} \end{cases} \tag{4.26}$$

for $j = M$, the input is a step function,

$$\frac{\partial u}{\partial u_{j-1}} = \begin{cases} I, & t \geq t_{M-1} \\ 0, & \text{otherwise} \end{cases} \tag{4.27}$$

Generally, the nonlinear time-varying equation (4.24) has no analytical solution although it can be represented in state-transition matrix form [ChC98]. Numerically, equation (4.24) can be solved together with the ODEs (4.2) using a differential equation solver. The total number of differential equations to be solved in (4.24) is $n_x \times n_u \times M$. Equation (4.24) can be simplified by approximated the sensitivity functions, f_x, f_u, g_x, and g_u with piecewise constant. In this case, within any one sampling period, analytical solutions can be obtained as follows

Let

$$\begin{aligned}
A_i &= e^{f_x(t_i)\tau}, \quad B_i = \int_0^T e^{f_x(t_i)\tau} d\tau f_u(t_i), && (4.28)\\
C_i &= g_x(t_i), \quad D_i = g_u(t_i), && (4.29)\\
z_{i,j} &= \partial x_i / \partial u_{j-1}, && (4.30)\\
v_{i,j} &= \partial u_i / \partial u_{j-1}, && (4.31)\\
w_{i,j} &= \partial y_i / \partial u_{j-1} && (4.32)
\end{aligned}$$

then equation (4.24) can be discretized as,

$$\begin{aligned}
z_{i,j} &= A_{i-1} z_{i-1,j} + B_{i-1} v_{i-1,j} && (4.33)\\
w_{i-1,j} &= C_{i-1} z_{i-1,j} + D_{i-1} v_{i-1,j}
\end{aligned}$$

For $j < M$, $v_{i-1,j} = 0$ if $i > j$, and $v_{i-1,j} = I$, $z_{i-1,j} = 0$ if $i = j$ since the future inputs can not effect the past states. Thus,

$$z_{i,j} = \begin{cases} A_{i-1} z_{i-1,j} & \text{for } i > j \\ B_{j-1} & \text{for } i = j \end{cases} \qquad (4.34)$$

Recursively, the following solution can be derived:

$$\begin{aligned}
z_{i,j} &= A_{i-1} A_{i-2} \dots A_j B_{j-1} && (4.35)\\
w_{i-1,j} &= C_{i-1} A_{i-2} A_{i-3} \dots A_{j-1} B_{j-2} + D_j && (4.36)
\end{aligned}$$

For $j = M$, $v_{i-1,j} = I$ if $i \geq j$. Denote,

$$\begin{aligned}
\Phi(i,k) &= A_i A_{i-1} \cdots A_k && (4.37)\\
\Phi(i-1,i) &= I
\end{aligned}$$

and

$$\begin{aligned} \Psi(i,k) &= C_{i-1}A_{i-1}A_{i-2}\cdots A_{k-1} \\ \Psi(i-1,i) &= I \end{aligned} \tag{4.38}$$

then the solution for this case is

$$z_{i,M} = \sum_{k=M}^{i} \Phi(i-1,k)B_{k-1} \tag{4.39}$$

$$w_{i-1,M} = \sum_{k=M}^{i} \left(\Psi(i-1,k)B_{k-1} + D_{k-1}\right) \tag{4.40}$$

$$\tag{4.41}$$

The value of the Jacobian matrix $J^u(U) \in \mathbb{R}^{n_uM \times n_uM}$ in equation (4.23) can be easily calculated by taking the partial derivatives of the algebraic equation (4.4). Since for taking $j < M$;

$$\frac{\partial \Delta u_i}{\partial u_{j-1}} = \begin{cases} I & \text{if } i = j-1 \\ -I & \text{if } i = j \\ 0 \ \ \text{else} & \end{cases} \tag{4.42}$$

and $\partial \Delta u_i / \partial u_{j-1} = 0$ for $j \geq M$ then

$$J^u(U) = S^{1/2} \begin{bmatrix} I & 0 & \cdots & 0 & 0 \\ -I & I & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & -I & I \end{bmatrix} \tag{4.43}$$

where I is an identity matrix with n_u dimension.

The forward mode of AD tool (using MAD software, the AD toolbox in MATLAB) is used to calculate the sensitivity variables f_x and f_u which are then used in the calculations of the time–varying matrices A, B, C and D respectively.

The above procedure is summarized as follows:

1. Give the initial values u_0, x_0, $u_1, \cdots, u_M$.
2. Integrate system (4.2) to get $x_1, x_2, \cdots, x_P$ and $y_1, y_2, \cdots, y_P$.
3. Use the forward mode of AD tool to calculate $f_x(t_k)$ and $f_u(t_k)$ for $k = 1, 2, \cdots, P$.
4. Calculate A_k, B_k, C_k, and D_k matrices using (4.28) and (4.29).
5. Use the above matrices to calculate the Jacobian matrix J.

4.4 Nonlinear Model Predictive Control Algorithm

In this section, the outline of the proposed new NMPC algorithm is stated. This description is general and is equally valid whether the model is first–principle or derived from a black box and whether a first–order or high–order approximation of the sensitivity functions is involved. The NMPC algorithm can be described as follows;

1. Collect current process information, such as measurements (measurable states, measurable disturbance), and reference trajectory.
2. Use current measurements to estimate unknown information such as, unmeasured state variables, unmeasured disturbance, etc.
3. Apply nonlinear least square optimization solver to get a promising guess of next control horizon.
4. The solver calls the objective function to calculate the cost corresponding to the control horizon provided.
5. The cost subroutine calls an ordinary differential equation solver to predict the state trajectory based on the control horizon provided.
6. Based on the prediction trajectory obtained, the cost subroutine applies an AD tool to get the sensitivity function of the trajectory.
7. The residual Jacobian matrix is calculated according to the sensitivity function obtained.
8. The residual vector and residual Jacobian matrix is returned to the optimization solver. If the terminal conditions are not satisfied, the solver updates a new control horizon and the procedural is repeated from step 3. Otherwise, the first point of the control horizon is implemented to the process control system and the procedural is repeated from step 1.

Remark 1. Most existing Newton–type dynamic optimization algorithms involve an inverse integration for co–state or adjoint variables in order to get gradient information. In the proposed algorithm, the gradient information is obtained without such integration. After calculating the state trajectory, only algebraic calculations are involved in the procedural to get residual Jacobian matrix. Therefore, efficiency is greatly improved in the proposed algorithm.

Remark 2. For small dimension systems, the sensitivity function might be derived analytically. However, for large dimension systems, particularly for a practical application, it is not a trivial task to calculate the analytic derivatives of a functions. Sometime, it is even not possible. Here the AD tool is used to provide help. The proposed algorithm provides a link point where an AD tool can be involved to solve the online dynamic optimization problem. In this work, the AD tool in MATLB, MAD Toolbox is used.

Remark 3. For a feasible problem, the convergence of the algorithm is mainly determined by the nonlinear least square solver used. In this work, the solver provided in MATLAB Optimization Toolbox is applied. It is based on the Levenberg Marquardt method [Mar63, Den77], which uses a scalar to control both search direction and magnitude. When the scalar is zero, the search direction is identical to that of the Gauss–Newton method, while as the scalar tends to infinity, search direction tends toward a steepest descent direction. Therefore, in most cases, the value of cost function is non–increasing and the algorithm is convergent.

Remark 4. The nonlinear least square method is a special case of general nonlinear optimization problems. The main advantage of using the nonlinear least square formulation is its efficiency due to the special structure of its gradient (4.15) and Hessian (4.16) whilst the disadvantage is that it cannot directly handle hard constraints on output variables. The hard constraints should be converted to soft ones using Lagrange multipliers in order to use this method.

4.5 Stability Analysis

To ensure the closed–loop stability for the above NMPC algorithm, one of many current solutions available in the literature can be used. For examples using long enough prediction horizon, or using QIH–NMPC scheme [ChJ98] by adding a terminal state penalty $||x(t+P)||_H^T$ to the objective function (after modify the objective function (4.1) to the form of equation (2.24) and adding a terminal region constraint $\Omega(x(t+P)) \geq 0$. However, adding an inequality constraint has the additional complications of determining the terminal region in most cases off–line, and adding an elliptical constraints that must be approximated in the optimization method [Ten02]. Tenny (2002) [Ten02] reported that;

"*our experience, dictates that, provided the terminal penalty is large enough or the prediction horizon is long enough, the terminal constraints is not required at all*"

Therefore, to simplify the NMPC problem the prediction horizon is kept long and the terminal penalty and the terminal constraints are not enforced. Good tuning of the NMPC parameters *i.e.* M, Q, and S was also found in the present work to be enough to ensure closed–loop stability for all the cases studies under widely different operating conditions.

4.6 Case Study: Evaporator Process

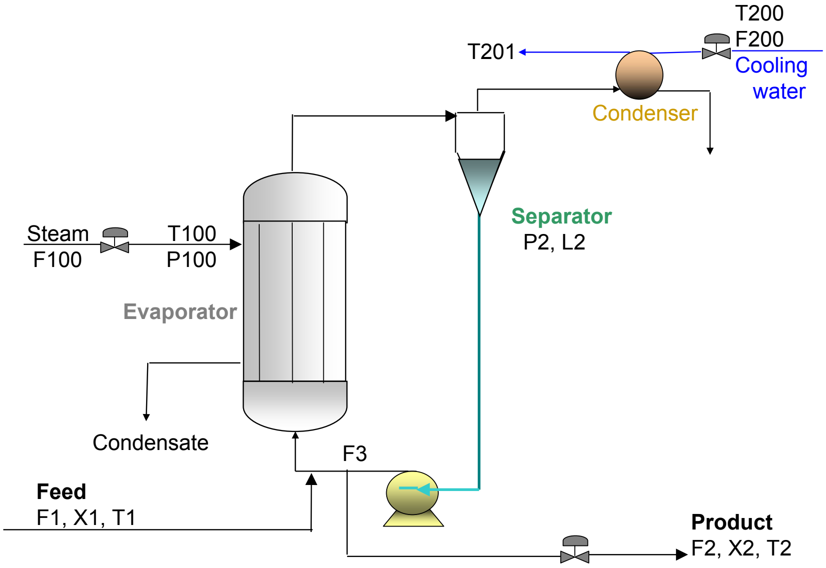

Figure 4.2: Evaporation system

Evaporation is an operation to remove a liquid from a solution, suspension, or emulsion by boiling off some of the liquid. It is thus a thermal separation, or thermal concentration process. Evaporation process can be defined as one that starts with a liquid product and ends up with a more concentrated, but still liquid and still pumpable concentrates as the main product from the process.

The concentration of dilute liquors by evaporating solvent from the feed stream is an important industrial process used in such industries as sugar mills, alumina production and paper manufacture, to name a few. An often used evaporator, known as a *forced circulation evaporator*, which descried in Newell and Lee [NL89] and shown in Figure 4.2. In this process, a feed stream enters the evaporator at concentration X_1 and temperature T_1, with flow rate F_1. It is mixed with recirculating liquor, which is pumped through the evaporator at a flow rate F_3. The evaporator itself is a heat exchanger, which is heated by steam flowing at rate F_{100} with entry temperature T_{100} and pressure P_{100}. The mixture of feed and recirculating liquor boil inside the heat exchanger, and the resulting mixture of vapour and liquid enters a separator where the liquid level is L_2 . The operating pressure inside the evaporator is P_2. Most of the liquid from the separator becomes the recirculating liquor; a small proportion of it is drawn off as the product, with concentration X_2, at flow rate F_2 and temperature T_2. The vapour from the separator flows to a condenser at flow rate F_4 and temperature T_3, where it is condensed by cooling water flowing at rate F_{200}, with inlet temperature T_{200} and outlet temperature T_{201}. Variable names, descriptions, standard steady state values, and engineering units are listed in Table A.1 (see Appendix A). Note that, C_P is the heat capacity of the liquid and is assumed as a constant of 0.07 $[kW/K(kg/m)]$.

The process is open–loop marginally stable due to the integrating characteristics of the liquid level in the evaporator separator. The first–principle model of the process is given in Appendix A.

4.7 Evaporator Control Specifications

Effective control of the evaporator system using traditional PID controllers was not very successful especially for large setpoint changes [NL89]. Predictive control was also considered by a number of workers. Linear model predictive control has also failed to fully control this process for both the regulating and tracking problems [Mac02]. A nonlinear MPC strategy based on successive linearization solution to control this process under a large setpoint change condition was proposed by Maciejowski [Mac02]. A good performance was observed after re-linearizing the nonlinear process model after every few steps. However, disturbances have not been considered there. In this chapter, the NMPC algorithms described in section 4.4 is applied to control the process for setpoint tracking and disturbance rejection tests described as follows.

The control objective of the case study is;

1. Track setpoint ramp changes of X_2 from 25% to 15% and P_2 from 50.5 kPa to 70 kPa.

2. Track setpoint changes as above when disturbances, F_1, X_1, T_1 and T_{200} are varied within ±20% of their nominal values.

3. Offset free performance when unknown non–zero mean disturbance applied at F_1 (a step change -20% of its nominal value at $t = 10$ minutes), or modelling error applied to the internal model.

The control system is configured with three manipulated variables F_2, P_{100} and F_{200} and three measured variables, L_2, X_2, and P_2 . All manipulated variables are subject to a first-order lag with time constant of 0.5 min and saturation constraints, $0 \leq F_2 \leq 4$, $0 \leq P_{100} \leq 400$, $0 \leq F_{200} \leq 400$.

All disturbances are simulated as a step signal passing through a first-order lag. This makes the process as close as possible to the real process. The amplitudes of step changes are randomly produced within the ±20% range of the nominal values. The changing intervals and time constants of the first–order delays are different for different disturbance variables shown in Table 4.1. It is assumed that F_1 is a measurable disturbance in this chapter. The other three disturbances are considered not measured at all tests. The nonlinear dynamic model of the process (first–principle) is used as the plant model. In this chapter, the same model is used also as the internal model for NMPC whilst the actuator lags and disturbances lags are ignored in the prediction in the controller. The ignored lags can be considered as plant/model mismatches for the NMPC.

Table 4.1: Disturbance model parameters

Disturbance	Interval [min]	Time constant [min]
F_1	5	0.2
X_1	2	0.2
T_1	1	1
T_{200}	1	1

4.7.1 Evaporator Control using NMPC

The proposed NMPC algorithm is used to control the plant. The NMPC parameters are chosen after online tuning with the following starting values; sampling period, $T_s = 1$ minutes, prediction horizon, $P = 5$ time steps (5 minutes), control horizon, $M = 2$ time steps (2 minutes), actuators lower limit, $\underline{U} = [0 \;\; 0 \;\; 0]^T$, actuators upper limit, $\overline{U} = [4 \;\; 400 \;\; 400]^T$, outputs weighting matrix, $Q = \text{diag}([1000 \;\; 100 \;\; 100])$, $S = \text{diag}([1 \;\; 0.5 \;\; 0.5])$. Good and stable response is obtained in the plant outputs in all the required tests using the above tuned parameters.

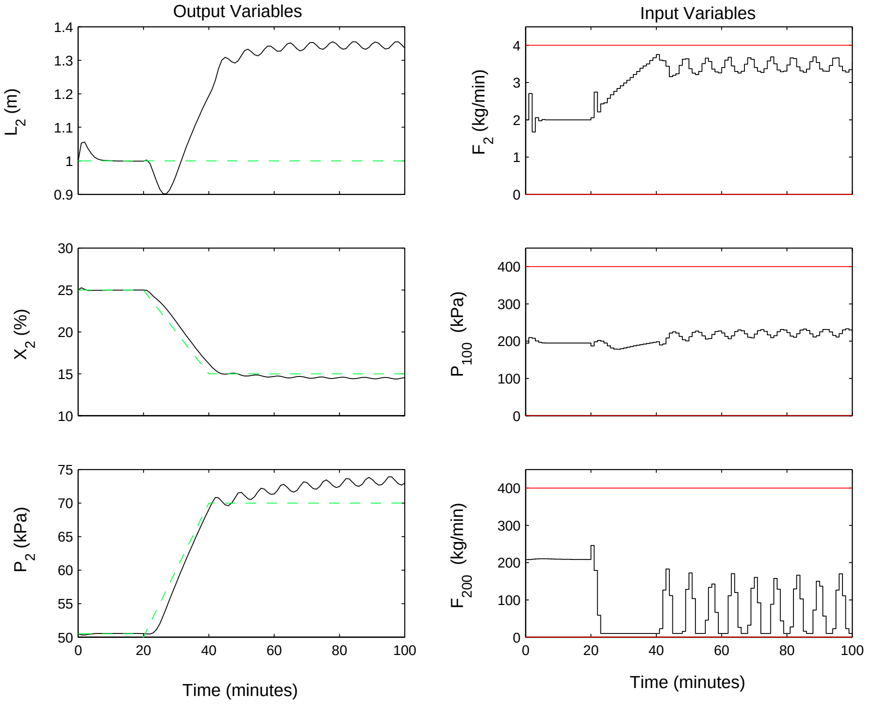

Figure 4.3: Evaporator performance at setpoints ramp changes using LMPC [Mac02].

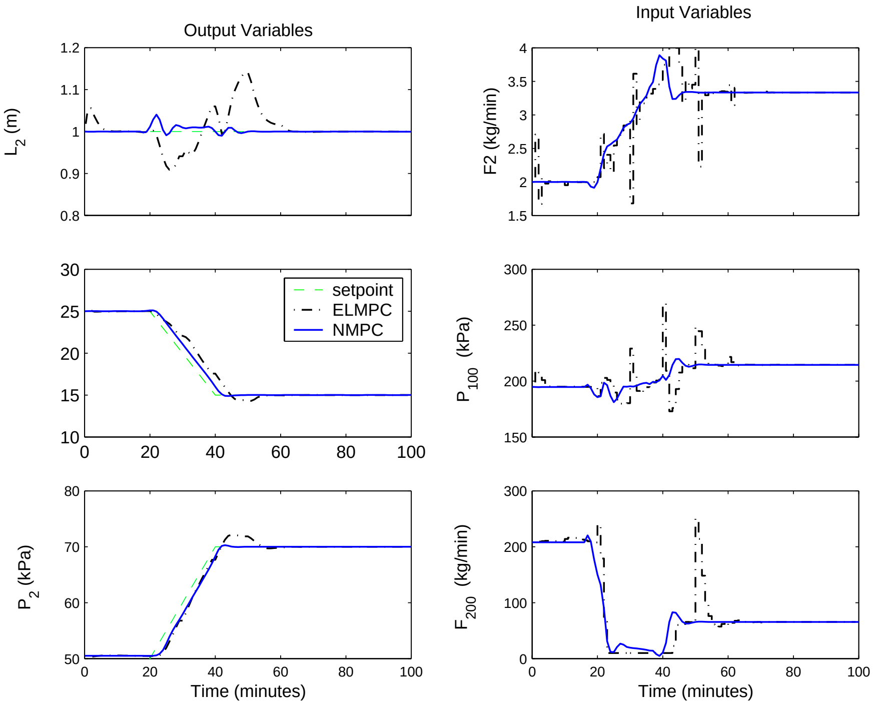

Figure 4.4: Evaporator performance at setpoints ramp change using; extended linearization MPC (ELMPC) of [Mac02], (dot–dashed lines); the NMPC of this work (solid lines).

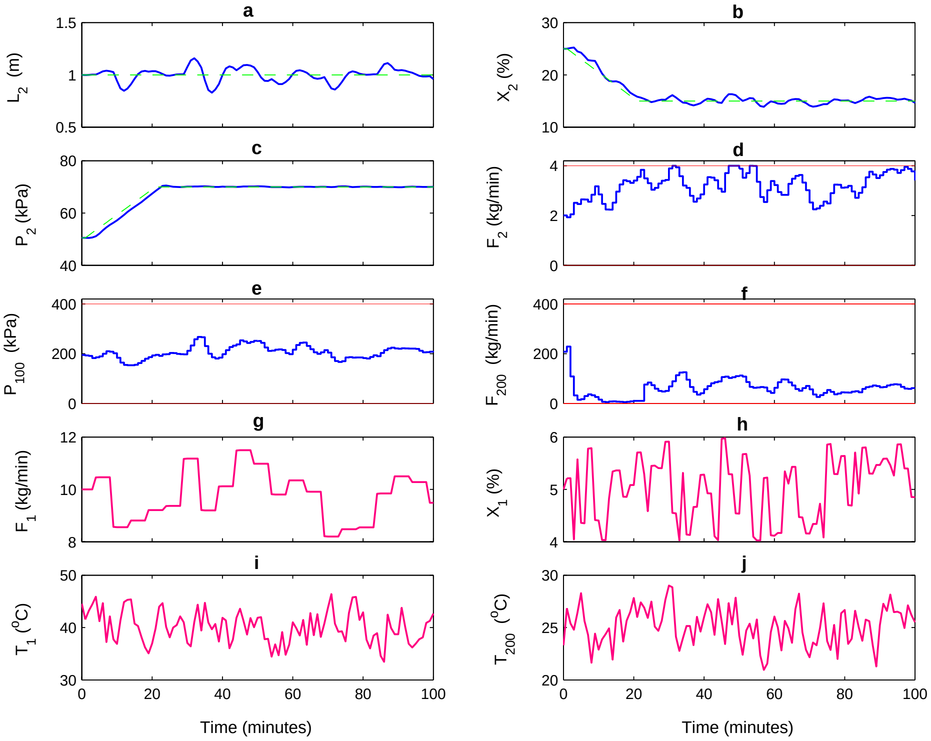

Figure 4.5: Evaporator performance using the present NMPC at setpoint changes plus random disturbances test. (a)–(c) Measured outputs (solid lines) with setpoints (dashed lines). (d)–(f) Manipulated variables. (g)–(j) Disturbances.

Simulation is performed with the above configuration where the NMPC controller is represented via S-function in MATLAB. The behavior of the process in a setpoint ramp change using the evaporator response when a LMPC [Mac02] is used for controlling the plant under setpoint change test is shown in Figure 4.3. The figure shows problems in L_2 as a result of setpoint ramp change in X_2 and P_2. A serious offset results and persists after the plant has settled to the new setpoints. The main reason for this result is the presence of a mismatch between the nonlinear plant and the linearized predictive model. Re-linearizing the internal model every 10 minutes during the prediction horizon (extended linearization MPC (ELMPC)) solved the setpoint tracking problem (see controller ELMPC performance in Figure 4.4) [Mac02]. When the proposed NMPC algorithm is used to control the process under the same condition, noticeable improvement in the system performance is observed as in Figure 4.4. In addition to the good performance associated with using the new NMPC algorithm, the online time required to solve the open–loop optimization problem was shorter by about 50% compared with the predictive controller based on re–linearized the process model every 10 minutes given in [Mac02].

Figure 4.5 shows the system performance using the proposed NMPC approach during

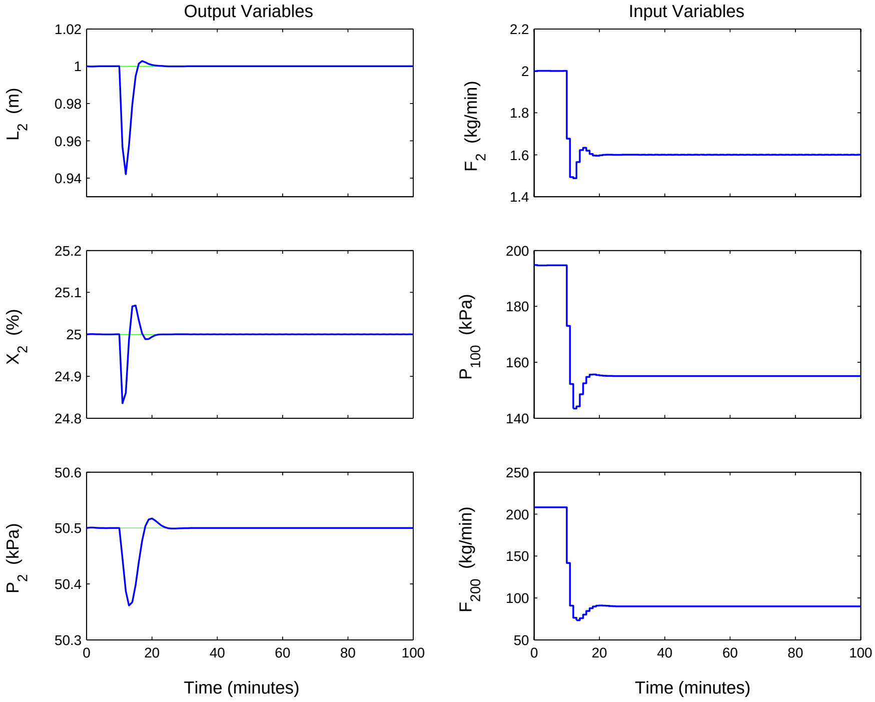

Figure 4.6: Simulation results under measured step change disturbance in F_1 at $t = 10$ minutes.

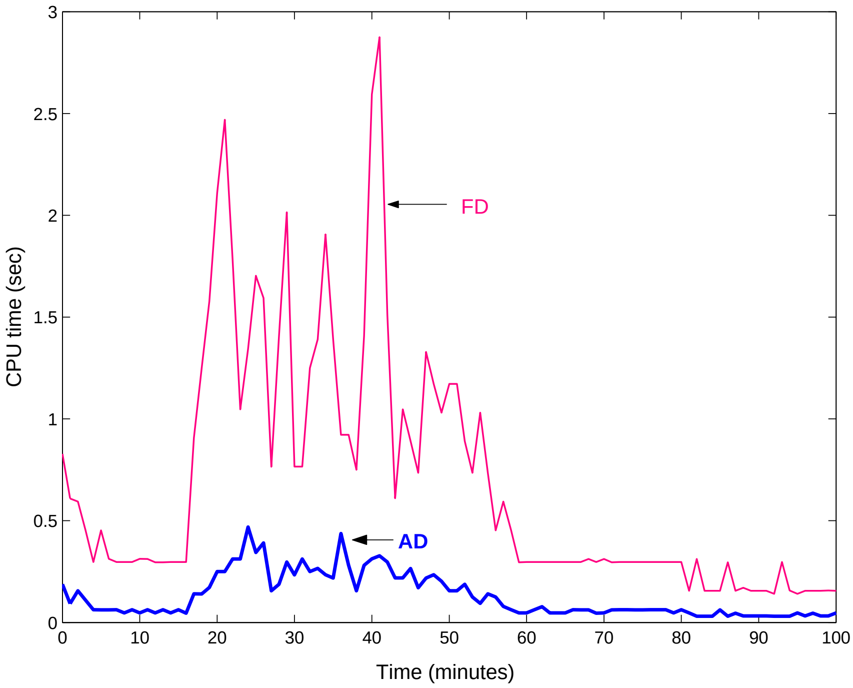

Figure 4.7: The CPU time used by MATLAB process to solve the online optimization problem during ramp setpoint tracking problem of the evaporator process.

a setpoint tracking test when disturbances F_1, $X1$, T_1, and T_{200} are varying within ±20% of their nominal values. The figure shows that the measured outputs follow the setpoints quite well (a)–(c) in spite of the existence of severe disturbances (g)–(j). This is achieved without violating the input constraints (d)–(f).

In a disturbance rejection test, a non–zero mean disturbance in F_1 is assumed. It is represented by a step change in -20% from the F_1 nominal value of F_1 applying to the process at $t = 10$ minutes. Note that the disturbance variable F_1 is assumed measurable in this case. Figure 4.6 shows the system performance using the proposed NMPC.

It can be concluded from the above results that, the NMPC controller is effective and achieves all the performance requirements.

To demonstrate the new NMPC algorithm efficiency, the CPU time in seconds that has been used by the MATLAB process to solve each optimization problem during ramp setpoint tracking problem is plotted in Figure 4.7. Note that, all computations perform on a Windows XP PC with an Inetl Pentium-4 processor running at 3.0 GHz. Large reduction in the computation time is observed using the NMPC approach with AD compared with the case when the numerical differentiation (FD) is used to calculate the gradient of the objective function.

4.8 Summary

A new NMPC algorithm is proposed in this chapter. Based on a nonlinear least square optimization problem, an efficient algorithm to calculate the residual Jacobian matrix is derived. With the new approach, the gradient information can be obtained without further integration of the sensitivity differential equations. The new algorithm also provides a link point where recently developed automatic differentiation techniques can be applied to get derivatives (sensitivity functions) accurately and efficiently. The evaporator case study shows that satisfactory performance is obtained with the controller using the new NMPC algorithm. Large time saving is obtained using the proposed algorithm compared with finite difference method. No terminal penalty is used in this work and a good tuning of T_s, P, M, Q, and S was found adequate to ensure the close-loop stability for the case study in different operation conditions.

Chapter 5

State Estimation Using EKF with AD

5.1 Introduction

The proposed NMPC algorithm is extended in this chapter to include an additional stage often required for many MPC applications. This is the state estimation stage. This part is necessary to estimate any unmeasurable states to be used as the initial values to solve the online optimization problem. The well known Extended Kalman Filter has been chosen for this task. In order to increase the calculation speed and accuracy in the state estimation stage, the model linearization step required for the EKF, is done automatically using AD tool. The two–CSTR process is used to test the new NMPC algorithm with the state estimator.

5.2 State Estimation in NMPC

In many practical problems, the states of the system are not directly accessible and must be estimated. The quality of state estimates has important bearings on the overall performance of a model predictive controller, especially of one based on a nonlinear model [ML99]. The implementation of the NMPC techniques discussed in the previous section requires knowledge of the current state of the nonlinear system in order to compute the solution to the open–loop optimal control problem formulated at each control interval. Feedback in this controller comes from the update of the

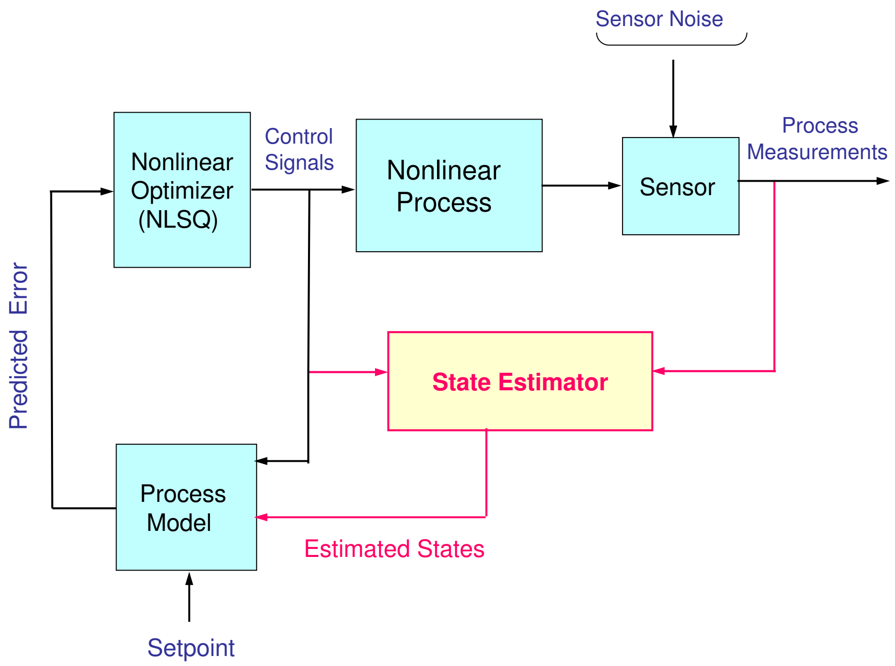

Figure 5.1: The state feedback control

current state. Since, the full state of the nonlinear system is not directly measurable in most applications, some method of reconstructing the current state of the system from the measured outputs must be employed. The importance of state estimation is clarified in Figure 5.1. Good estimation to the process states, which is used as initial values to solve the NLP each time step, will lead usually to a better and more stable performance. The two basic goals of state estimation are; (i) get estimates of unmeasured states from output measurements, (ii) Reduce the influence of measurement noise on state estimates. The details of the EKF is given below.

5.3 Principle of Extended Kalman Filter

The Kalman filter [Kal60] is an optimal state estimator applied to a dynamic system that involves random noise and includes a limited amount of noisy real–time measurements. Although it was originally derived for linear systems, the Kalman filter can also be extended for application to nonlinear systems via specific online Taylor expansions of the originally nonlinear system. The Kalman filter so obtained is called *extended Kalman filter*.

A straight forward approximation to optimal nonlinear state estimation is to *linearize* the nonlinear model about a given operating point and apply optimal linear state estimation to the linearized system. The EKF computes a state estimate at each sampling time by the use of Kalman filtering on a linear time–varying system of the nonlinear system. Application of the EKF is therefore contingent upon the assumption that the required derivatives exist and can be obtained with a reasonable effort. The Taylor linearization provides an insufficiently accurate representation in many cases, and significant bias, or even convergence problems, are commonly encountered due to the overly crude approximation.

The detailed theory and equations of the EKF are given in the literature [GA93, StP93, WB04, Nor01]. The KALMTOOL Toolbox in MATLAB [Nor01] is used in this work. The toolbox was developed based on the following formula;

Re–write system (4.2) as follows:

$$\begin{aligned} \dot{x}(t) &= f(t,x,u,\upsilon) \\ y(t) &= g(t,x,\omega) \end{aligned} \tag{5.1}$$

where, $\upsilon(t)$ and $\omega(t)$ are the actuators and measurement noise. The discrete version of the continuous–time state–space model (5.1) is given as:

$$\begin{aligned} x(k+1) &= F(x(k),u(k),\upsilon(k)) \\ y(k) &= g(x(k),\omega(k)) \end{aligned} \tag{5.2}$$

where $F(x(k),u(k),\upsilon(k)) := \int_{kT_s}^{(k+1)T_s} f(t,x,u,\upsilon)dt$ denotes the terminal state vector obtained by integrating the ODEs (5.1) for one sample interval T_s with the initial condition of $x(k)$ and constant inputs of $u(k)$.

Assume that the actuators noise υ, output noise ω, and the system's initial state $x(0)$ satisfy the following conditions:

1. υ and ω are zero-mean Gaussian white random processes. For any $k \geq 0$ and $l \geq 0$,

$$\mathbf{E}\{\upsilon(k)\} = 0 \tag{5.3}$$

$$\mathbf{E}\{\omega(k)\} = 0 \tag{5.4}$$

$$\mathbf{E}\{\upsilon(k)\upsilon(l)^T\} = \begin{cases} R^\upsilon & \text{if } k=l \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

$$\mathbf{E}\{\omega(k)\omega(l)^T\} = \begin{cases} R^\omega & \text{if } k=l \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

where R^{υ} and R^{ω} represent the noise covariance matrices which are known positive semi–define matrices.

2. $x(0)$ is a Gaussian random vector with known mean and auto–covariance matrix

$$\begin{aligned} \hat{x_0} &= \mathbf{E}\{x(0)\} && (5.7) \\ \Sigma_0 &= \mathbf{E}\left\{[x(0) - \hat{x_0}]^T \cdot [x(0) - \hat{x_0}]\right\} && (5.8) \end{aligned}$$

3. $x(0)$ is uncorrelated to υ and ω at any k.

EKF solves the problem by making the following linear approximation with respect to $x(k|k-1)$ where $x(k|l)$ denote the optimal estimates (*i.e.* minimum variance estimates) for based on the measurements up to time l.

$$\begin{aligned} x(k+1|k) &\approx F(x(k|k-1), u(k), 0) + \alpha(k)(x(k) - x(k|k-1)) + \beta(k)\upsilon(k) \\ y(k) &\approx g(x(k|k-1), 0) + G(k)(x(k) - x(k|k-1)) + \Gamma(k)\omega(k)) && (5.9) \end{aligned}$$

where

$$\begin{aligned} \alpha(k) &= e^{\widetilde{\alpha}(k)\tau} && (5.10) \\ \beta(k) &= \int_0^{T_s} e^{\widetilde{\alpha}(k)\tau} d\tau \cdot \widetilde{\beta}(k) && (5.11) \\ G(k) &= \left.\frac{\partial g}{\partial x}\right|_{\hat{x}(k|k-1), u(k), 0} && (5.12) \\ \Gamma(k) &= \left.\frac{\partial g}{\partial \omega}\right|_{\hat{x}(k|k-1), u(k), 0} && (5.13) \end{aligned}$$

and

$$\begin{aligned} \widetilde{\alpha}(k) &= \left.\frac{\partial f}{\partial x}\right|_{\hat{x}(k|k-1), u(k), 0} && (5.14) \\ \widetilde{\beta}(k) &= \left.\frac{\partial f}{\partial \upsilon}\right|_{\hat{x}(k|k-1), u(k), 0} && (5.15) \end{aligned}$$

At each time instant k, given $y(k)$ (and its available value at the previous time instants, *i.e.* $y(k-1)$), it is the goal of the extended Kalman filter to deliver state estimate $\hat{x}(k+1|k)$ so as to minimize the covariance of the estimation error;

$$\mathbf{E}\left\{[x(k+1) - \hat{x}(k+1|k)]^T \cdot [x(k+1) - \hat{x}(k+1|k)]\right\} \qquad (5.16)$$

where $\hat{x}(k+1|k)$ denotes the mathematical expectation of $x(k+1)$ conditional on measurements available up to the kth time instant (actually $\hat{x}(k+1|k)$ is one–step prediction of $x(k+1)$).

The recursive equations of the EKF are as follows:

$$\hat{x}(k+1|k) = \underbrace{F\left(\hat{x}(k|k-1), u(k), 0\right)}_{\text{model}} + \underbrace{L(k)\left[y(k) - g\left(\hat{x}(k|k-1), 0\right)\right]}_{\text{correction}} \tag{5.17}$$

where $L(k)$ is the Kalman filter steady–state gain given as;

$$L(k) = \Sigma(k|k-1)G^T(k)\left[G(k)\Sigma(k|k-1)G^T(k) + \Gamma(k)R^{\omega}(k)\Gamma(k)^T\right]^{-1} \tag{5.18}$$

The estimation error covariance $\Sigma(k+1|k)$ is computed from $\Sigma(k|k-1)$ as follows;

$$\Sigma(k+1|k) = \alpha(k)\Sigma(k|k-1)\alpha^T(k) + \beta(k)R^v\beta(k)^T \tag{5.19}$$

with

$$\hat{x}(0|-1) := \hat{x}_0 \tag{5.20}$$

$$\Sigma(0|-1) := \Sigma_0 \tag{5.21}$$

In order to obtain the value of the Jacobian linearization matrices $\alpha(k)$, $\beta(k)$, $G(k)$, and $\Gamma(k)$ with high accuracy and less computational efforts, the forward mode of the AD toolbox (MAD) in MATLAB has been used in this work successfully.

5.4 Case Study: Two CSTR Processes

A chemical system common to many chemical processing plants, known as a Continuous Stirred Tank Reactor (CSTR), was utilized as a suitable test for many control methods [Cao95, CB96]. It suffices to know that the CSTR constituted by a jacketed, perfectly mixed reactor, where an exothermic, first order and irreversible chemical transformation from reactionant **A** to product **B** takes place.

A process comprising of two CSTRs (CSTR1 and CSTR2) in series with an intermediate mixer introducing a second feed (see Figure 5.2) [Cao95] is investigated. Several possible input–output configurations have been considered to determine the best control scheme, the lowest integral square error (ISE) cost over alternative configurations of manipulated variables corresponds to the best option.

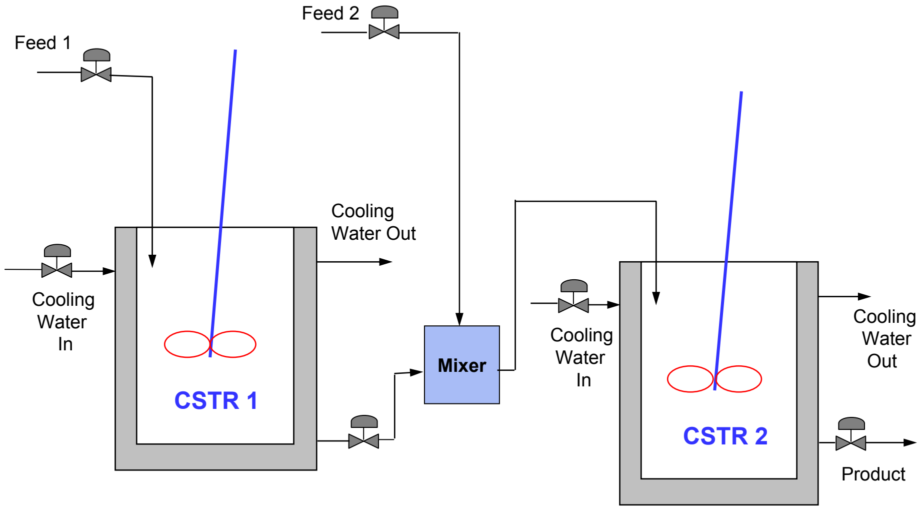

Figure 5.2: The two CSTR Process

5.4.1 Modelling of Two CSTR's in Series

A first–order irreversible exothermic reaction:

$$\mathbf{A} \longrightarrow \mathbf{B} \tag{5.22}$$

is carried out in the process. The reactors are cooled by cooling water (temperatures as T_{CW1} and T_{CW2}) within the jacket surrounding each reactor. The mixture densities and heat capacities are assumed to be constant and independent of temperature and concentration.

Thus the process can be modelled in terms of the concentration of the raw material (A) by the nonlinear differential equations given in Appendix B. Note that, the original eight states of the model are reduced to six assuming that a constant volume is applied to the process. The six states are: $x_1 = C_{o1}$ outlet concentration of CSTR 1; $x_2 = T_{o1}$, outlet temperature of CSTR 1; $x_3 = T_{CWo1}$, cooling water outlet temperature of CSTR 1; $x_4 = C_{o2}$, outlet concentration of CSTR 2; $x_5 = T_{o2}$, outlet temperature of CSTR 2; and $x_6 = T_{CWo2}$, cooling water outlet temperature of CSTR 2, respectively. The physical process constants are given in Table B.1 and the process is assumed to be operated at the equilibrium point given in Table B.2 (see Appendix B).

5.4.2 Input–Output Specifications

The input variables u can be chosen from Q_{I1}, Q_{I2}, Q_{cw1}, and Q_{cw2}. The state variables $x_2 := T_{o1}$ and $x_5 := T_{o2}$ are chosen as the controlled (measured) variables. These were chosen since economic analysis of this case study for control structure selection [CB96] shows that the optimal operation point with respect to economics is very sensitive to variation in the outlet temperatures of the reactors, hence these need to be controlled. For this output specification, two possible two–input, two-output configurations, named S1 and S2 (see Table 5.1) are considered in this work. In Table 5.1, Q_{I1} and Q_{I2} are the inlet flowrates of CSTR 1 and CSTR 2 respectively, and Q_{CW1} and Q_{CW2} are the cooling water flowrates of CSTR 1 and CSTR2 respectively. For both configurations the cooling water temperatures T_{cw1} and T_{cw2} are considered as disturbance variables.

Table 5.1: Input and output specifications

Name	u_1	u_2	y_1	y_2
S1	Q_{I1}	Q_{I2}	T_{o1}	T_{o2}
S2	Q_{CW1}	Q_{CW2}	T_{o1}	T_{o2}

5.5 Process Control Specification

The control objective in the two–CSTR process is to maintain both tank temperatures at the desired values in the presence of

1. Cooling–water temperature $\pm 10^o K$fluctuations in T_{CW1} and T_{CW2} in the presence of actuator constraints.
2. Setpoints change in the two output variables in the presence of actuator constraints.

where the actuator constraints of systems S1 and S2 are given in the expressions below

$$\text{for S1} \begin{cases} Q_{I1} + Q_{I2} \leq 0.8 \ (m^3 s^{-1}) \\ Q_{I1} \geq 0.05 \ (m^3 s^{-1}) \\ Q_{I2} \geq 0.05 \ (m^3 s^{-1}) \end{cases} \tag{5.23}$$

$$\text{for S2} \begin{cases} 0.05 \ (m^3 s^{-1}) \leq Q_{CW1} \leq 0.8 \ (m^3 s^{-1}) \\ 0.05 \ (m^3 s^{-1}) \leq Q_{CW2} \leq 0.8 \ (m^3 s^{-1}) \end{cases} \tag{5.24}$$

Cao and Biss, [CB96] designed a multi–loop PI controller to control the two system configuration under regulating control. The PI controller designed for the system was successful in rejecting non–zero mean disturbances ($\pm 10^o K$ in T_{cw1} and T_{cw2}), but had somehow a long settling time ($\cong$ 75 sec) and high peaks ($\cong$ [1.25 0.86] ^oK) although S1 configuration was able to reduce the disturbance in a shorter interval and with less peak than S2.

Cao and Yang (2004), designed a linear optimal controller using the H_2 and H_∞ norm to control the process in three control configurations (S1, S2, and S3) in a disturbance rejection test [CY04]. In this test, the proposed linear optimal controllers were able to reject the disturbance effects in a short time with some small peaks compared with the PI controller. Setpoint tracking tests were not included in that work.

In this thesis, the linear optimal controller above was tested at first to control the two–CSTR process in S1 and S2 configurations at setpoint tracking test. In this test, a step changes in the nominal values of $+2^o K$ at $t = 2$ sec is assumed. The controller was able to control the S2 system configuration. Long settling time ($\cong$ [26 23] sec), and approximately high overshoots [$\cong$ [57% 55%]) are observed for this case. The system performance was unstable and poor in the case of S1 configuration. As a result linear and nonlinear MPC are designed in this work to control the two–CSTR process at both the servo and the regulating problems as reported in the coming sections.

5.6 Process Control Using NMPC

The proposed NMPC algorithm is used here to control the two–CSTR process. In this process, only two from six state variables can be measured (*i.e.* $x_2 = T_{o1}$ and $x_5 = T_{o2}$) while the other four states (*i.e.* $x_1 = C_{o1}$, $x_3 = T_{cwo1}$, $x_4 = C_{o2}$ and $x_6 = T_{cwo2}$) are not. Thus, a state estimate stage is added to the control loop to estimate the unmeasured states. Three types of state observers were considered for this job. These are; the extended Kalman filter, the state estimators first-order divided difference (DD1), and second-order divided difference (DD2) filters [NPR00] using second-order divided difference filters developed by Norgaard et al. [NPR00]. The three estimators which are available in a MATLAB toolbox called KALMTOOL [Nor01], worked with the same accuracy in the case of two–CSTR process, but EKF was faster. The main difficulty with the EKF method over the others (which did not contain any differentiation stage in the theory) is the additional computation efforts required for the local 'Jacobian' linearization stage. This effort is reduced in this work for the case of two–CSTR using the forward mode of AD tool. The AD toolbox in MATLAB, MAD, is used to calculate the required partial derivatives automatically and simultaneously with the function values. Also, the linearized model will also be more accurate because the derivatives are exact. An accurate model will be helpful in reducing the state estimation error.

The predictive control parameters (*i.e.* T_s, P, M, Q, and S) are chosen using online tuning and the final values are; for system S1; $T_s = 0.1$ sec, $P = 10$ time steps (1 seconds), $M = 2$ time steps (0.2 seconds), $Q = diag(50, 100)$ and $S = diag(2, 2)$. For system S2; the same sampling time is chosen, while $P = 5$ time steps (0.5 seconds), $M = 2$ time steps (0.2 seconds), $Q = diag(100, 150)$, and $S = I$ are chosen. The first–principle model given in Appendix B is used as the plant and also the internal model of the controller in this chapter.

In the disturbance rejection test, all disturbance variables were considered measured (unmeasured disturbances are considered in the next chapter). The proposed NMPC algorithm is tested on the two configurations of the two–CSTR process. Figures 5.3 shows S1 and S2 performance in the present of measured T_{cw1} and T_{cw2} disturbances with positive step change $+10\ ^oK$ applied at $t = 2$ sec. System S1 needed longer time to reject the disturbance than S2 as a result of the selection of the manipulated variables and the NMPC tuning parameters.

In a setpoint tracking test, the performance of S1 and S2 systems using NMPC is given in Figure 5.4. The results show that system S1 has faster response compared with S2 but the later was more stable than S1 if the prediction horizons of both are chosen equal to 5 time steps. Therefore a longer prediction horizon was needed in the case of S1 (=10 time steps) to ensure the response stability.

To examine the NMPC efficiency the same tests above are repeated on the S2 configuration using a linear predictive controller. The LMPC controller is designed using the Model Predictive Control Toolbox in MATLAB (`nlmpcsim` Simulink block) with the same parameters used for the NMPC approach. Figure 5.5 shows the two–CSTR/S2 response using LMPC approach at the measured disturbance rejection test. The linear predictive controller was able to reject the disturbances but that combined with a big overshoot and long settling time compared with the NMPC results in Figure 5.3 (solid lines). Also, in the setpoint tracking test, Figure 5.6 shows the two–CSTR/S2 response using the designed linear and nonlinear predictive controllers. Also, in this test the nonlinear controller performance is better than the linear one especially in the manipulated variables behaviors since strong *ringing* in u is observed when the LMPC is used. The previous results showed that NMPC is more efficient and reli-

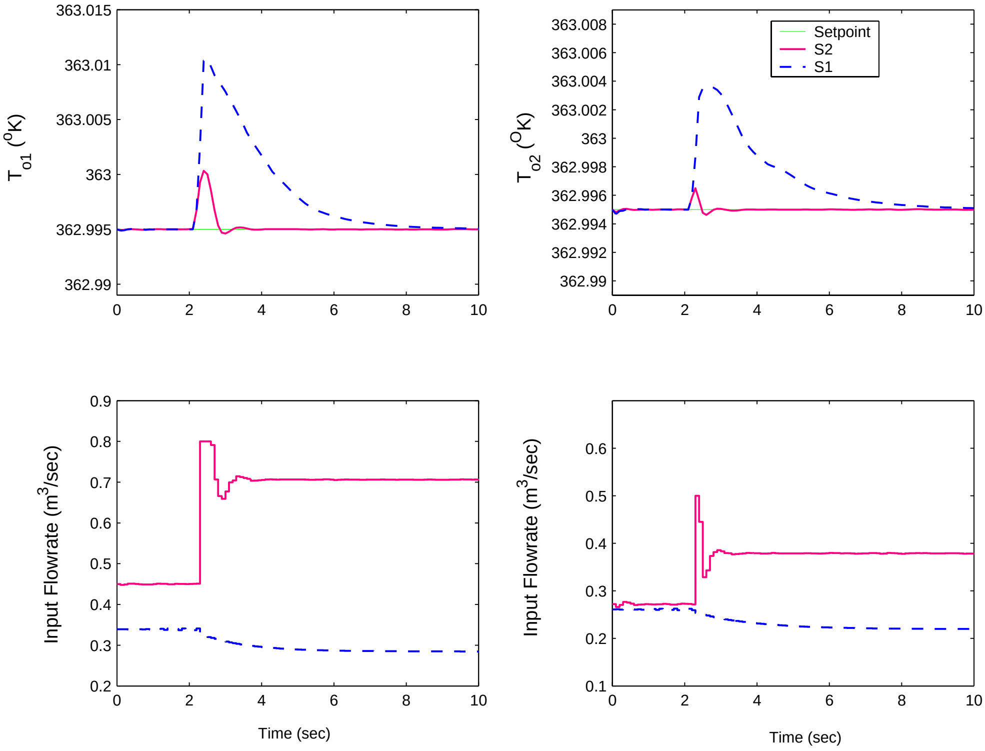

Figure 5.3: The response of two–CSTR process at measured disturbance in T_{cw1} and T_{cw2} (+10 ^{O}K step change at $t = 2$ sec) test NMPC.

able to control the two–CSTR process due to the accurate prediction of the system behavior using a nonlinear model which gives a better representative of the real plant.

To show the EKF capability to estimated the true values of the unmeasured states, errors in the initial states are assumed as $x_1 = -0.1$ mol/m^3 (true value= 0.084), $x_3 = 300$ ^{o}K (true value=327.56), $x_4 = 0.06$ mol/m^3 (true value=0.053), and $x_6 =$ 300.447 ^{o}K (true value=335.447). Figure 5.7 shows a comparison between the true and estimated states during the setpoint tracking test of S2 system. Good convergence to the true states is achieved after few time steps. Similar results was obtained for S1 system.

The main advantage of using AD in the linearization step of EKF is a saving in both the time and effort needed to calculate the required derivatives as this is done automatically via AD tool for any new application. In addition, a high accurate linearized model is be obtained using AD tool compared with the numerical differentiation. For the two–CSTR case study, the simulation time with EKF using AD is reduced to only 10% compared to the time using FD in model linearization case. This is because the dimension of the two–CSTR process is small (*i.e.* 2 inputs, 2 outputs, and 6 states). For higher dimension systems the time saving would be much more significant.

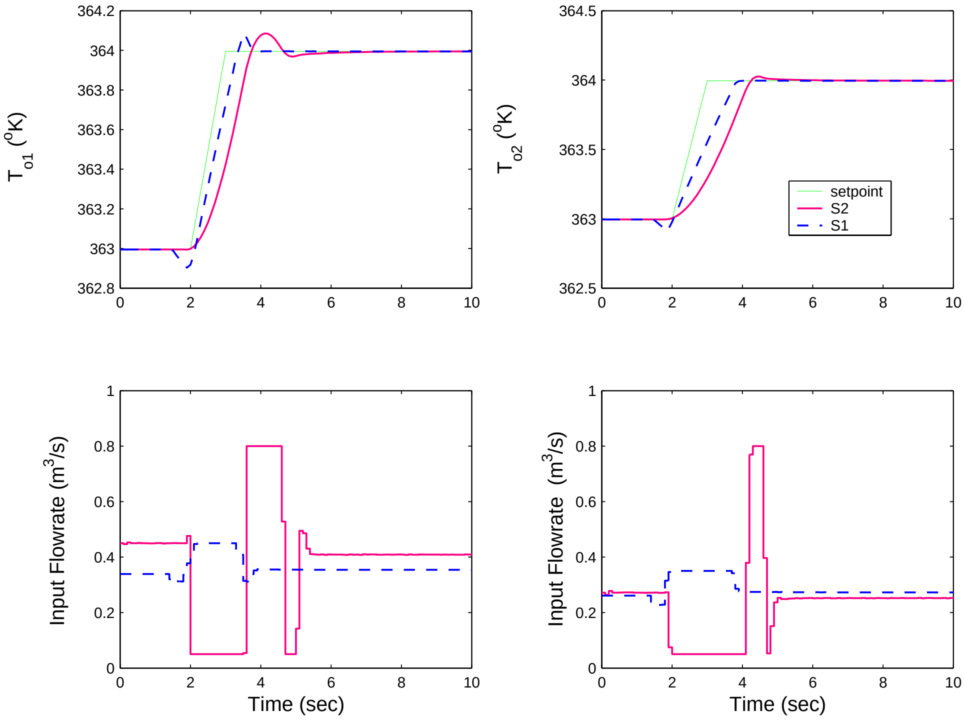

Figure 5.4: The two–CSTR S1 and S2 responses at setpoint tracking test.

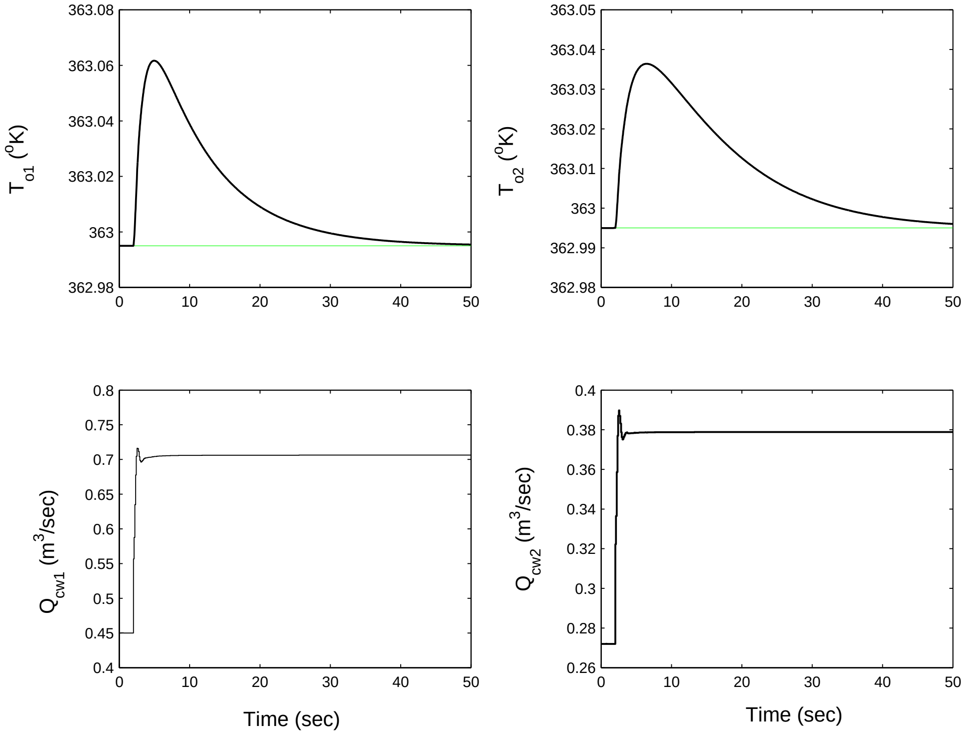

Figure 5.5: The two–CSTR/S2 response at T_{cw1} and T_{cw2} unmeasured disturbances test using LMPC.

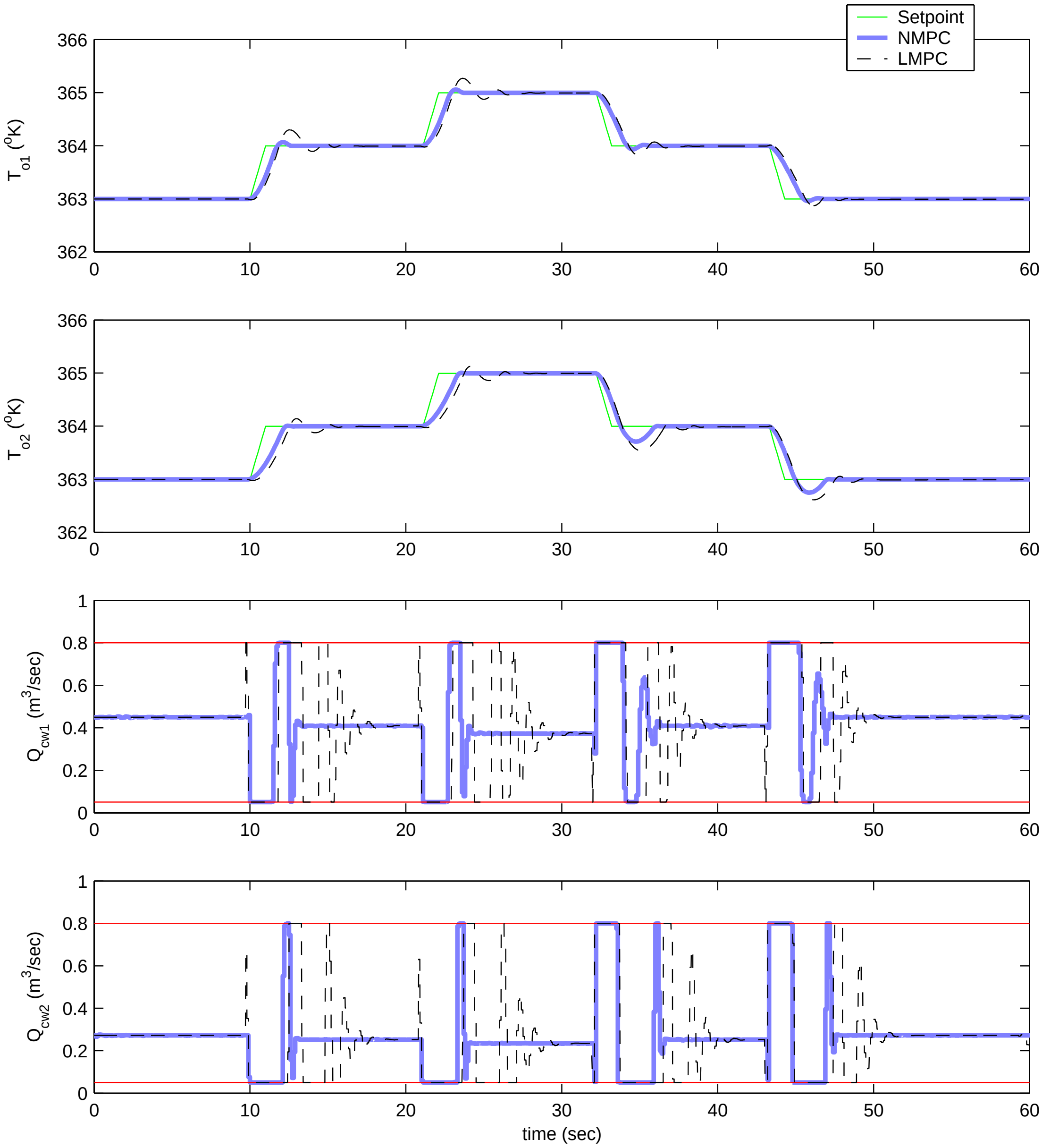

Figure 5.6: The two–CSTR/S2 response at setpoint tracking test using linear and nonlinear MPC.

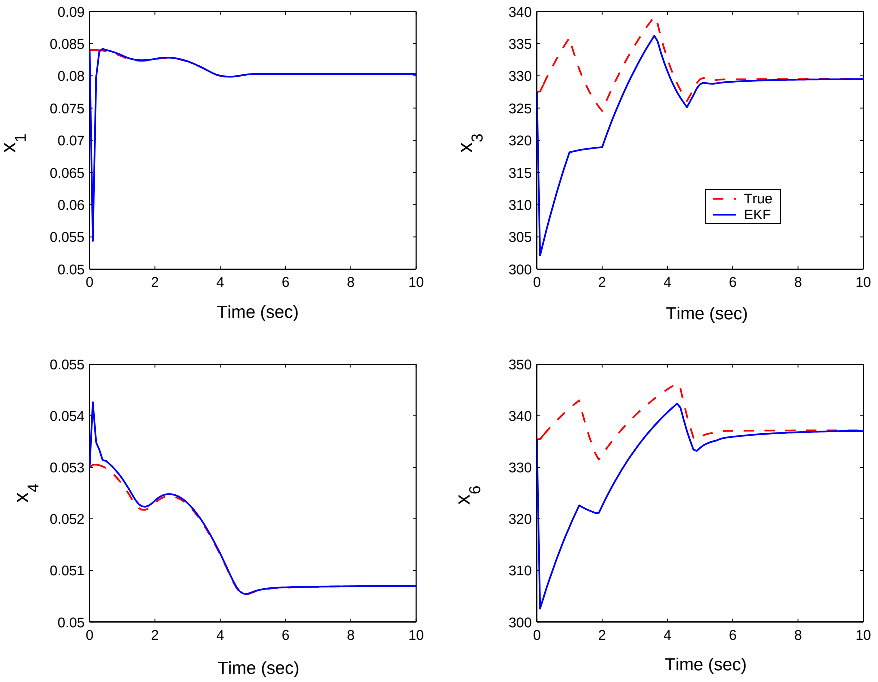

Figure 5.7: A comparison between the real and estimated state variables using EKF for the two–CSTR/S2 system at setpoint tracking test

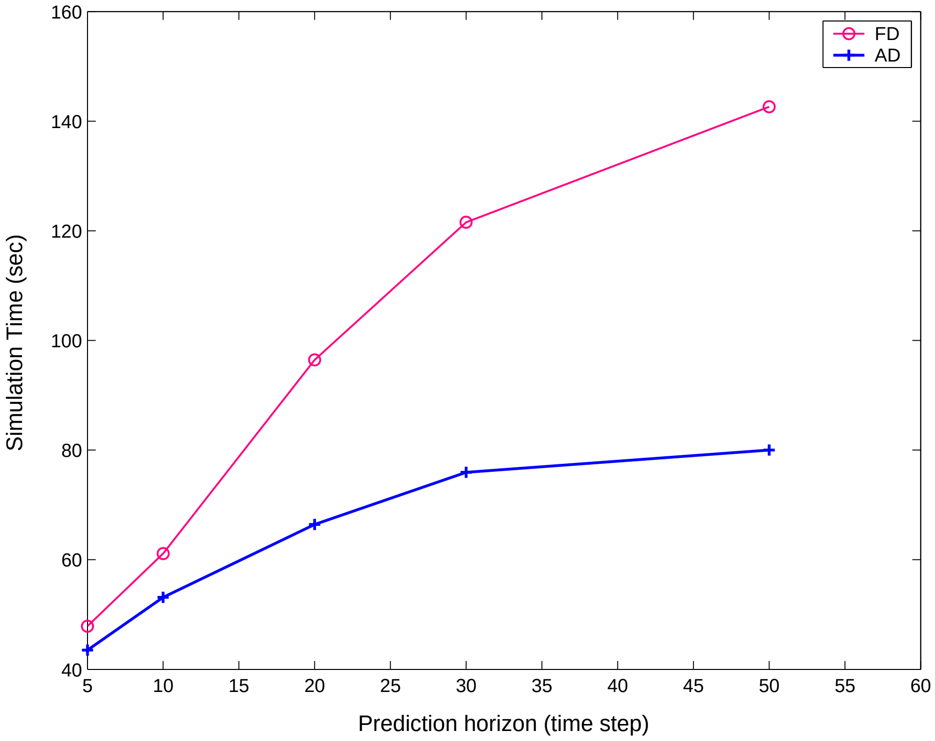

Figure 5.8: The CPU time comparison for the two–CSTR/S2 system at setpoint tracking test

In the optimization section of the proposed NMPC it is possible to use numerical methods to evaluate the sensitivity equations. This is used to check the difference in simulation time when using the the sensitivity approximation with AD approach compared to using a perturbation method (FD). Figure 5.8 shows a comparison in the CPU time used by MATLAB process to run the simulation results at setpoint tracking test of S2 system, at different prediction horizon lengths. When the prediction horizon is short the time saving using the sensitivity approximation method with AD is small but it increased rapidly when the prediction horizon increased.

5.7 Summary

The proposed NMPC algorithm is developed further to handle nonlinear processes with the presence of unmeasured state variables. In this case, a state estimate stage of the extended Kalman filter type is added to the predictive controller. The AD tool is used to develop the EKF state estimate. Two advantages can be obtained by using AD for calculating the Jacobian linearization matrices required for the filter. First, using AD gives as accurate calculation as the hand differentiation (analytically) but without the need to re-calculate this derivatives for every new application. This will be done automatically for any application which makes the EKF more general and so it can be used as a package directly for any plant. Second, the linearized model will be more accurate when AD is used compared with FD. Also this will be done in a shorter time which is very important to reduce the online calculation time of the NMPC. The new NMPC approach is used to control the nonlinear MIMO process of the two–CSTR process. This process presented in two control configurations, S1 and S2 depending on the manipulated variables selection. In this process only some of the state variables can be measured, so the developed EKF is used to estimate the remain unmeasured state variables. The controller capability to control the process without any constraint violation is proved in different operations conditions such as setpoint change and measured disturbances. The results are also compared with that of LMPC approach with large performance improvement using the proposed NMPC algorithm. Also the online computation time using the proposed NMPC is compared with another using FD for sensitivity calculation. The comparison shows a big time saving using AD tool especially in the case of long prediction horizons.

Chapter 6

Offset–Free NMPC Using Nonlinear Integration

6.1 Introduction

This chapter discusses the development of a new offset removal method to be used with the NMPC approach. In this method, the offset error is fed back to create a disturbance model which can be applied to the predictive model input or output to correct the model steady–state targets and eliminate the offset error. An adaptive nonlinear integrator is used in this case to improve the system responses during the offset error rejection period. The resulting NMPC is tested on two case studies; the evaporator process and the two–CSTR process in presence of unmeasured disturbances, model uncertainty, and measurements noise. In all cases, a response with zero offset errors is obtained and the developed integrating disturbances were also useful to enhanced the controller robustness as shown by the different tests.

6.2 Offset in NMPC

In some situations, the predictive controller manages to control the process adequately in both the setpoint tracking and regulating problems, but a small constant error (or offset) remain between the setpoint and the actual process output due to model/plant mismatch. Clearly this is not acceptable in many cases and a good controller should be able to produce an offset free result. This problem is well known even in simple linear controllers and is usually removed using an integral action that works on the offset between process output and setpoint value.

When a non–zero mean (non–stationary) disturbance affects a system, the origin of the state and input needs to be shifted in order to cancel the effect of such a disturbance on the controlled variables [MuR93]. To this aim, at each sample instant a disturbance model is created to correct (shift) the steady–state values of the predictive model as the actual steady–state of the plant. Such that one can drive the offset error to zero. In general, offset error in MPC algorithms could be caused by process/model mismatch due to unmeasured disturbances, model uncertainty, and /or finite prediction horizon [RST02]. A number of solutions can be used to eliminate this error such as increasing the prediction horizon length, or adding the difference between the measured outputs and model outputs at the latest time to the predicted outputs in the optimization step [Gar84]

$$y(k+i) = g\left(x(k+i), u(k+i)\right) + \widetilde{d}(k), \text{ for } \; i \in [1, P] \tag{6.1}$$

where

$$\begin{aligned} \widetilde{d}(k) &:= y_m(k) - y(k) && (6.2)\\ \widetilde{d}(k+i) &= \widetilde{d}_k, \text{ for } i \in [1, P] && (6.3) \end{aligned}$$

In many cases, increasing the prediction horizon is not sufficient to remove the offset error and also will increase the computation burden of the NMPC algorithm. On the other hand, adding the disturbance vector generated by comparing the plant and the model outputs does not necessarily remove the error completely in some applications. As reported by Meadows and Rawling [MR97] this method does not guarantee an offset–free performance in the presence of modelling errors.

Another way to eliminate the offset in linear MPC is by augmenting a disturbance vector with state observer vector, that shifts the target values of the desired states (through some observer gain matrices) into an input–output linearizing (IOL) controller [KH97]. Similar theory for a disturbance model has been proposed for linear MPC by Campbell and Rawlings [CR98], who categorized the disturbance models into three types, viz. generic disturbance model, output disturbance model and measured input disturbance model. They used integrating disturbance models to achieve offset–free control.

6.3 Offset–free Control via Parameter Adaptation

Motivated by the success of the parameter adaptation approach [ShD93, NKR97, RST02], two new simple offset removal techniques are proposed here to be used in NMPC context. In both, the error between the plant output and the desired value is used to estimate (calculate) an integrated disturbance which can be then added to the process model input (=d_I) or output (=d_o) as shown in Figure 6.1. These disturbances work to cancel the model steady–state shifting due to unmeasured disturbances, modelling error, or measurement noise. In this way, an integral action will be added to the MPC algorithm to produce zero offset error control. Note that, the block (D) is the disturbance model which can be static as in equation (6.2) or dynamic as the proposed disturbance model. The full description of these two methods is given below.

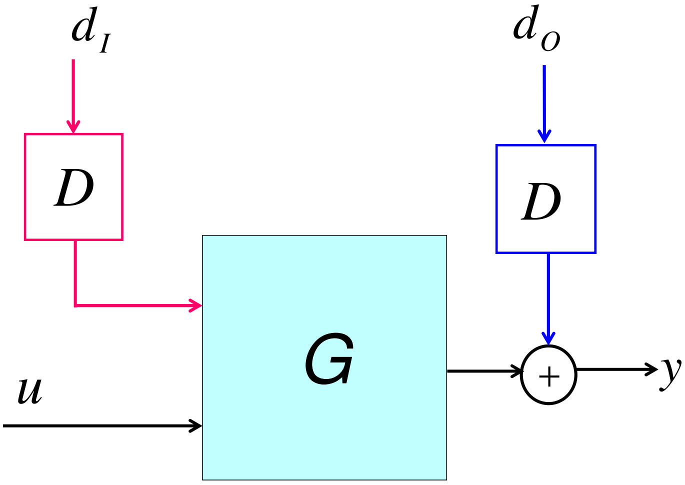

Figure 6.1: Simple diagram represents the way of adding the integrated disturbances to the process model

6.3.1 Disturbance Estimation Using Nonlinear Integration

The main reasons behind offset error in MPC is model/plant mismatch due to unmeasured disturbance or model uncertainty. For cases without such mismatching, the updating of state variables may provide offset–free control when using NMPC. It is clear that when it is possible to measure a disturbance, the offset error can be eliminate from the process outputs (see Figures 4.6 and 5.3). Since disturbances in practice might be unmeasured and unknown, a better technique to eliminate the offset is to estimate these variables and use this in the response prediction step. In this section, NMPC performance with zero offset error is obtained by estimating the unknown disturbances via nonlinear integration of the offset error.

In NMPC, integral action is required to integrate any small error to maintain the process variable at setpoint. This can be obtained by using the following adaptation technique to estimate the unknown disturbance. The adaptation dynamics is in the form of first–order differential equation;

$$\dot{\hat{d}}(t) = \gamma_c(e_{ss}(t), t) \tag{6.4}$$

where $\hat{d} \in \mathbb{R}^{n_{\hat{d}}}$ is the estimated value of the unknown disturbance (which is equivalent to d_I in Figure 6.1), $e_{ss}(t) := y_m(t) - y_r(t)$ is the offset error which is defined as the difference between the measured output vector y_m and the desired setpoint vector y_r. In this work, the term $\gamma_c(e_{ss}(t))$ has been chosen as a continuous nonlinear function of the offset error instead of using a linear gain which is the traditional case. The differential equation is approximated by Euler's method with step size equal to sampling time T_s as;

$$\hat{d}(k+1) = \hat{d}(k) + \gamma(e_{ss}(k)) \tag{6.5}$$

where;

$$\gamma := \int_{kT_s}^{(k+1)T_s} \gamma_c(e_{ss})dt \tag{6.6}$$

The initial value of $\hat{d}$ is chosen to be equal to the nominal value of the original unmeasured disturbances that need to be estimated. This value will be kept constant along the prediction horizon during solving the online NLP problem *i.e.*,

$$\hat{d}(k+i) = \hat{d}(k+1) = \hat{d}_{k+1}, \quad i \in [1, P] \tag{6.7}$$

The function $\gamma(\cdot)$ is a nonlinear function which can be any general continuous nonlinear function of the offset error $e_{ss}(k)$.

6.3.2 Nonlinear Gain

There is a broad range of options available for the nonlinear mapping γ. Here, the smooth sigmoidal–tanh function of the tracking error e_{ss} is chosen (see Figure 6.2). Mathematically this function is given as:

$$\gamma(e_{ss}) := K_1[\frac{2}{1 + exp(-K_2 e_{ss})} - 1] \tag{6.8}$$

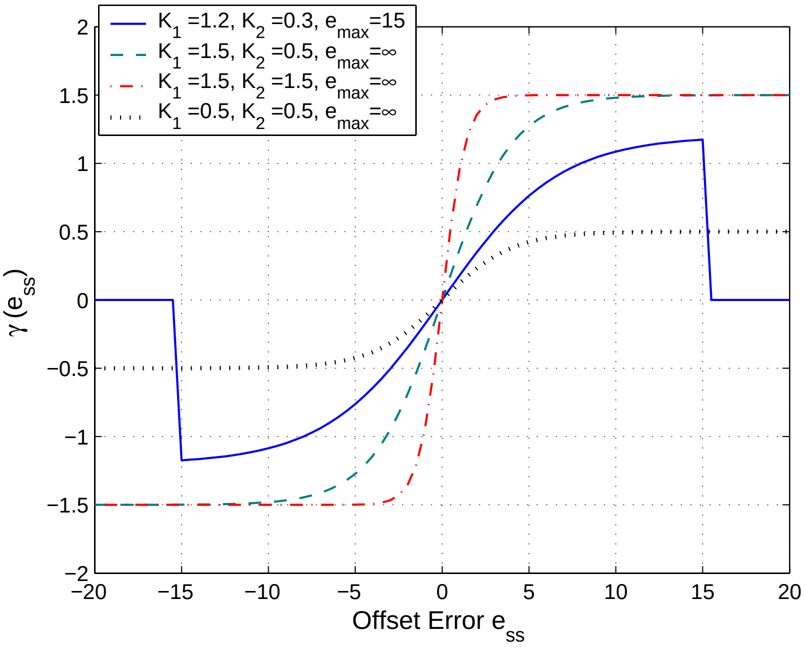

Figure 6.2: The nonlinear integration function for different values of gain parameters

where, $K_1 \in \mathbb{R}^{n_{\hat{d}} \times n_y}$, and $K_2 \in \mathbb{R}^{n_y \times n_y}$ are a positive constants. Re–write $\gamma(e_{ss}(k))$ as γ for simplicity, then the value of γ is lower–bounded by $\gamma_{min} = -K_1$ when $e_{ss} = -\infty$; and upper–bounded by $\gamma_{max} = +K_1$ when $e_{ss} = +\infty$, that is, $\gamma_{min} \leq \gamma \leq \gamma_{max}$; and $\gamma = 0$ when $e_{ss} = 0$. The term K_1 determines the range of variation of γ ($=\gamma_{max} - \gamma_{min} = 2K_1$) with $K_1 \geq 0$, and K_2 specifies the rate of variation (slope) of γ. Figure 6.2 shows a typical variation of γ as a function of e_{ss} with a different values of K_1 and K_2, and shows that γ has an S–shaped curve. It is noted that when using equation (6.8), the nonlinear value γ has equal excursions of $\pm K_1$ for positive and negative error e_{ss}.

The disturbance model above works at all times to remove offset error observed in the system. There is no information about the source of this offset if it is occurring due to unmeasured disturbance, or setpoint change. In the case of setpoint change this might lead to an additional overshoot in the output response due to the over excitation in the manipulated variables. This problem is solved by using another shape of the nonlinear function $\gamma(\cdot)$. This function is the same as (6.8) but with cut–off beyond a specified range of the error e_{ss} (the solid–line in Figure 6.2). In this case, it can be called as a *conditional nonlinear integration*. The conditional integration feature turns–off the integral action (*i.e.* the integrated disturbance changes) when the error is large (as the time instant where the setpoint begin changed for example), and turns–on the integral action when the error is sufficiently small (*i.e.* near a steady–state point):

$$\gamma(e_{ss}) = \begin{cases} \gamma & \text{if } e_{ss,min} \leq e_{ss} \leq e_{ss,max} \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

where $e_{ss,min}$ and $e_{ss,max}$ is the lower and upper bound of the offset error. Outside this range, the estimated disturbance $\Delta\hat{d}$ is set to zero. In this work $e_{ss,min}$ is chosen equal to $-e_{ss,max}$ thus reducing the number of tuning parameters.

The nonlinear function slope K_2 is chosen as a diagonal matrix with positive elements to reduce the number of tuning parameters too. Each element of K_1 and K_2 is chosen by considering the response of the predictive model to step changes in the unknown disturbances (through few open–loop simulations) and the signs of the gain K_1 elements are selected depending on the effect of each estimated disturbances on the corresponding output variable. Then few closed–loop simulations are carried out with different values for the parameters K_1 and $\hat{d}$.

The strategy of finding the suitable values of K_1 and K_2 is summarized as follows;

1. To find the best value of K_2 (*i.e.* the slope of the nonlinear function), a linear integration is considered at first as;

   $$d(k+1) = d(k) + K_L e_{ss}(k) \tag{6.10}$$

   where,

   $$K_L = \left.\frac{\partial\gamma(e_{ss})}{\partial e_{ss}}\right|_{e_{ss}=0} = \frac{K_1 K_2}{2} \tag{6.11}$$

   If each element of K_1 is set to 2, then $K_L \equiv K_2$ and only K_2 is needed to be tuned. The suitable value of each diagonal element of K_2 will be determined using simulation tests. If the values of the tuning K_2 are put too large unstable response will be detected so that it should be reduced slightly until a stable response is obtained. On the other hand, if the parameters are too small, a stable response will be detected but with sluggish behavior. The suitable values of K_2 are to be chosen between these boundaries to ensure stable response with fast changing rate. Note that, in this stage, small oscillation in the response with over damping response is acceptable because this behavior is to be considered when using the nonlinear integrator at the next step.

2. After determining the value of K_2, the linear integration is replaced by the nonlinear integration as in equation (6.8). The suitable value of K_1 can then be tuned online until a smooth and fast response is satisfied.

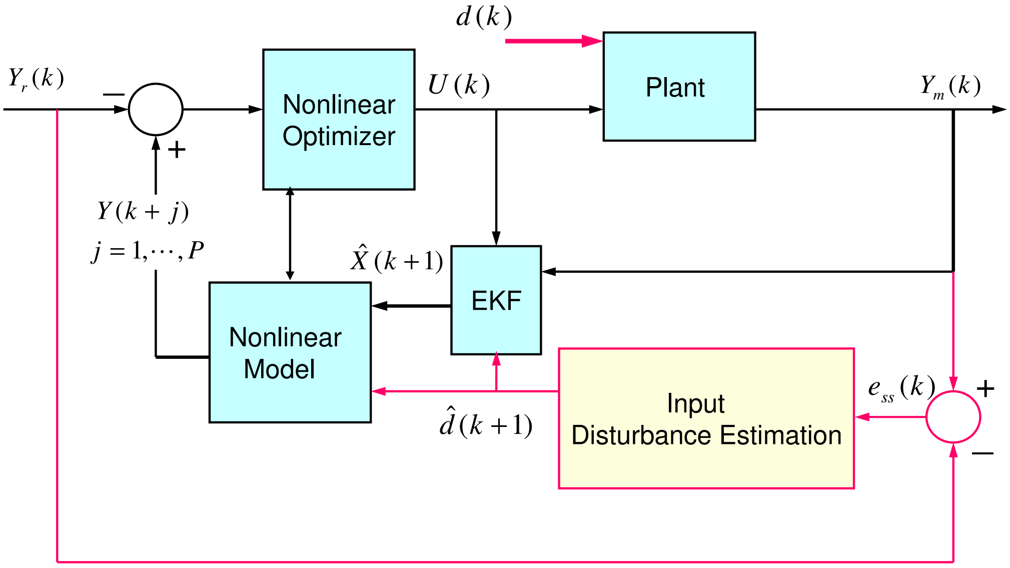

Figure 6.3: Simple diagram of the proposed NMPC after adding input disturbance integrator for offset–free control.

3. To avoid a large increase in the estimated disturbances prediction due to large tracking errors which may lead to closed–loop instability problem or large overshooting in the servo control problems, the conditional integration is used. The value of the positive boundary $e_{ss,max} = -e_{ss,min}$ is tuned as follows, at first it is set at high value then reduced slightly during tests and the final value kept constant depended on the quality of the output response.

Note that, incorporating the estimated value of the unmeasured disturbances in the process model will not only increase the prediction accuracy, but also the EKF predictions, since such state estimation needs the same nonlinear model to generate the time–varying linearized model. In fact, any error in the initial states of the open–loop optimal problem could lead to the wrong solution or to an unstable response. Therefore, a correct estimation of the unknown disturbances will help the estimator to find the correct states as well as to reduce the prediction error in the controller.

A simple diagram about how to link the offset removal part to the original NMPC is shown in Figure 6.3.

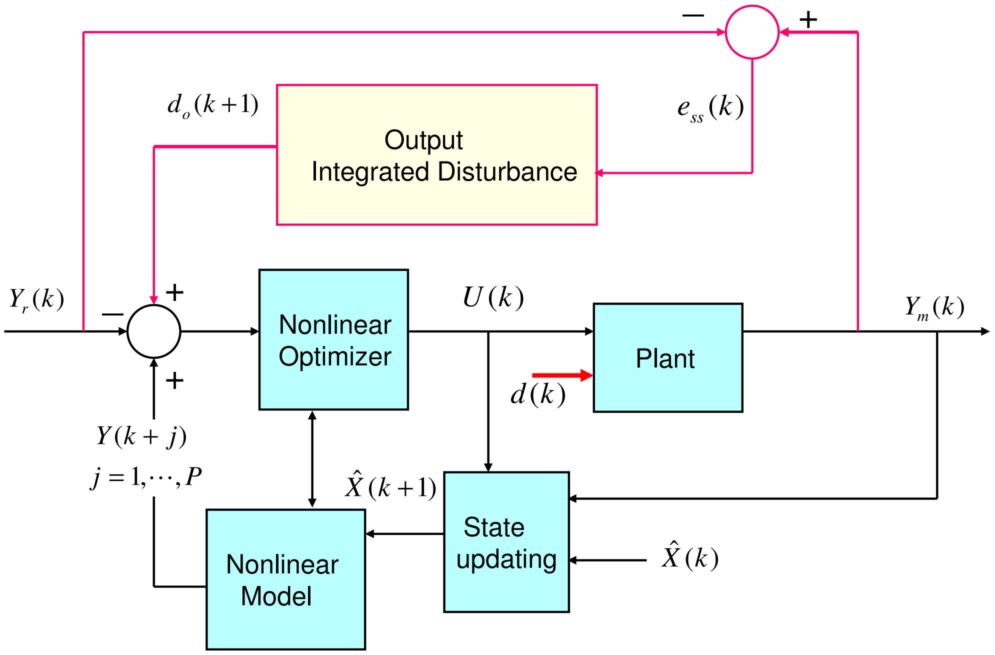

Figure 6.4: Offset free NMPC using output disturbance integrator.

If the number of estimated disturbances $n_{\hat{d}}$ is equal to the number of outputs n_y the matrix K_1 can be chosen as a diagonal matrix to reduce the number of tuned parameters. However, a large number of tuning parameters for the K_1 matrix will be needed if n_d is greater than n_y. This is one of the drawbacks of the present method which can be solved if an optimal problem is solved off–line to tune K_1 and K_2 or by using a self—tuning approach online.

6.3.3 Output Disturbance Modelling Using Nonlinear Integration

In this case, the adaptive integrator approach is used to create an output *virtual* disturbance which is then added to the predicted output to shift the process model steady–state targets to the correct place. The output disturbance model is calculated using the same formula of the input disturbance approach (*i.e.* equations (6.5) to (6.9)). The only difference in this case is that, K_1 is a diagonal matrix with n_y dimension and its elements are all positive. The estimated output disturbance $d_o \in$

$\mathbb{R}^{n_y}$ will be added to the process model output as follows;

$$\begin{aligned} d_o(k+1) &= d_o(k) + \gamma(e_{ss}(k)) && (6.12) \\ d_o(k+i) &= d_o(k+1) = d_{k+1,o}, \quad i \in [1, P] && (6.13) \\ y(k+i) &= g(x(k+i), u(k+i)) + d_{k+1,o}, \quad i \in [1, P] && (6.14) \end{aligned}$$

Since, the dimension of d_o is always equal to the dimension of the measured outputs and the matrices K_1 and K_2 are always positive and diagonal, the effort required to tune the adaptation parameters is less than the input disturbance estimation approach. The second advantage of using the output integrated disturbance is that, the state estimation stage (*i.e.* the EKF stage) can be removed and replaced by a simpler state updating step (*i.e.* $x_{k+1} = \int_0^{T_s} f(x_k, u_k)dt$) and the remaining estimation error will be corrected via the output disturbance d_o. This method is implemented here in the control of the two–CSTR case study and it is shown that the EKF is not necessary there.

The tuning strategy of the nonlinear integration function in this approach is similar to the previous case and can be obtained by online tuning as before. The simple diagram of the NMPC algorithm with the output integrated disturbance is shown in Figure 6.4.

Note that the two proposed offset removal approaches can be used together in parallel in some difficult applications.

6.4 Example 1: Evaporator Process

To examine the proposed NMPC robustness, an additional test is performed to the evaporator system by applying a non–zero mean disturbance of F_1 (-20% step change at t = 10 minutes). Note that F_1 is assumed to be unmeasured disturbance in this chapter. The process response in this test without using any offset removal technique is shown in Figure 6.5 (solid lines). Offset errors can be observed on the output responses due to the plant/model mismatch caused by the unknown disturbance F_1. It is noted here that the offset persists even when the exact set of ODEs of the evaporator system are used as the actual plant and also the predictive model (except ignoring the actuators lags in the predictive model). In addition, this happens even when all the states are measurable in this process.

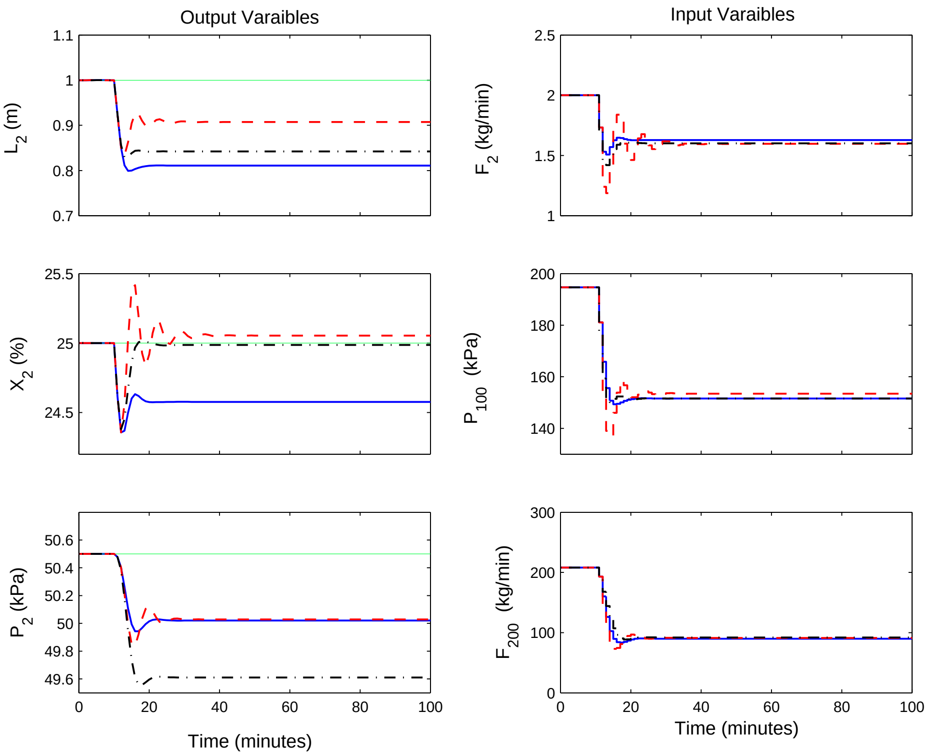

Figure 6.5: Evaporator response at unmeasured disturbance test; NMPC without offset removal stage (solid lines); NMPC with output disturbance (the difference between the plant and the model outputs) (dashed lines); NMPC after increasing the prediction horizon from 5 minutes to 50 minutes (dash–dotted lines).

The regulatory performance of the NMPC shown in Figure 6.5 is far from satisfactory because of the offset error problem in the three output variables. The problem of offset error is not solved when adding the difference between the measured outputs and model outputs at the latest time to the predicted outputs in the optimization step (dashed lines in Figure 6.5). On the other hand, increasing the prediction horizon from 5 to 50 minutes in this test eliminates the offset error in X_2 variable, but could not do the same for L_2 and P_2 variables, leading to an even bigger offset in P_2 (dash–dotted lines in Figure 6.5).

6.4.1 Offset–free NMPC/Input Disturbance Estimation

Offset practically disappears if the disturbance F_1 is measured and known to the controller calculations as shown previously in Figure 4.6. The challenge is for the case where the disturbance is unmeasured or not known to the controller. The proposed offset removal techniques based on input disturbance estimation is used with the

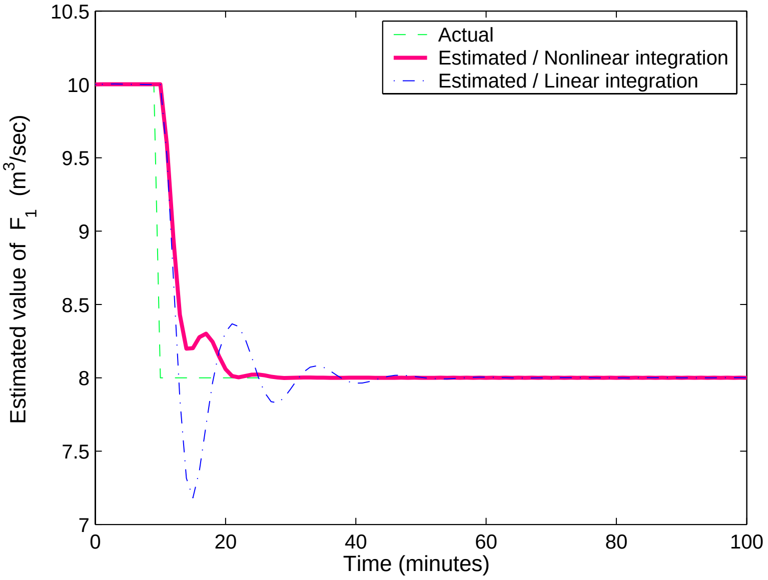

Figure 6.6: Input disturbance estimation using linear and nonlinear integral action with the NMPC algorithm at -20% step change in F_1 at $t = 10$ minutes.

original NMPC algorithm to solve the offset problem of the evaporator system.

The quantity F_1 is chosen as the model parameter to be updated irrespective of the actual disturbance in the real process. The adaptation law described in equation (6.5) provides updated $\hat{F}_1$ (the estimated value of F_1) for use in the internal model of the predictive controller. The nonlinear integrator parameters are chosen using online tuning and the values were fixed at $K_1 = [0.75\ 2.0\ 0.1]$, and $K_2 = I$. At the start of a large setpoints change which is usually known to the controller, it is desirable to switch off the contribution of the nonlinear integrator leaving the controller to work freely until the error reaches a predetermined small amount. These values $e_{ss,max} = -e_{ss,min}$ were set to $[0.1\ 1\ 1]$.

The actual and estimated value of F_1 are plotted in Figure 6.6 showing the capability of the proposed controller to find the correct value of the unknown disturbance after few calculation steps. Using this value in the model prediction removes the offset error as shown in Figure 6.7.

The NMPC performance is also given for the case when linear integration is used (the integrator gain is approximately similar to the tanh function slope). Both linear or nonlinear integration in the disturbance estimation step succeed in eliminating the offset error, however the linear integrator is incapable of accomplishing the two contradictory requirements of a fast response with no overshoot simultaneously. When the gain is a nonlinear function of the offset error, such as a sigmoidal–tanh function,

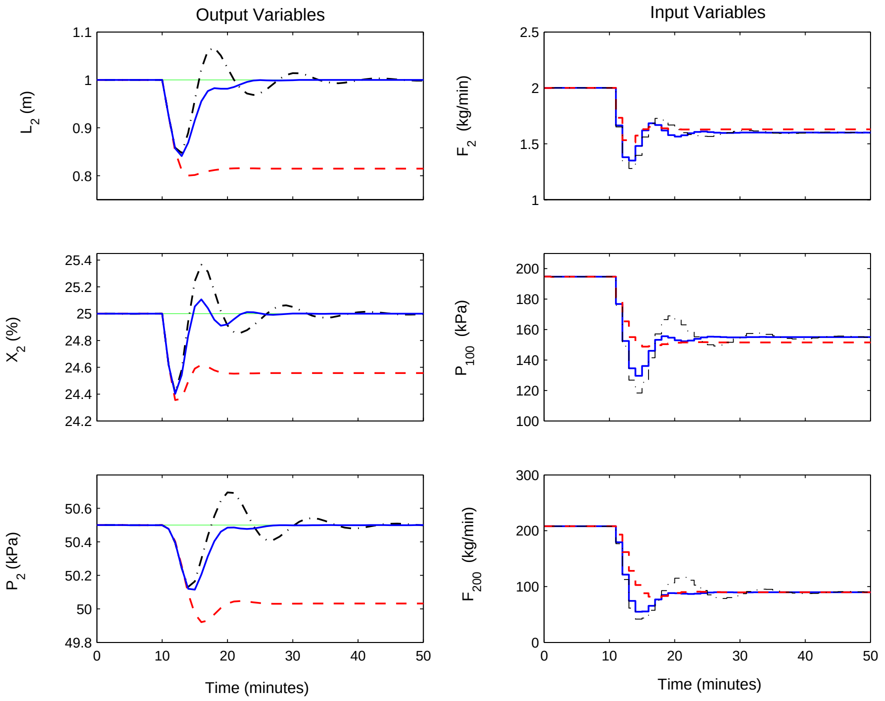

Figure 6.7: Evaporator response at step change in F_1 at $t = 10$ minutes; NMPC without integral action (dashed); NMPC using linear integration (dot–dashed lines); NMPC with nonlinear integration in the model input (solid lines).

initially the error e_{ss} between the desired value and the measured output increases; hence the gain γ will be increased too, producing a fast response. If the error is too large the gain γ will be saturated on the value $\pm K_1$ producing a stable response with small overshoot. As time proceeds and e_{ss} is diminished, the gain γ will be reduced automatically. Therefore, the automatic adjustment of the gain γ as a function of the error e_{ss} produces fast response with small overshoot. This cannot be achieved by a linear (fixed–gain) integrator at the same value of the gain. Figure 6.6 shows that the transient response is fast in linear gain but with a lot of oscillation, high overshoot (=2.735 kg/min), and long settling time (=44 min.) while these values are reduced using nonlinear gain as a settling time is down to 18 minutes with no overshoot. The results show clearly that a nonlinear gain improves the system response considerably by speeding up the transient response without causing oscillations.

6.4.2 Offset–free/Output Integrated Disturbances

A persisting offset error due to the unknown non–stationary disturbance in F_1 can also be removed using the proposed offset–free control via added output integrated

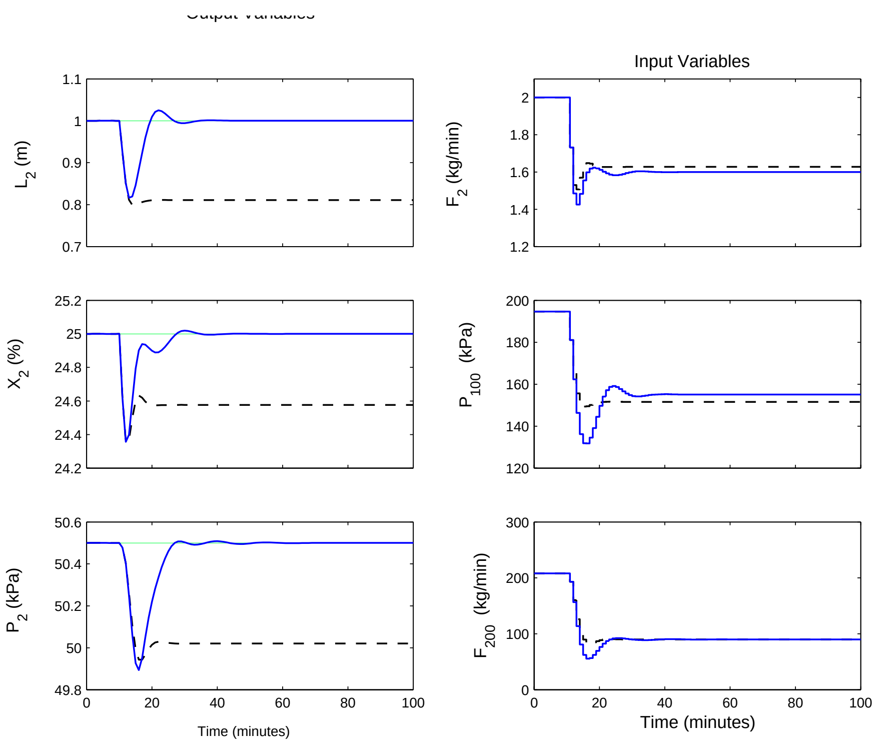

Figure 6.8: Evaporator response at unmeasured disturbance test using; NMPC without integrated disturbance (dashed lines), NMPC with the proposed output integrated disturbance (solid lines).

disturbance models as shown in Figure 6.8 (solid lines). A good performance is obtained during unmeasured disturbance test and stepoint change test when the nonlinear gain parameters are chosen at $K_1 = diag(0.25, 0.25, 0.2)$, $K_2 = diag(1, 1, 1.5)$, and $e_{ss,max} = -e_{ss,min}$ =[0.5 1 1]. The virtual output disturbance vector $d_o \in \mathbb{R}^3$ is added to the model output vector to correct the prediction results in different operation tests which are plotted in Figure 6.9. In the case of unmeasured disturbance test, the disturbance d_o changes from its initial value [0 0 0] to a new steady-state values [-0.1856 -0.4466 -0.4694] which compensates for the steady–state shifting in the internal model due to the unknown disturbances F_1 during the prediction phase. As a result the optimizer can decide the correct manipulated variables movements without producing offset error as shown in Figure 6.8.

The offset removal is also useful in increasing the system robustness in the presence of modelling errors. A small modelling error in C_p is introduced (the heat capacity of the liquid) by using 0.2 kW/K(kg/min) (the original value 0.07). The system response in the setpoint tracking test in the presence of this modelling error and using the NMPC without the offset removal routine shows offset errors on all the three plant outputs as shown in Figure 6.10 (dashed lines). Adding an output disturbance as the difference

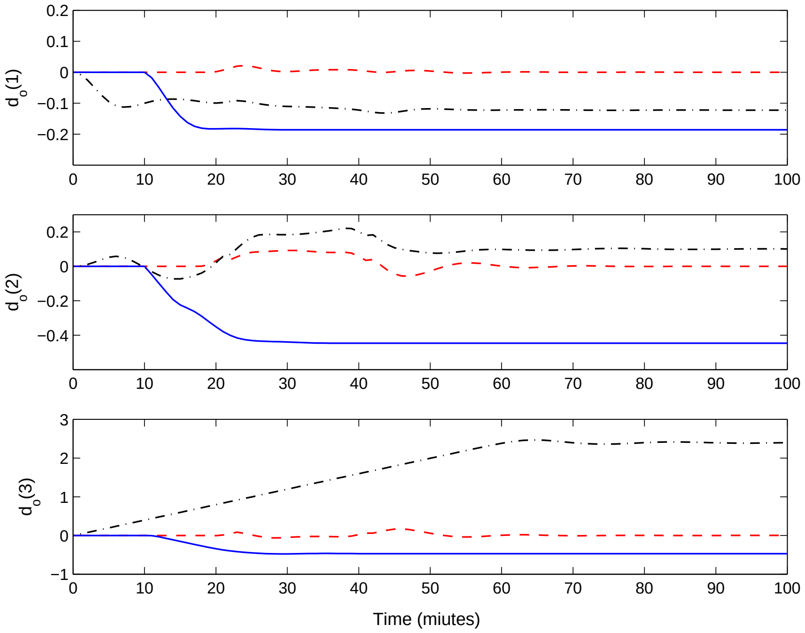

Figure 6.9: The output integrated disturbances required to obtained offset free response of the evaporator at different tests; (1) Unmeasured disturbance (solid lines) (2) Setpoint tracking (dashed lines) (3) Setpoint tracking in present of model uncertainty (dash–dotted lines).

between the plant and the model outputs to the predictive outputs reduced the offset errors slightly but not to zero (dash–dotted lines) in the same figure above. On the other hand, offsets are quickly removed by adding the proposed output integrated disturbances as shown in Figure 6.10.

Also, the NMPC using output integrated disturbance was successful in eliminating the offset errors (due to unmeasured disturbance in F_1 in present of modelling error in C_p) (the solid lines in Figure 6.11). It is noted that, the offset–free NMPC with input integrated disturbance (dashed lines in Figure 6.11) is unable to eliminate the offset error in this test. The reason behind this result comes from that, one adaptation parameter ($\hat{F}_1$) seems not sufficient to correct the steady–state shifting in three outputs ($i.e.$ $n_y > n_d$). This can be solved by choosing another model variable such as another disturbance as an additional input integrated disturbance but this of course will need more tuning parameters. Therefore, it could be said that offset–free NMPC using output disturbance models is more reliable, efficient, and easier to tune compared with the input integrated disturbance method because there will always be one output integrated disturbance for each one offset error ($i.e.$ the chance to find the solution is bigger).

In an additional test, the NMPC robustness is examined in present of measurement

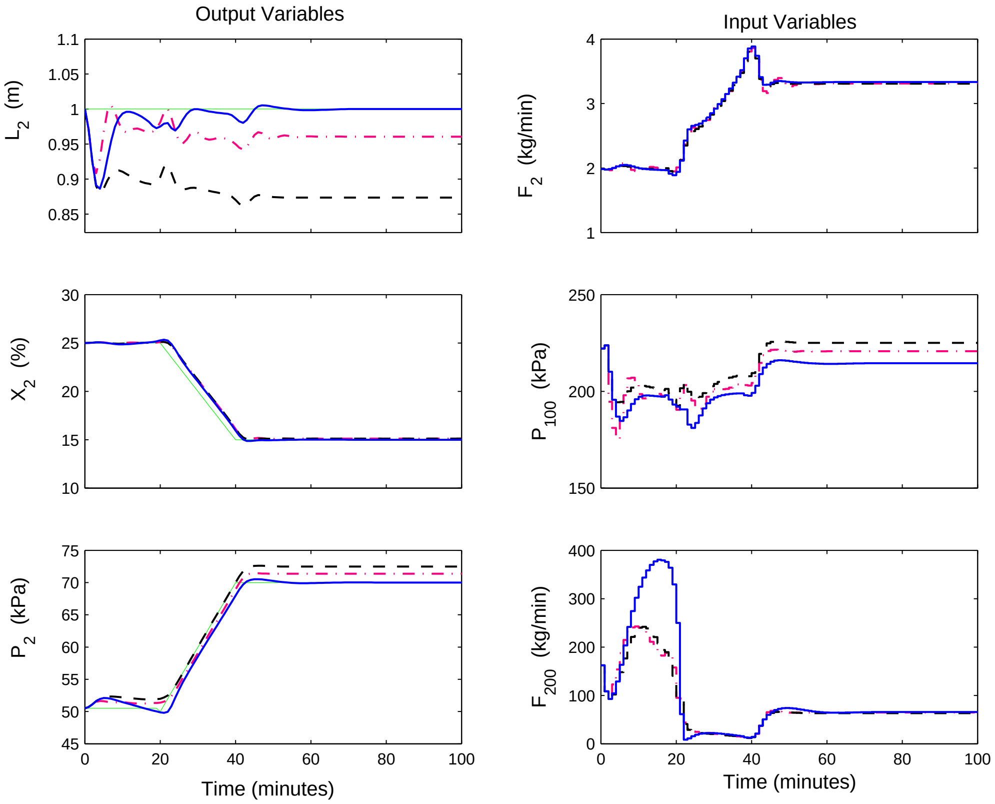

Figure 6.10: Evaporator response at setpoint tracking test with present of modelling error using; NMPC without adding disturbances (dashed lines), NMPC with output disturbance as the difference between the plant and the model outputs (dash–dotted lines), NMPC with output integrated disturbance (solid lines).

noise with covariance $R^{\omega} = diag(6e^{-4}, 0.4, 1.5)$. The response of the evaporator process in the presence of this value of measurement noise at setpoint ramp change is shown in Figure 6.12. In this case, output integrated disturbances are used for offset free NMPC. The result reflected the capability of the proposed NMPC algorithm to control the evaporator plant in this situation as well.

6.5 Example 2: Two–CSTR Process

The two–CSTR process is chosen to demonstrate the addition offset removal techniques. In a disturbance rejection test, all disturbance variables (T_{cw1} and T_{cw2}) are considered unmeasured in this case. The proposed NMPC algorithm without using integrated disturbances is tested first to control the two–CSTR/S2 process. Figure 6.13 shows the system performance in the presence of unmeasured disturbances as a positive step change $+10^{o}$K applied to the process input at $t = 2$ sec. In the response,

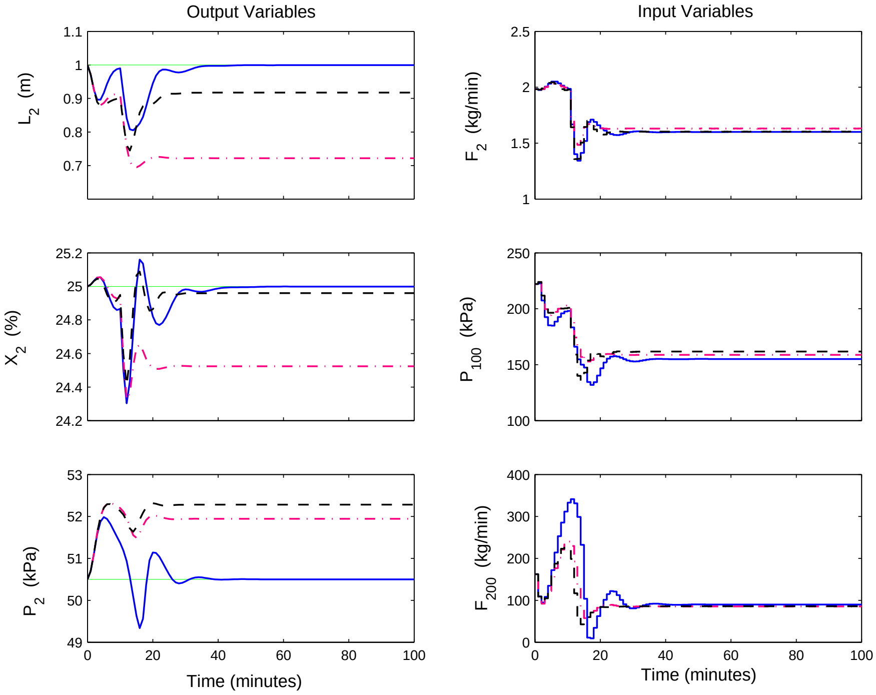

Figure 6.11: Evaporator response in the present of unmeasured disturbance and modelling error; NMPC without integrated disturbances (dash-dotted lines), NMPC with input integrated disturbance (dashed lines), NMPC with output integrated disturbance (solid lines).

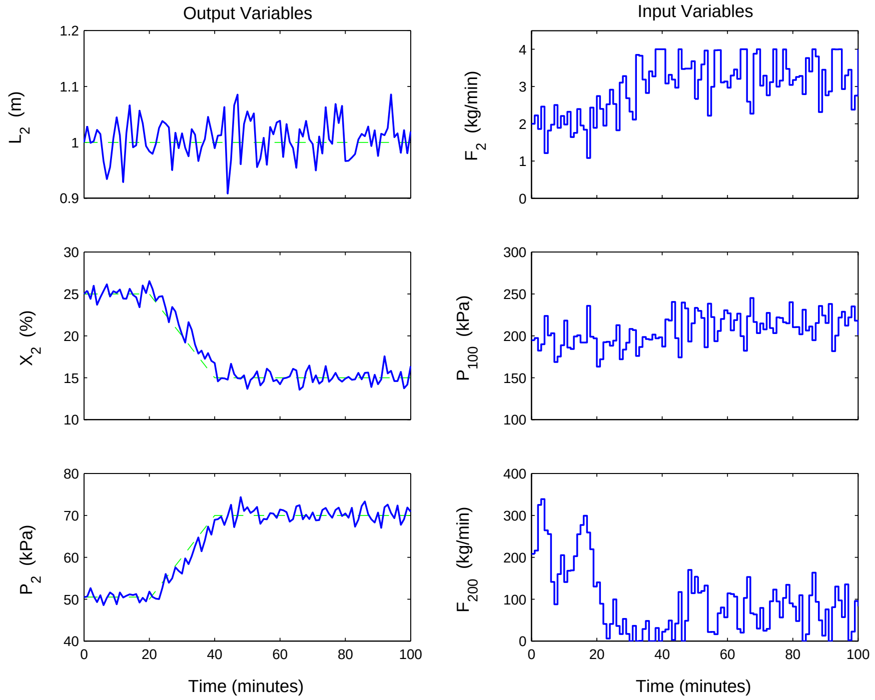

Figure 6.12: Evaporator response at setpoint ramp changes in present of measurement noise.

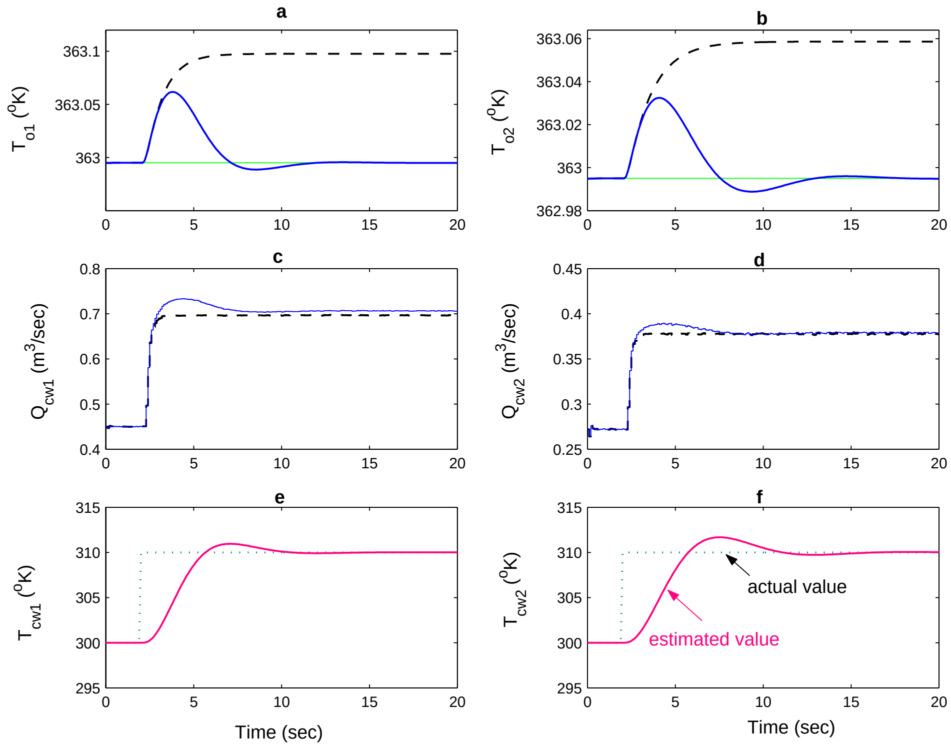

Figure 6.13: (a)–(d) Two–CSTR/S2 response at unmeasured disturbances test using; NMPC without integrated disturbance (dashed), NMPC with input integrated disturbances (solid). (e)–(f) actual and estimated disturbances.

big offset errors are observed in the output variables due to model/plant mismatching associated with unknown disturbances. Also, the same errors are observed in the estimation error of the states using EKF for the same reason as shown in Figure 6.14.

6.5.1 Offset–Free NMPC using Input Integrated Disturbance

The main reason for the offset error observed on the process outputs is the missing information about the actual values of the disturbances variables T_{cw1} and T_{cw2} for the process model which is used in both the predictive controller and the state estimator. This conclusion is supported by the fact that no such errors are observed when the disturbances are measured (see Figure 5.3). The solution for this problem lay in estimating the unknown disturbances values using the proposed nonlinear integrator of section 6.3.1 and used in the prediction calculation as follows.

The suitable values of the nonlinear integrator parameters are tuned online by running the simulations in different tests such as unmeasured disturbances, modelling error, and setpoint tracking tests. The final values of these parameters for S1 are; $K_1 =$

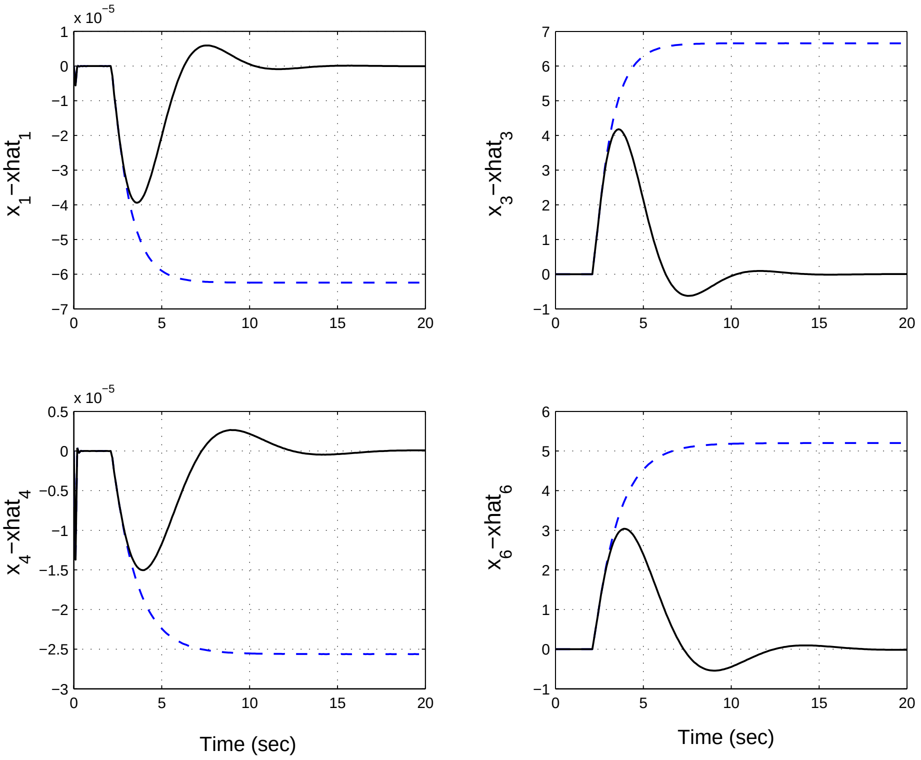

Figure 6.14: State estimation errors during unmeasured disturbances tests of the two–CSTR/S2 process using; NMPC without input integrated disturbances (dashed lines); NMPC with input integrated disturbances (solid lines).

$diag(10,10)$, $K_2 = I$, $e_{ss,max} = -e_{ss,min} = [0.5\ 0.5]$, and for S2: $K_1 = diag(8,8)$, $K_2 = diag(1.5,2)$, and $e_{ss,max}$ as in S1 system. The estimated values of T_{cw1} and T_{cw2} are applied to the model input instead of the nominal values in both the predictive model and the EKF to correct the model steady–state error. The resulting offset free output responses as shown in Figure 6.13. The corresponding state estimation errors are plotted in solid lines of Figure 6.14. It can be observed that, all the estimation errors converge to zero after a number of time steps. The actual and estimated values of the disturbances using the proposed nonlinear integration are given in the subplots (e)–(f) in Figure 6.13. This results show clearly the offset removal capability in finding the true values of the unknown disturbance with good transient response.

The NMPC algorithm is now tested for robustness to control the two–CSTR/S1 in the presence of setpoint change plus modelling uncertainty. Modelling errors are assumed in the values of some of the physical constants in the internal model as $V_1 = 3\text{m}^3$ (true value=4.489), V_2=6 m^3 (true value=5.493), $U_{a1} = 0.2$ m^{3}s^{-1} (true value=0.35), and $U_{a2} = 0.6$ m^{3}s^{-1} (true value=0.35). Without any integral action offset errors are observed on the two plant outputs as shown in Figure 6.15. These

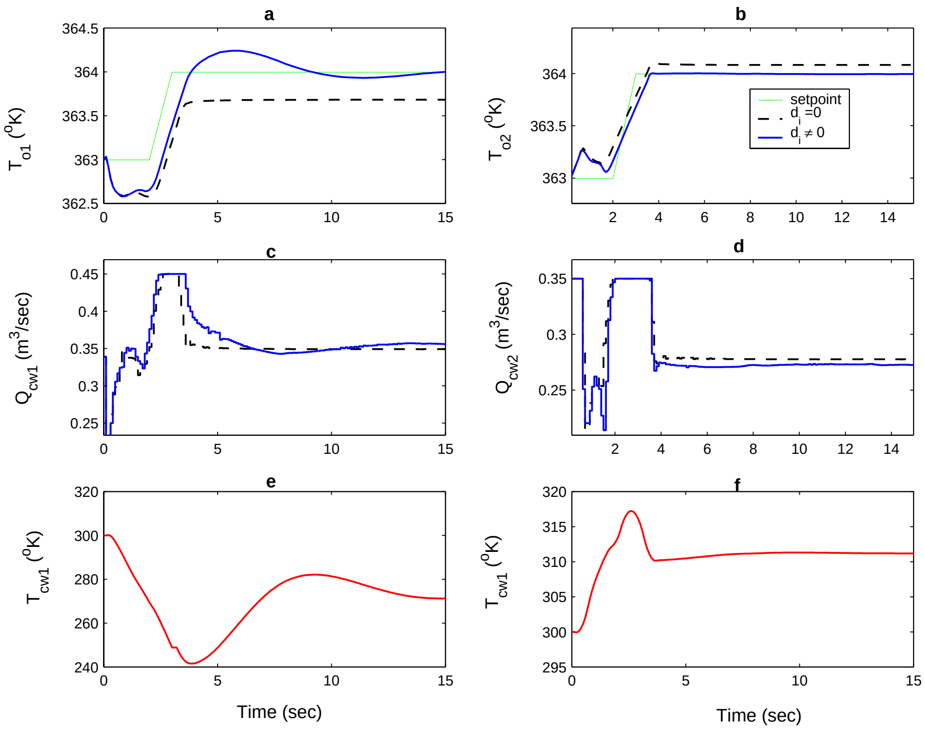

Figure 6.15: (a)–(d) TwoCSTR/S1 responses at setpoint tracking plus modelling error test using input integrated disturbances, (e)–(f) input integrated disturbances.

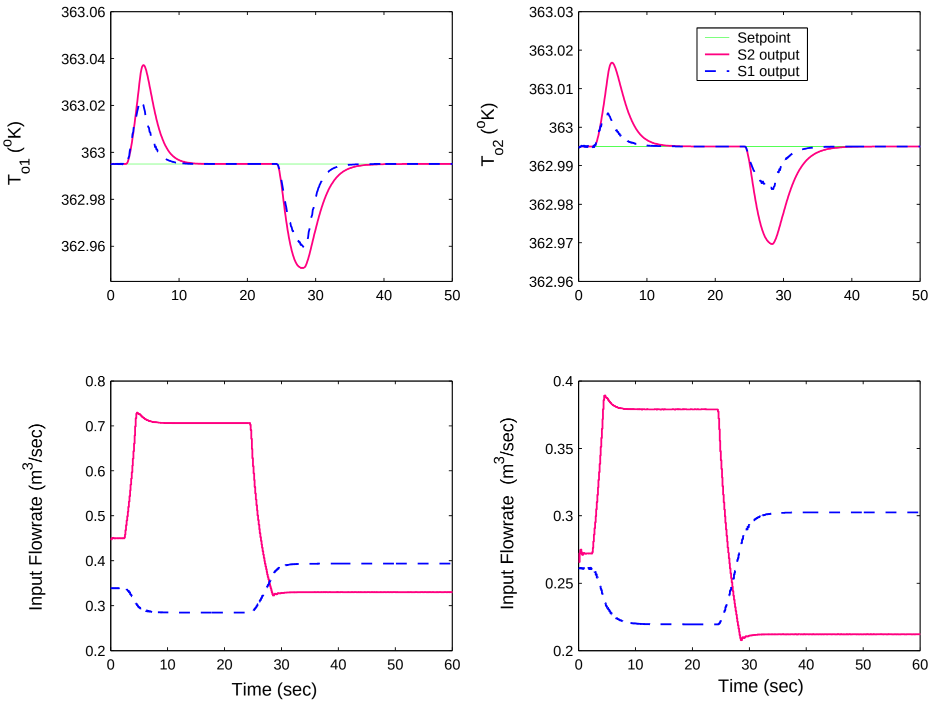

Figure 6.16: The two–CSTR responses during unmeasured disturbance test using offset free NMPC via output integrated disturbances.

offset errors are completely eliminated from the output response when using input integrated disturbances as shown in Figure 6.15. At the internal model, the estimated disturbances (or adaptation parameters in this case) $\hat{T}_{cw1}$ and $\hat{T}_{cw2}$ are shifted from their steady–state values to new values to compensate the steady–state error between the model and the plant as shown in the subplots e and f of Figure 6.15.

6.5.2 Offset–free NMPC using Output Integrated Disturbances

The other way to eliminated the offset error is by adding a virtual output disturbance model to the outputs to shifted its steady–state values to that of the plant. After online tuning the final values of the parameters are chosen as follows; $K_1 = diag(0.3, 0.6)$, $K_2 = I$, $e_{ss,max} = -e_{ss,min} = [0.2 \;\; 0.2]$, for S2 $K_1 = diag(0.1, 0.08)$, $K_2 = diag(1, 2)$, and $e_{ss,max} = -e_{ss,min} = [0.2 \;\; 0.2]$). Note that, in this method the EKF is removed from the close loop and replaced by simple state updates and the remaining estimation error will be corrected via the output disturbance model d_o. Figures 6.16 shows the offset free responses of the two control configurations of the process at unmeasured disturbances test ($+10\ ^oK$ step change in T_{cw1} and T_{cw2} at $t = 2$ sec followed by -10^oK step change at $t = 24$ sec).

The same method is used to eliminate the offset errors due to modelling error from system S1 outputs. This test is similar to that given in the previous section (*i.e.* setpoint change plus model uncertainty). The offset errors are completely eliminated from the output response when using output integrated disturbances as shown in Figure 6.17. The output integrated disturbances d_o in this case are moving in the direction that shift the steady–state values of the internal model to the correct place as shown in the third subplot of Figure 6.17.

The presence of measurement noise is examined next with covariance $R^\omega = diag(10^{-5}, 10^{-5})$. The response of S2 in the presence of this value of measurement noise at unmeasured disturbances test is shown in Figure 6.18. Note that output integrated disturbances are used for offset free NMPC. The result reflected the capability of the proposed NMPC algorithm to control the two–CSTR plant in this situation too. Similar performance is obtained with S1 system in different operation tests.

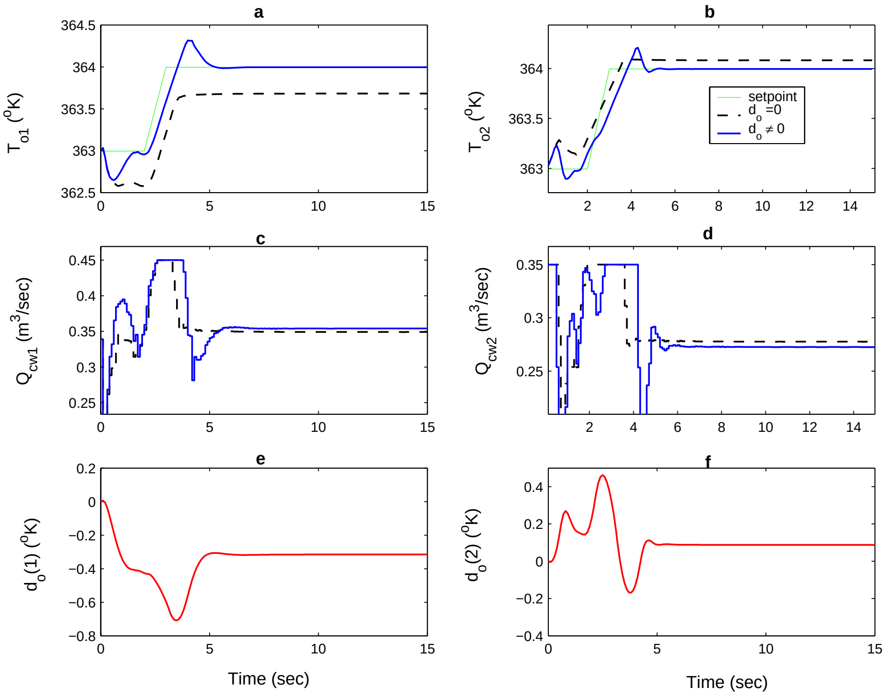

Figure 6.17: (a)–(d) Two–CSTR/S1 responses at setpoint tracking test in present of model uncertainty using NMPC with output integrated disturbances, (e)–(f) The output integrated disturbances.

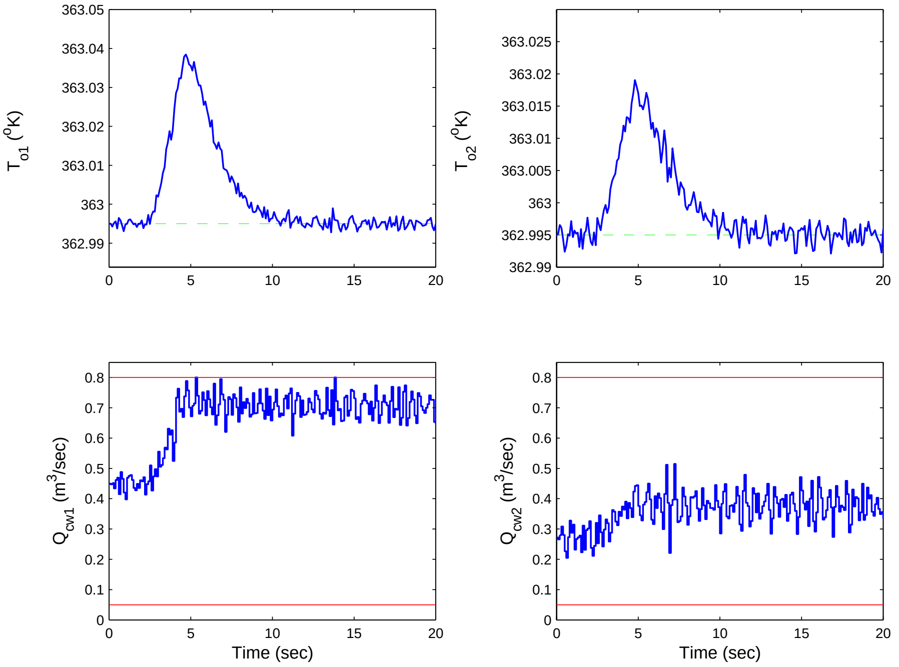

Figure 6.18: TwoCSTR/S2 response at T_{cw1} and T_{cw2} unmeasured disturbances test (+10 step change at $t = 2$ sec) in present of measurement noise.

6.6 Summary

Two integrating disturbance techniques are proposed and successfully applied to obtain offset-free NMPC. In the first type, the disturbances is injected in the process model input as an estimated value of the unknown disturbances. An offset free response is obtained after applying the integrated disturbances to the model input in few time steps. A similar performance is observed using an integrated disturbances on the process model outputs. The output disturbances are created from the current tracking error and used to shift the internal model steady–state target to the correct place. A nonlinear integrator represented by the sigmoidal–tanh function is used in both methods to improve the system transient responses during the offset error rejection time. Fast and smooth response is obtained using the nonlinear integration compared with a linear type. The evaporator process, and the two–CSTR process are chosen to demonstrate the new offset removal techniques. The proposed input and output disturbance were capable of removing the offset error due to unmeasured disturbance or/and model uncertainty. The results show that the output integrated disturbance technique was more practical and easier to tune than the input integrated disturbance technique.

Chapter 7

NMPC for a Black Box Process using High–Order Taylor Series

.

7.1 Introduction

In many practical applications of NMPC, a mathematical model based on physical principles is either unknown or too complicated to be used for control. In this case, nonlinear system identification is an inevitable step in a NMPC project. Sometimes, it is also the most costly and time consuming part of the project [ZGS98]. In this chapter a new efficient NMPC algorithm, and nonlinear system identification algorithm using a continuous–time recurrent neural network (CTRNN) model, are developed. In both algorithms, the ODEs and the dynamic sensitivity are solved simultaneously using Taylor series expansion and AD tool. The evaporator process and the two–CSTR/S2 process are chosen as applications for these methods.[1]

7.2 Nonlinear System Identification using CTRNN

An efficient and effective approach of nonlinear system identification is critical to the success of NMPC. Unlike linear systems, there is no uniform way to parameterize a

[1] The material in this chapter is the subject of accepted for publication and submitted papers e.g. [ASC05c] and [ASC06]

general nonlinear dynamic system. Among many existing techniques, the universal approximation properties of neural networks makes them a powerful tool for modelling nonlinear systems [FN93]. Most of the publications in nonlinear system identification use FFNN with backpropagation or some other variations for training, for example [TSM95, TC96]. The static FFNNs together with tapped-delay lines provide a means to model nonlinear dynamic systems in discrete–time [MSW90, NP90]. The main drawback of this approach is that it can only provide predictions for a predetermined finite number of steps, in most cases, only one step. This drawback makes such models not well suited for predictive control, where variable multi–step predictions are desired. RNNs on the other hand are capable of providing long range predictions even in the presence of measurement noise [SMc97]. Therefore, RNN models are better suited for NMPC. RNNs with internal dynamics has been adopted in several recent works. Models with such networks are shown [FN93, JNG95], to have the capability of capturing various plant nonlinearities. They have also been shown to be more efficient than FFNNs in terms of the number of neurons required to model a dynamic system of a certain order [DKW95, HH93]. In addition, they are more suitable to be represented in state-space format, which is quite commonly used in many important control algorithms [ZV98].

In this work, a continuous time version of the recurrent neural networks in state-space form is used as the internal model of NMPC. The continuous–time RNN brings further advantages and computational efficiency over the discrete formulation even if at the end both are represented on the computer using only discrete values [PeB95]. Using a discrete time RNNs causes a great dependence of the resulting models on the sampling period used in the process and no information is given about the model trajectories between the sampling instants. The sampling period used with CTRNNs, on the other hand, can be varied without the need for re-training [KaC00, KGW00].

The main difficulty with recurrent neural networks is in their training [PeB95, KaMC96, PBV03]. Various training strategies have been suggested in the literature, such as the backpropagation method [RuH86], the conjugate gradient method [LK99], Levenberg-Marquardt optimization [Mar63], or methods based on genetic algorithm (GAs) [Gol89]. Neural networking training is actually a nonlinear optimization problem. To solve the nonlinear optimization problem associated with CTRNN training, the calculation of a large number of dynamic sensitivity equations is required. Depending on the number of sensitivity equations involved, the sensitivity calculation could take more than 90 percent of the total computation time required for solving a training problem. Hence, sensitivity calculation is a bottleneck in training CTRNNs.

In this chapter, a CTRNN is developed to be used in a NMPC context. The neural

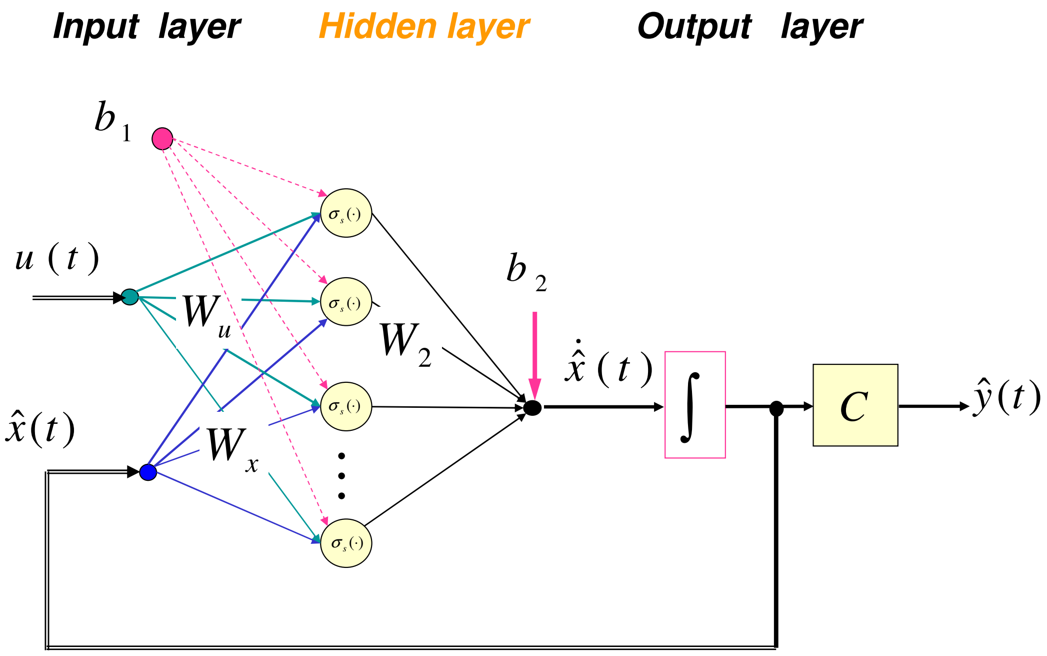

Figure 7.1: Continues–Time Recurrent Neural Network Structure

network is represented in a general nonlinear state–space configuration and used to predict the future dynamic behavior of the nonlinear process in realtime. An efficient training algorithm for the proposed network is developed using AD tool. The algorithm calculates the Taylor series coefficients required to solve the ODEs of the network and also produces the sensitivity for the training problem. The same approach is also used to solve the online optimization problem in the predictive controller.

7.2.1 Dynamic Recurrent Neural Networks

A dynamic recurrent neural network is a complex nonlinear dynamic system described by a set of nonlinear differential equations (continuous–time RNN) or difference equations (discrete–time RNN) with extensive connection weights. In this chapter, only continuous–time version of the dynamic RNNs is discussed.

It has been shown that CTRNNs are able to approximate trajectories generated by nonlinear dynamic systems given by:

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= g(x) \end{aligned} \tag{7.1}$$

A key to the approximating capability of this type of networks is the use of hidden neurons, [DKW95, GKW00]. The CTRNN is used to approximate the state–space model of the above nonlinear systems.

In the absence of any prior information, the nonlinear functions $f(\cdot)$ and $g(\cdot)$ in equation (7.4) must be left as general as possible. Since neural networks can approximate continuously differentiable nonlinear function arbitrarily well, it is a good choice for parameterizing f and g. Also the resulting continuous–time state-space have the neural network structure.

Hence, the CTRNN to be considered is represented in the following general form

$$\begin{aligned} \dot{\hat{x}}(t) &= \hat{f}(\hat{x}(t), u(t), \theta) \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{7.2}$$

where $u(t) \in \mathbb{R}^{n_u}$ is the external input, $\hat{y} \in \mathbb{R}^{n_y}$ the network output, $\hat{x} \in \mathbb{R}^{n_{\hat{x}}}$ the network's state vector, $\theta \in \mathbb{R}^{n_\theta}$ the network parameter vector and the output matrix C is fixed as

$$C = \left[I_{n_y \times n_y}, \mathbf{0}_{n_y \times (n_{\hat{x}} - n_y)} \right] \tag{7.3}$$

Hence the outputs are the first n_y states of the networks.

The nonlinear process is approximated using a continuous-time recurrent **M**ulti **L**ayer **P**erceptron (MLP) network with one hidden layer as shown in Figure 7.1:

$$\begin{aligned} x_h(t) &= \sigma_s \left(W_x \hat{x}(t) + W_u u(t) + b_1 \right) \\ \dot{\hat{x}}(t) &= W_2 x_h(t) + b_2 \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{7.4}$$

where, $W_x \in \mathbb{R}^{n_h \times n_{\hat{x}}}$, $W_u \in \mathbb{R}^{n_h \times n_u}$, and $W_2 \in \mathbb{R}^{n_{\hat{x}} \times n_h}$ are connection weights, $b_1 \in \mathbb{R}^{n_h}$ and $b_2 \in \mathbb{R}^{n_{\hat{x}}}$ are bias vectors, whilst each element of the vector $\sigma_s(\cdot) \in \mathbb{R}^{n_h}$ represents the sigmoid-*tanh* function as the neural activation function, *i.e.*

$$\sigma_s(n) = \frac{2}{1 + e^{-2n}} - 1$$

The parameter vector is $\theta = \begin{bmatrix}\mathrm{vec}(W_x)^T & \mathrm{vec}(W_u)^T & b_1^T & \mathrm{vec}(W_2)^T & b_2^T\end{bmatrix}^T \in \mathbb{R}^{n_\theta}$, where $n_\theta = n_{\hat{x}} \times (n_h + 1) + n_h \times (n_{\hat{x}} + n_u + 1)$.

7.2.2 CTRNN sensitivity calculation using AD

Re–define system (7.2) as the following form

$$\begin{aligned}\dot{\hat{x}}(t) &= \hat{f}(\hat{x}(t), \eta) \\ \hat{y}(t) &= C\hat{x}(t)\end{aligned} \tag{7.5}$$

where $\eta \in \mathbb{R}^{n_\eta}$ represents the general parameters vector, $\eta = \theta$ in network training problem, and $\eta = u$ in NMPC problem. The definition of the sensitivity is the variation of the network output against the variation of η. Assume the function $\hat{f}$ is d-time continuously differentiable. Then, the sensitivity can be calculated by taking partial derivative for both sides of equations (7.5):

$$\begin{aligned}\dot{x}_\eta(t) &= f_x x_\eta(t) + f_\eta \\ y_\eta(t) &= C x_\eta(t)\end{aligned} \tag{7.6}$$

where, $x_\eta := \partial\hat{x}/\partial\eta$, $y_\eta := \partial\hat{y}/\partial\eta$, $f_x := \partial\hat{f}/\partial\hat{x}$, and $f_\eta := \partial\hat{f}/\partial\eta$.

Equation (7.6) is a linear time–varying system with initial condition, $x_\eta(0) = \partial\hat{x}(0)/\partial\eta$. Generally, system (7.6) has no analytical solution although it can be represented in a state-transition matrix form [ChC98]. The dynamic sensitivity function x_η can be calculated using different method as mentioned earlier. Numerically, equation (7.6) can be solved together with the state equation (7.2) using a differential equation solver. The total number of differential variables to be solved at each time instant is $n_{\hat{x}} \times (1 + n_\eta)$. Depending on the size of a network, this number of differential variables could grow so large that the calculation causes a significant burden on network training. To tackle this problem, the sensitivity calculation method proposed in [Cao05] to do the optimization step in a NMPC is extended to find the optimal CTRNN parameters as well.

Using normalized time, $\tau = t/T_s$, the right–hand-side of the state equation becomes $z(\hat{x}(\tau), \eta) := T_s\hat{f}(\hat{x}(\tau), \eta)$ and the solution interval is $0 \le \tau \le 1$ for each integration step. Consider $\hat{x}(\tau)$ is given by the truncated Taylor series:

$$\hat{x}(\tau) = \hat{x}_{[0]} + \hat{x}_{[1]}\tau + \cdots + \hat{x}_{[d]}\tau^d \tag{7.7}$$

with coefficients $\hat{x}_{[i]} \in \mathbb{R}^{n_{\hat{x}}}$ given as follows respectively:

$$\hat{x}_{[i]} = (i!)^{-1} \left. \frac{\partial^i \hat{x}(\tau)}{\partial \tau^i} \right|_{\tau=0} \tag{7.8}$$

Then, $z(\tau)$ can be expressed by a Taylor expansion:

$$z(\tau) = z_{[0]} + z_{[1]}\tau + \cdots + z_{[d]}\tau_d + \mathcal{O}(\tau^{d+1}) \tag{7.9}$$

where coefficients $z_{[j]}$ are given as;

$$z_{[j]} = (j!)^{-1} \left. \frac{\partial^j z(\tau)}{\partial \tau^j} \right|_{\tau=0} \tag{7.10}$$

From the chain rule, $z_{[j]}$ is uniquely determined by the coefficient vectors, $\hat{x}_{[i]}$ and η with $i \leq j$, *i.e.*

$$z_{[j]} \equiv z_{[j]}(\hat{x}_{[0]}, \hat{x}_{[1]}, \cdots, \hat{x}_{[j]}, \eta) \tag{7.11}$$

Inherently, functions $z_{[j]}$ are also d-time continuously differentiable and their derivatives satisfy the identity [Chr92];

$$\begin{aligned} \frac{\partial z_{[j]}}{\partial \hat{x}_{[i]}} &= \frac{\partial z_{[j-i]}}{\partial \hat{x}_{[0]}} = A_{x[j-i]} \equiv A_{x[j-i]}(\hat{x}_{[0]}, \hat{x}_{[1]}, \cdots, \hat{x}_{[j-i]}, \eta) \\ \frac{\partial z_{[j]}}{\partial \eta} &= A_{\eta[j]} \equiv A_{\eta[j]}(\hat{x}_{[0]}, \hat{x}_{[1]}, \cdots, \hat{x}_{[j-i]}, \eta) \end{aligned} \tag{7.12}$$

(7.13)

where, $A_{x[j]} \in \mathbb{R}^{n_{\hat{x}} \times n_{\hat{x}}}, j = 0, \cdots, d$, and $A_{\eta[j]} \in \mathbb{R}^{n_{\hat{x}} \times n_{\eta}}$, $j = 0, \cdots, d$ are also the Taylor coefficients of the Jacobian path, *i.e.*;

$$\frac{\partial z}{\partial \hat{x}_{[0]}} = A_{x[0]} + A_{x[1]}\tau + \cdots + A_{x[d]}\tau^d + \mathcal{O}\tau^{d+1} \tag{7.14}$$

$$\frac{\partial z}{\partial \eta} = A_{\eta[0]} + A_{\eta[1]}\tau + \cdots + A_{\eta[d]}\tau^d + \mathcal{O}\tau^{d+1} \tag{7.15}$$

AD techniques provide an efficient way to calculate these coefficients vectors, $z_{[j]}$ and matrices $A_{[i]}$ [Gri00]. For example, with the software package, ADOL-C [GDJ96, Gri95], using the forward mode of AD all Taylor coefficient vectors for a given degree, d can be calculated simultaneously, whilst the matrices, $A_{[i]}$ can be obtained using the reverse mode of AD. The run time and memory requirement associated with these calculations grow only as d^2.

Using AD for the CTRNN system (7.5), the Taylor coefficients of $\hat{x}(\tau)$ can be iteratively determined from $\hat{x}_{[0]}$ and η:

$$\begin{aligned} \hat{x}_{[k+1]} &= \frac{1}{k+1} z_{[k]}(\hat{x}_{[0]}, \cdots, \hat{x}_{[k]}, \eta), \quad k = 0, \cdots, d-1 && (7.16) \\ \hat{y}_{[k]} &= C\hat{x}_{[k]}, \quad k = 0, \cdots, d && (7.17) \end{aligned}$$

Then, by applying AD to (7.16), the partial derivatives are obtained and partitioned as;

$$A_{[k]} = \left[A_{x[k]} \;\; | \;\; A_{\eta[k]} \right] := \left[\tfrac{\partial z_{[k]}}{\partial \hat{x}_{[0]}} \;\; | \;\; \tfrac{\partial z_{[k]}}{\partial \eta} \right], \qquad (7.18)$$

The total derivatives are accumulated from these partial derivatives as follows:

$$B_{[k+1]} = \left[B_{x[k+1]} \;\; | \;\; B_{\eta[k+1]} \right]$$

where

$$\begin{aligned} B_{x[k+1]} &:= \frac{dx_{[k+1]}}{dx_{[0]}} = \frac{1}{k+1} \sum_{j=0}^{k} \frac{\partial z_{[k]}}{\partial x_{[j]}} \cdot \frac{dx_{[j]}}{dx_{[0]}} = \frac{1}{k+1} \sum_{j=0}^{k} A_{x[k-j]} B_{x[j]} && (7.19) \\ B_{\eta[k+1]} &:= \frac{dx_{[k+1]}}{d\eta} = \frac{1}{k+1} \left(\frac{\partial z_{[k]}}{\partial \eta} + \sum_{j=0}^{k} A_{x[k-j]} \cdot B_{\eta[j]} \right), \quad k = 0, \cdots, d-1 && (7.20) \end{aligned}$$

and

$$B_{[0]} = \left[B_{x[0]} \;\; | \;\; B_{\eta[0]} \right] = \left[\; I \;\; | \;\; B_{\eta[0]} \right] \qquad (7.21)$$

where $B_{\eta[0]} := d\hat{x}_{[0]}/d\eta$. Note that, in the NLP of the controller $B_{\eta_{[0]}} = 0$ because $\hat{x}_{[0]}$ is independent on the initial value of u.

$$\hat{x}(T_s) = \sum_{i=0}^{d} \hat{x}_{[i]}, \qquad \hat{y}(T_s) = C\hat{x}(T_s) \qquad (7.22)$$

whilst their sensitivities to initial value, $\hat{x}_{[0]}$ and coefficients η are,

$$\begin{aligned} B_x(T_s) &:= \frac{d\hat{x}(T_s)}{d\hat{x}_{[0]}} = \sum_{i=0}^{d} B_{x[i]} = I + \sum_{i=1}^{d} B_{x[i]} && (7.23) \\ B_\eta(T_s) &:= \frac{d\hat{x}(T_s)}{d\eta} = \sum_{i=0}^{d} B_{\eta[i]} = B_{\eta[0]} + \sum_{i=1}^{d} B_{\eta[i]} && (7.24) \\ D_x(T_s) &:= \frac{d\hat{y}(T_s)}{d\hat{x}_{[0]}} = CB_x(T_s) && (7.25) \\ D_\eta(T_s) &:= \frac{d\hat{y}(T_s)}{d\eta} = CB_\eta(T_s) && (7.26) \end{aligned}$$

7.2.3 Network training algorithm

Training produces the optimal connection weights for the networks by minimizing a quadratic cost function of the errors between the neural network output and the plant output over the entire set of samples. Among many network training algorithms, Levenberge-Marquardt (LM) algorithm [Mar63] is known to be a robust and fast gradient method because of its second-order converging speed without having to compute the Hessian matrix. For this reason, the LM algorithm is combined with the sensitivity algorithm using AD described above for the dynamic network training.

Assuming the dynamic system (7.1) is initially at steady-state and introducing a set of random inputs to the system, the outputs of the plant are then collected with the inputs for N sampling points at sampling rate T_s. The unknown network parameters θ are estimated from the input-output data set by minimizing the sum of squared approximation errors, i.e.

$$\min_{\theta} \ \Phi = \min_{\theta} \ \frac{1}{2} \sum_{i=0}^{N} e_i^T e_i \tag{7.27}$$

where, e_i is the error between the actual plant output and the network output at i-th sampling point which is a function of the model parameter vector given by:

$$e_i \equiv e_i(\theta) = \hat{y}(t_i, \theta) - y(t_i), \quad i = 1, 2, \cdots, N \tag{7.28}$$

Let:

$$E(\theta) = \begin{bmatrix} e_1^T & \cdots & e_N^T \end{bmatrix}^T \tag{7.29}$$

The $n_y N \times n_\theta$ Jacobian matrix of E is defined as

$$J(\theta) := \frac{\partial E(\theta)}{\partial \theta} \tag{7.30}$$

Then, the gradient of Φ is $J(\theta)E(\theta)$, whilst the Hessian of Φ can be approximated as $J^T(\theta)J(\theta)$. The training algorithm based on the nonlinear least square approach of Levenberg–Marquandt [Mar63] is:

$$\theta_{k+1} = \theta_k - \left[J(\theta_k)^T J(\theta_k) + \mu I \right]^{-1} J(\theta_k)^T E(\theta_k) \tag{7.31}$$

where, θ_{k+1} is an updated vector of weights and biases, θ_k the current weights and biases, and I the identity matrix. When the scalar μ is zero, this is a quasi-Newton approach, using the approximate Hessian matrix, $J^T J$. When μ is large, it is equivalent to a gradient descent method with a small step size. Quasi-Newton method is faster and more efficient when Φ is near the error minimum. In this way, the performance function Φ will always be reduced at each iteration of the algorithm.

The Jacobian matrix can be partitioned into N blokes as:

$$J(\theta) = [J_1^T(\theta) \cdots J_N^T(\theta)]^T \tag{7.32}$$

where each block is an $n_y \times n_\theta$ matrix as:

$$J_i(\theta) = \frac{\partial e_i(\theta)}{\partial \theta} = \frac{\partial \hat{y}(t_i, \theta)}{\partial \theta} = C \frac{\partial \hat{x}(t_i, \theta)}{\partial \theta} \tag{7.33}$$

For accurate and fast calculation of the sensitivity equations required for the Jacobian matrix above, the method described in the previous section is adopted here. Note that, η is replaced by the network parameters vector θ in this case.

For given $\eta_{[0]}$, $\hat{x}(k+1) := \hat{x}(t_{k+1})$, and $\hat{y}(k) := \hat{y}(t_k)$ are iteratively determined from $\hat{x}(0) = [y^T(0), \mathbf{0}_{1\times(n_{\hat{x}}-n_y)}]^T$ using (7.22). Then the value of $J_k(\theta) = d\hat{y}(k)/d\eta_{[0]}$ can be calculated using (7.19) and (7.23) – (7.26) as:

$$\begin{aligned} B_\theta(0) &= 0 = B_{\theta[0]}(0) && (7.34)\\ B_x(k) &= I + \sum_{i=1}^{d} B_{x[i]}(k-1) && (7.35) \end{aligned}$$

$$\begin{aligned} B_\theta(k) &= B_\theta(k-1) + \sum_{i=1}^{d} B_{\theta[i]}(k-1) = B_{\theta[0]}(k) && (7.36)\\ D_\theta(k) &= CB_\theta(k) = J_k(\theta) && (7.37) \end{aligned}$$

Hence, with AD, the nonlinear training problem can be efficiently solved using the NLSQ method.

7.2.4 Model Validation

Many model validity tests for nonlinear models have been developed [ZM99], for example, the Akaike information criterion (AIC), the statistical χ^2 tests, the predicted squared error criterion, and the higher–order correlation tests.

The most common method of validation is to investigate the residual (prediction errors) by cross validation on a *test data set*. A number of such tests, including autocorrelation function of the residual and cross–correlation function between controls and residuals. If the identified model based on CTRNN is adequate, the prediction errors should satisfy the following conditions of high–order correlation tests [BVS86]:

$$\begin{aligned} R_{ee}(\tau) &= E[e(t-\tau)e(t)] = \delta(\tau), \quad \forall\tau & (7.38)\\ R_{ue}(\tau) &= E[u(t-\tau)e(t)] = 0, \quad \forall\tau & (7.39)\end{aligned}$$

where $R_{xz}(\tau)$ indicate the cross-correlation function between $x(t)$ and $z(t)$, e is the model residual. These tests look into the cross-correlation amongst model residuals and inputs. Normalization to give all tests a range of ±1 approximate the 95% confidence bounds, makes the tests independent of signal amplitude and easy to interpret [BVS86]. If these correlation tests are satisfied then the model residuals are a random sequence and are not predictable from the model inputs. This provides additional evidence of the validity of the identified model.

7.3 NMPC Algorithm using Taylor Series

Once the CTRNN has been trained, the network can be used as an internal model of a predictive controller. The recurrent neural network generates prediction of future process outputs over a specified prediction horizon P, which allows the performance criterion of the predictive controller to be minimized. Note that the same objective function and variables constraints of the NMPC algorithm presented in chapter 4 is used here, except that, the ODEs of system (4.2) is replaced by the steady–state CTRNN equations (7.2), and the state variable x is replaced by the network state variable $\hat{x}$ as follows;

$$\begin{aligned} \min_{\underline{U}\leq U\leq\overline{U}} \quad V(U) &= \frac{1}{2}E^T(U)E(U) & (7.40)\\ \text{s.t.}\ \dot{\hat{x}}(t) &= \hat{f}(\hat{x}(t),u(t)), \quad t\in[t_0,t_P] & (7.41)\\ \hat{y}(t) &= C\hat{x}(t)+d_{o,k} & (7.42)\\ \hat{x}(t_0) &= \hat{x}(0), \quad \hat{x}(k) := \hat{x}(t_0+kT_s) & \\ E(U) &= \left[e(1)^{QT} \quad \cdots \quad e(P)^{QT},\ \Delta u(1)^{ST}\cdots\Delta u(M)^{ST}\right]^T & (7.43)\\ e^Q(k) &:= Q_1(k)(\hat{y}(k)-r(k)), \quad k\in[1,P] & (7.44)\\ \Delta u(k)^S &:= S_1(k)^{1/2}\Delta u(k) & (7.45)\end{aligned}$$

where U, $\underline{U}$, $\overline{U}$, Δu, Q, S, Q_1, S_1, and S are defined in chapter 4, and $d_o(k+i) = d_o(k) = d_{o,k}$ for $i = 1,\cdots,P$, is an output integrated disturbance vector (proposed in chapter 6) used here for offset free control. The prediction horizon $[t_0, t_P]$ is divided into P intervals, $t_0, t_1, \cdots, t_P$ with $t_{i+1} = t_i + T_{s,i}$ and $\sum_{i=0}^{P-1} T_{s,i} = t_P - t_0$. For piecewise constant control, assume the optimal solution to (7.40) is $u(t) \equiv u(t_k) = u(k)$ for $t_k \leq t \leq t_{k+1}$, $k = 0, \cdots, P-1$. Then, only the solution in the first interval is to be implemented and whole procedure will be repeated at next sampling instant.

7.3.1 Jacobian Matrix Calculation

To efficiently solve the optimal control problem (7.40), the same strategy descried in section 7.2.2, is used here.

Since the independent variables in this case are the control signals, the variable η in Eqs. (7.5)–(7.26) will be replaced by u.

Let $u(k)$, be input coefficient at $t = t_k$, and $U \in \mathbb{R}^{n_u \times P}$ be;

$$U := [u^T(0) \cdots u^T(P-1)]^T \tag{7.46}$$

For given $u(k)$, $\hat{x}(k+1) := \hat{x}(t_{k+1})$ and $\hat{y}(k) := y(t_k)$ are iteratively determined from $\hat{x}(k)$ using (7.16). Hence (7.40) can be represented in discrete form;

$$\begin{aligned} \min_{\underline{U} \leq U \leq \overline{U}} V(U) &= \frac{1}{2} E^T(U) E(U) && (7.47) \\ \text{s.t. } \dot{\hat{x}}(k+1) &= \hat{f}(\hat{x}(k), u(k)), \quad \hat{x}(0) = \hat{x}_{[0]} && (7.48) \\ \hat{y}(k) &= C\hat{x}(k), \quad k \in [0, P-1] && (7.49) \end{aligned}$$

Problem (7.47) is a standard NLP problem which can be solved by any modern NLP solvers with $P \times n_u$ degrees of freedom. In this work the nonlinear least square method is used. The Jacobian matrix (*i.e.* the first order derivatives of E against U) can be obtained using (7.25) and (7.26) repeatedly. More specifically, define $\frac{d\hat{y}(k)}{dU} = \left[\frac{d\hat{y}(k)}{du(0)} \cdots \frac{d\hat{y}(k)}{du(P-1)}\right]$. Then,

$$\frac{dy(k)}{du(j)} = \begin{cases} 0 & k \leq j \\ D_u(j+1) & k = j+1 \\ D_x(k) B_x(k-1) \cdots B_x(j+2) B_u(j+1) & k > j+1 \end{cases} \tag{7.50}$$

where, B_u and D_u matrices have the same definition as the B_η and D_η matrices in equations (7.24) and (7.26) respectively, after replacing the term η by u.

For MPC with moving horizon, $M < P$, *i.e.* $u(k) = u(M-1)$, $k = M, \cdots, P-1$, the derivatives against u_{M-1} is a summation of derivatives against u_k, $k = M-1, \cdots, P-1$, *i.e.* $d/du(M-1) = \Sigma_{k=M-1}^{P-1} d/du(k)$.

Hence, using AD the nonlinear model predictive control problem can be efficiently solved by any modern NLP software. In this work NLSQ method is used to solve the optimization problem while ADOL–C is the AD package in C++ langauge.

7.4 Example 1: Evaporator Process

7.4.1 System Identification Using CTRNN

Many researchers, [KGW00, GKW00] have studied the evaporator system to demonstrate the modelling capability of the CTRNN in the form of the non–autonomous system $\dot{x} = f(x) + Bu$. In this section, the evaporator is approximated using the proposed CTRNN as the system (7.4). The identification scheme assumes that the plant model equations are unknown and the only available information is the input-output data which is generated through various runs of the first–principle model of the plant (A.1) by [NL89]. Two different structures of the CTRNN are studied to model the process. The first network (*Net1*) was trained with $n_{\hat{x}} = n_y = 3$, and $n_h = 8$ ($n_\theta = 83$), while the second one (*Net2*) was trained with $n_{\hat{x}} = 5$ and $n_h = 8$ ($n_\theta = 117$). The training was carried out repetitively over the data collected within a fixed time interval of 500 minutes and sampled at every 0.2 minutes. The inputs training data was random pulses with a different amplitude and durations with the range chosen to cover all the region of operation of the plant and the expected length of the prediction horizon (see Figure 7.2). Another set of data at sampling time 0.05 minutes is randomly generated from the plant to be used for network validation. The output data are corrupted with a normally distributed zero mean noise with variance 5% of the steady state values of the output variables. The initial values of the first n_y network states were chosen equal to the steady state values of the plant outputs while the $(n_{\hat{x}} - n_y)$ remaining states were set equal to zero. The three sets of data were normalized within ±1 to ensure that the data fall in the range of the nonlinear *tanh-sigmoid* activation function of the hidden neurons.

To examine the CTRNN capability for evaporator model approximation, the actual plant output and the trained neural networks output are compared in Figure 7.3 and Figure 7.4. A good model fitting is observed for both networks with approximately similar accuracy with the training data. In terms of model validation, *Net1* is better than *Net2* as shown in Figure 7.4. This means that increasing the network states dimension does not necessarily improve the model fitting. Sometimes networks with high order could include undesirable eigen values which may induce unstable or poor performance. *Net1* is chosen as a result as the internal model for the predictive controller for accurate and fast online calculations. Also, the validation results show the capability of the network to approximate the plant output with a sampling time less than that used in the training, without the needed to re-train the network. In fact, this is one of the most important advantages of CTRNNs over discrete-time recurrent networks.

Also, as a confidence test of the resulting model, the correlation–based model validation results for the CTRNN model can be calculated according to equations (7.38) and shown in Figure 7.5. The dotted lines in each plot are the 95% confidence bounds $(\pm 1.96/\sqrt{500})$. It can be seen that only a small number of points are outside the bounds. This demonstrates that the model can be considered as being adequate for modelling this plant.

To solve the training problem, a total $n_{\hat{x}} \times N \times n_{\theta} = 3 \times 2500 \times 83 = 622500$ sensitivity variables need to be calculated in addition to the original 3 ordinary differential equations of *Net1*. For *Net2*, the number of sensitivity is $5 \times 2500 \times 117 = 1462500$. To demonstrate the efficiency of the new algorithm, it is compared with the traditional sensitivity equation integrating approach using a typical numerical ODE solver, e.g. the MATLAB function *ode23*.

To compare the computation time associated with a given accuracy, a reference solution is produced using *ode23* solver and setting the error tolerance to 10^{-14}. Then with four tolerance settings, (10^{-3}, 10^{-6}, 10^{-8}, 10^{-10}) and four different Taylor series orders (3, 6, 8, 10), computation time and accuracy against the reference solutions using two different approaches are compared in Table 7.1. Note that the computation time in Table 7.1 is the time required to calculate the cost function and the sensitivity variables over one optimization iteration. While the error term in the same table is the maximum absolute error against the reference solution. The table shows that training algorithm using AD perform better than the traditional sensitivity approach in both efficiency and accuracy. It can be seen that the order of Taylor series plays an important role in error control. Increasing the order by a small number, the error would be reduced by an order of magnitude with only small increase in the computation time. However, using traditional approaches, significant computation time may have to be traded off for a reduction in computation error.

Table 7.1: Computing Time and Accuracy Comparison/Evaporator

Traditional Sensitivity Approach

Tolerance	*Net1*: ($n_{\hat{x}} = 3$)		*Net2*: ($n_{\hat{x}} = 5$)	
	Time, sec	Error	Time, sec	Error
1e-3	13.859	0.555	48.328	4.7175
1e-6	45.046	0.0153	257.454	0.1351
1e-8	69.437	4.0183e-4	434.547	8.973e-4
1e-10	77.906	1.1103e-8	580.125	1.3316e-5

ADOL–C Software

Actual Order	$Net1$: ($n_{\hat{x}} = 3$)		$Net2$: ($n_{\hat{x}} = 5$)	
	Time, sec	Error	Time, sec	Error
3	2.609	3.276e-5	3.11	1.396e-4
6	4.031	2.095e-10	5.281	1.7862e-9
8	5.207	1.136e-13	6.875	1.2301e-12
10	6.813	8.881e-16	8.875	5.6843e-14

All tests are done on a Windows XP PC with an Inetl Pentium-4 processor running at 3.0 GHz. The proposed algorithm is implemented in C using ADOL-C and interfaced to MATLAB via a *mex* wrap.

7.4.2 Evaporator Control Using NMPC

The NMPC algorithm descried in section 7.3 and the new offset removal approach using output disturbance adaptation method is used to control the evaporator system. The design parameters chosen after an online tuning are as follows: sampling time $T_s = 1$ minutes, prediction horizon $P = 7$ time steps (7 minutes), and control horizon $M = 4$ time steps (4 minutes).

The control signal change weighting matrix S is set to I and the output weighing matrix Q is initially set to be the inverse of the output error bounds. After online tuning, the final values is $Q = diag(1000, 500, 200)$, and the d_o model constants are chosen as $K_1 = diag(0.2, 0.5, 0.1)$, $K_2 = diag(1, 0.5, 1)$, and $e_{ss,max} = -e_{ss,min}$ =[0.5 1 1].

By using piecewise constant input, the result NLP problem has $n_u \times M = 12$ degrees

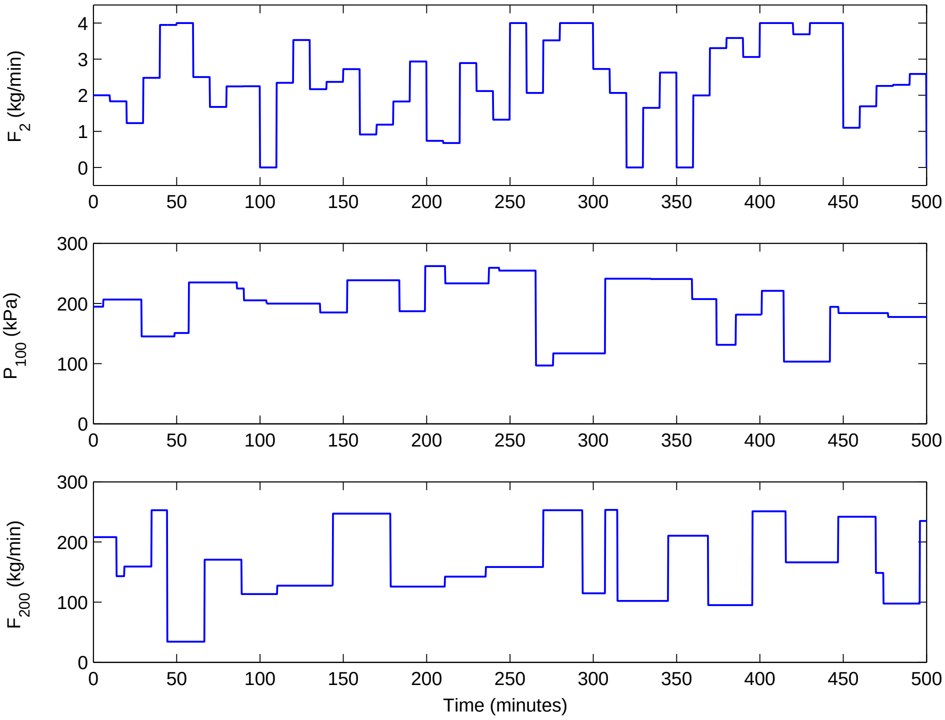

Figure 7.2: Training data set, Inputs, $\triangle t = 0.5$ min.

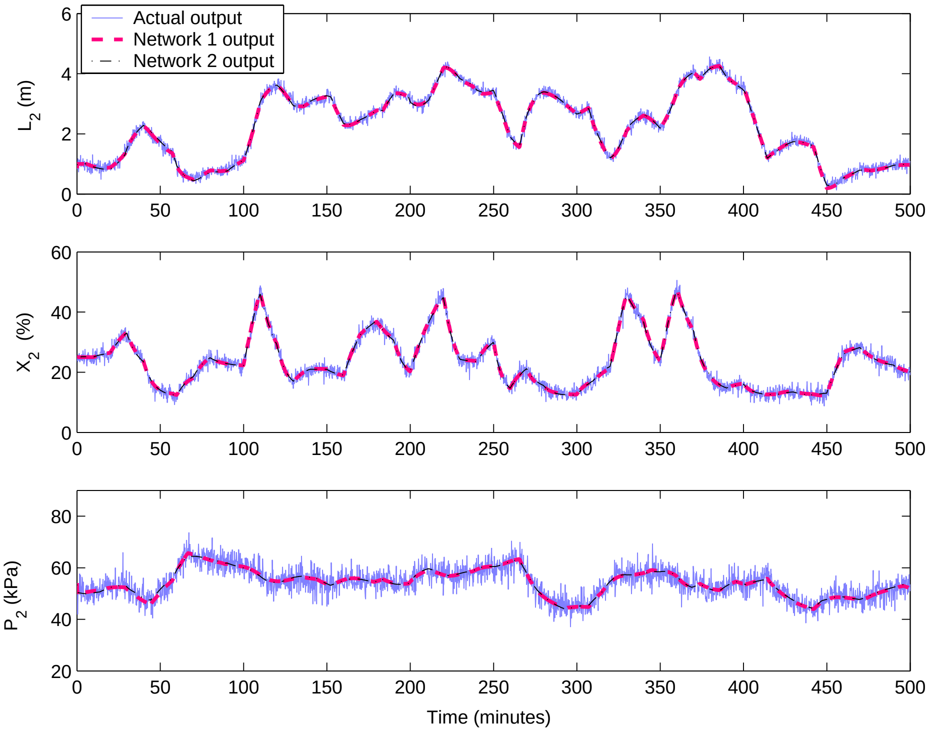

Figure 7.3: Training data set, Outputs, $\triangle t = 0.2$ min.

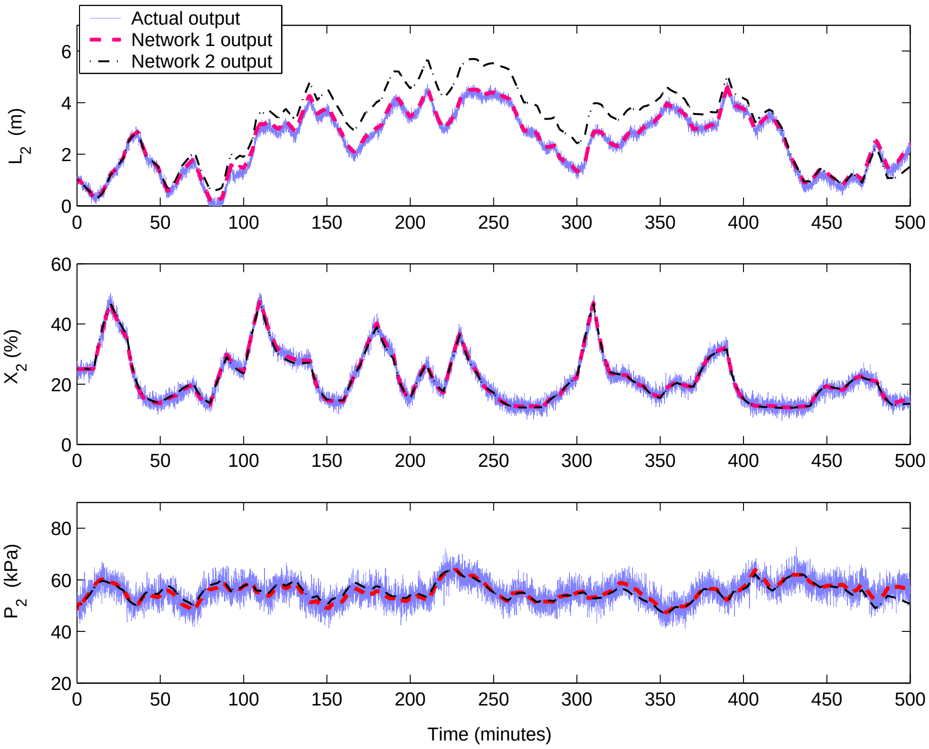

Figure 7.4: Validation data set, Outputs, $\triangle t = 0.05$ min.

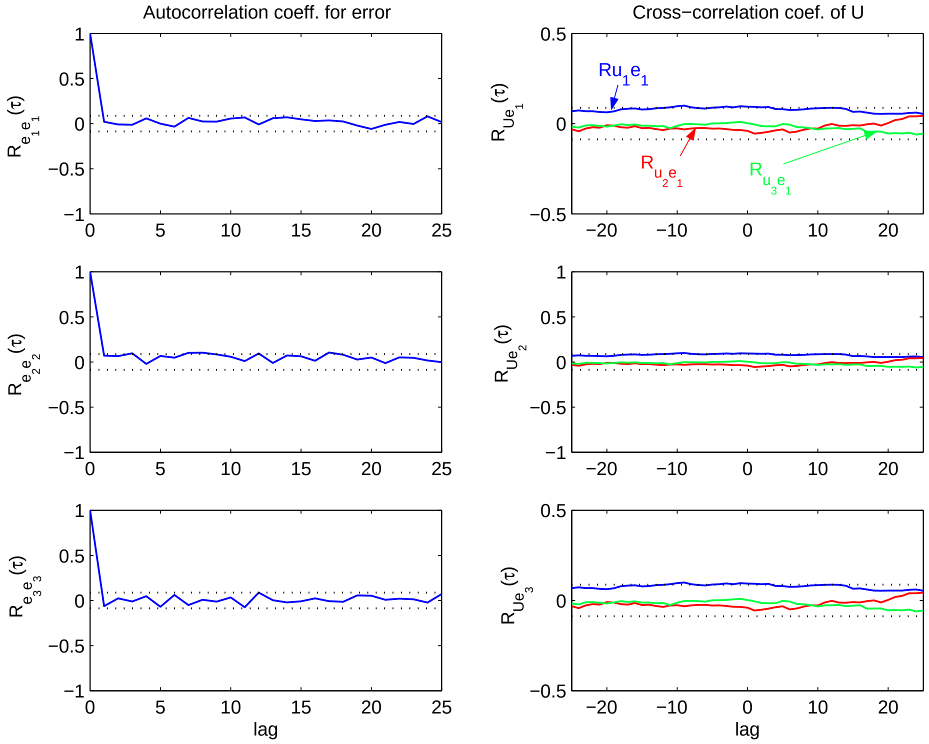

Figure 7.5: Evaporator model validation tests (validation data set)

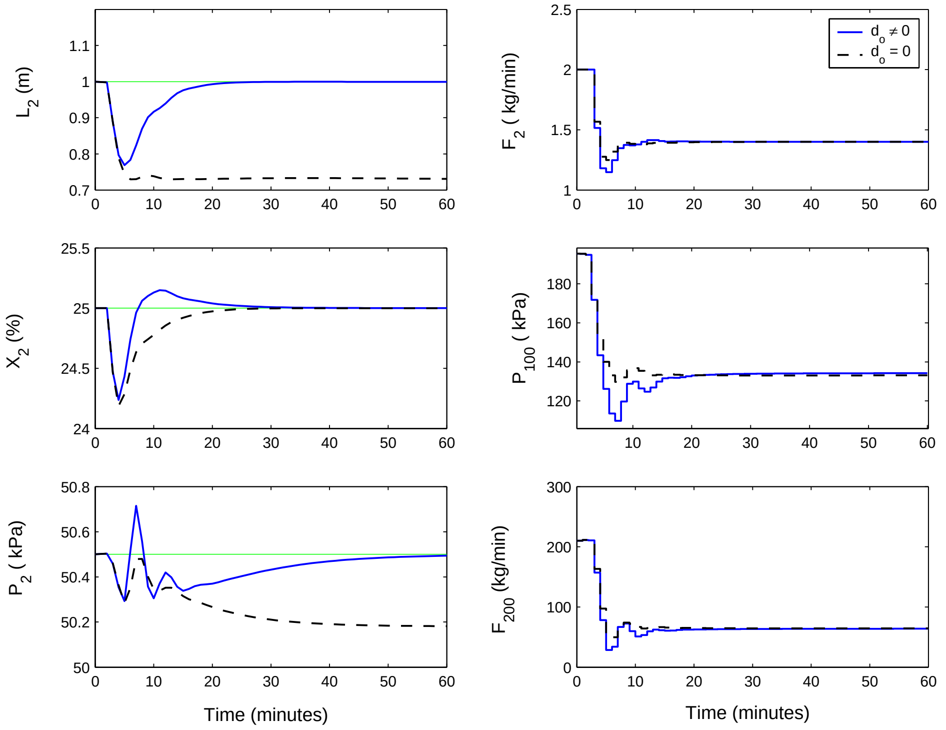

Figure 7.6: Evaporator performance during unmeasured disturbance as -30% step change in F_1 at $t = 2$ minutes using NMPC based on CTRNN, $\triangle t = 1$ min.

of freedom. To solve the NLP problem of the NMPC, a total $(n_{\hat{x}} \times P) \times (n_u \times M) =$ 252 sensitivity variables have to be calculated in addition to original 3 ODEs of the neural network. In this work, the sensitivity equations are solved using the sensitivity algorithm of section 7.3.

The control system is subject to the following tests. In the first test, an unmeasured non–zero mean disturbance of a -30% step change in the nominal value of the flowrate F_1 is applied to the process at $t = 2$ minutes. Figure 7.6 show a comparison in the system performance using NMPC with output integrated disturbance d_o and without. The results indicate the importance of using the disturbance d_o to enhance the predictive controller robustness in the presence of model/plant mismatch.

In a setpoint tracking test, ramp changes of X_2 from 25% to 15% and P_2 from 50.5 kPa to 70 kPa are assumed. Then, at $t = 60$ minutes, an unmeasured disturbance, a step change in F_1 about -30% of its nominal value is injected to the process to test the disturbance rejection performance of the NMPC controller. Simulation results of the setpoint tracking plus unmeasured disturbance rejection test are shown in Figure 7.7. The results show that the NMPC managed to force the measured outputs to follow the setpoints quite well without any offset error or input constraints violation.

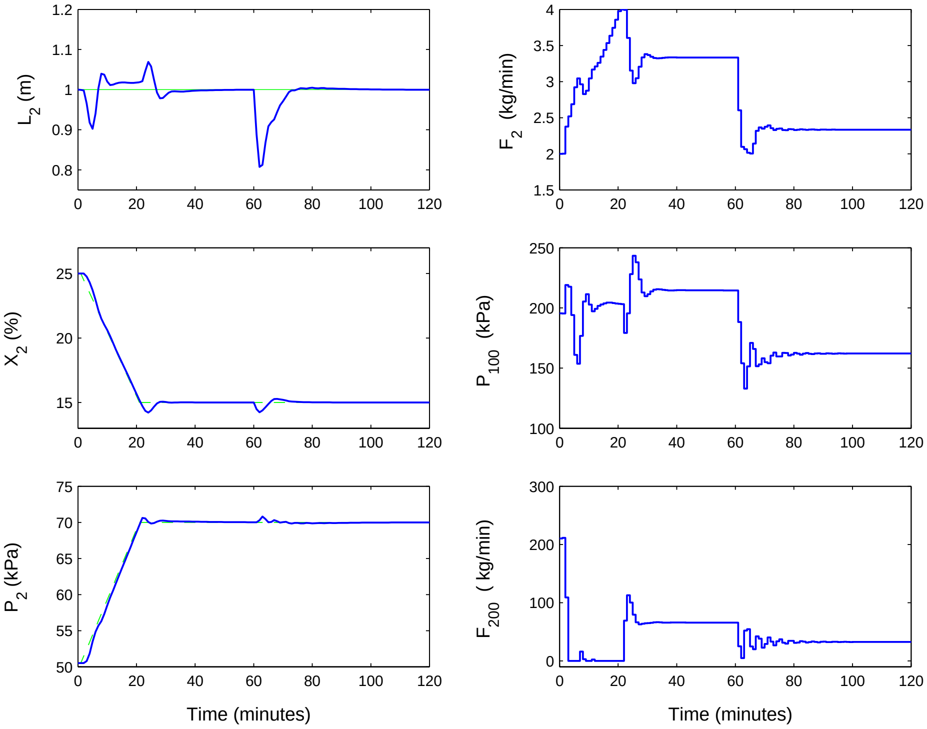

Figure 7.7: Evaporator performance during ramp setpoint tracking, and unmeasured step disturbance in $F1$ at $t = 60$ min. using NMPC based on CTRNN, $\triangle t = 1$ min.

To demonstrate the efficiency of the new NMPC algorithm using AD tool the following cases are compared; C1 (for first–order approximation of sensitivity equations with AD), and C2 (for sensitivity calculations using Taylor series expansion with AD), the computational time is compared with two other NMPC controllers (Table 7.2); C3 using MATLAB *ode23* solver plus a perturbation to get the sensitivity and C4 using AD to solve the differential equations but with perturbation to get sensitivity. Table 7.2 shows that the differential equation solver using AD reduced computational time (*i.e.* the total simulation time for the setpoint changes test shown in Figure 4.4) approximately by an order of magnitude (comparing C2 with C1 and C3 with C4) and $(n_yP \times n_uM)$ sensitivity variables calculation using AD saves another order of magnitude in time (comparing C2 with C4 and C2 with C3). For differentiation using the perturbation approaches, the computational time is very sensitive to the number of independent variables (C3 and C4) whilst for AD approaches, it is insensitive in this case study (C1 and C2).

Table 7.2: Total Simulation Time (sec) Comparison

Controller Parameters	C1	C2	C3	C4
$P = 5, \quad M = 2$	16.719	**4.125**	48.64	8.203
$P = 10, \quad M = 5$	18.875	**4.3910**	115.86	13.328
$P = 50, \quad M = 10$	42.0	**17.484**	395.891	129.968

7.5 Example 2: Two CSTR Process

To test the proposed CTRNN generality to approximate nonlinear process models for NMPC schemas, the two–CSTR/S2 has been chosen as an additional application.

7.5.1 System Identification Using CTRNN

The identification scheme assumes that the plant model equations are unknown and the only available information is the input-output data which is generated through various runs of the first principle–model of the plant (B.1) descried in [HPB94]. The CTRNN is trained with $n_h = \ 6$, $n_{\hat{x}} = 6$. The training was carried out repetitively over the fixed time interval $[0, 600]$ seconds and sampling time equal 0.1 seconds to minimize the performance index (7.27). The training data sets are given in Figures 7.8 to 7.9. The initial values of the first two states of the network were chosen equal to the nominal values of the two tanks output temperature (= 362.995 K^o), while the other four states were set equal to zero. Also, a measurement noise with a normally distributed, zero mean, and variance 2.5% of the steady state values of the output variables was added to the plant outputs. The network capability to approximate the two–CSTR dynamic response were demonstrated by the validation results shown in Figures 7.9 to 7.11. Note that the sampling time of the first validation data set is equal to the training data sampling time (i.e. 0.1 second), while the second data set was sampled at 0.02 second to demonstrate the network approximation capability at different sampling rates.

The correlation–based model validation results for the two–CSTR model were calculated according to equations (7.38) and shown in Figures 7.12 to 7.13. The dash–lines in each plot are the 95% confidence bounds ($\pm 1.96/\sqrt{600}$). It can be seen that only a small number of points are slightly outside the given bounds. This demonstrates that the model can be considered as being adequate for modelling this plant.

In the network training step, the number of the sensitivity variables equal to $n_{\hat{x}} \times N \times n_\theta = 6 \times 6000 \times 96 = 3456000$. The proposed training algorithm is used to train the

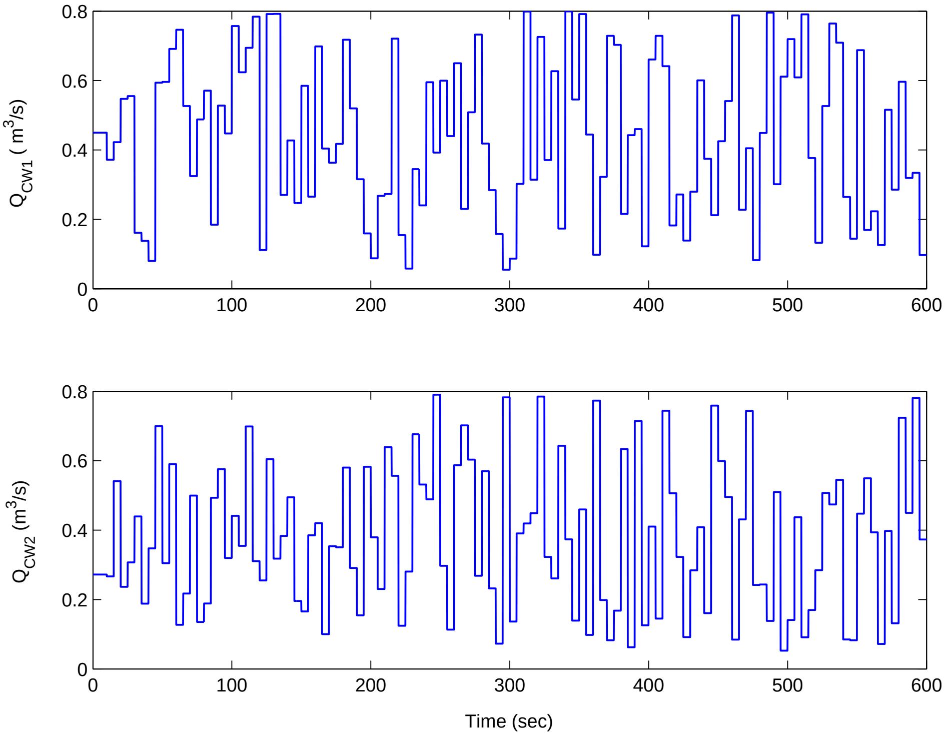

Figure 7.8: Training data set, Inputs, $\triangle t = 0.1$ sec.

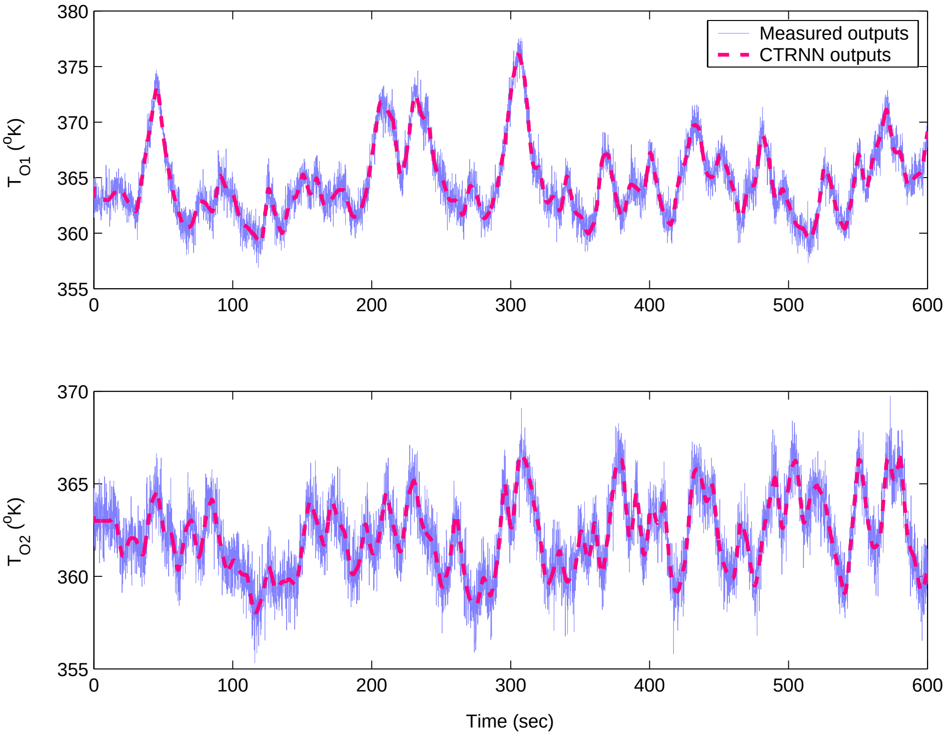

Figure 7.9: Training data set, Outputs, $\triangle t = 0.1$ sec.

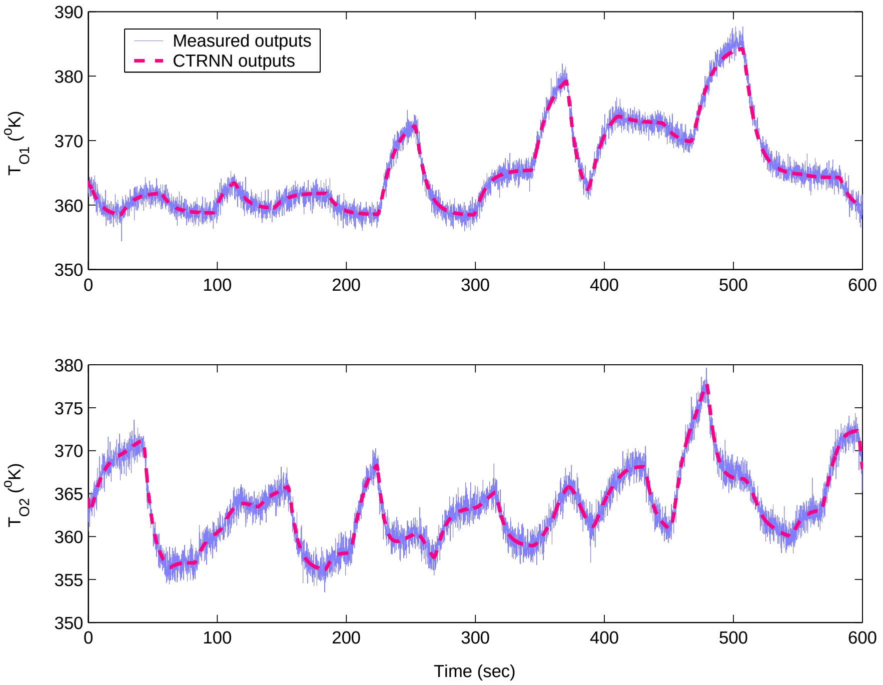

Figure 7.10: Validation data set, Outputs, $\triangle t = 0.1$ sec.

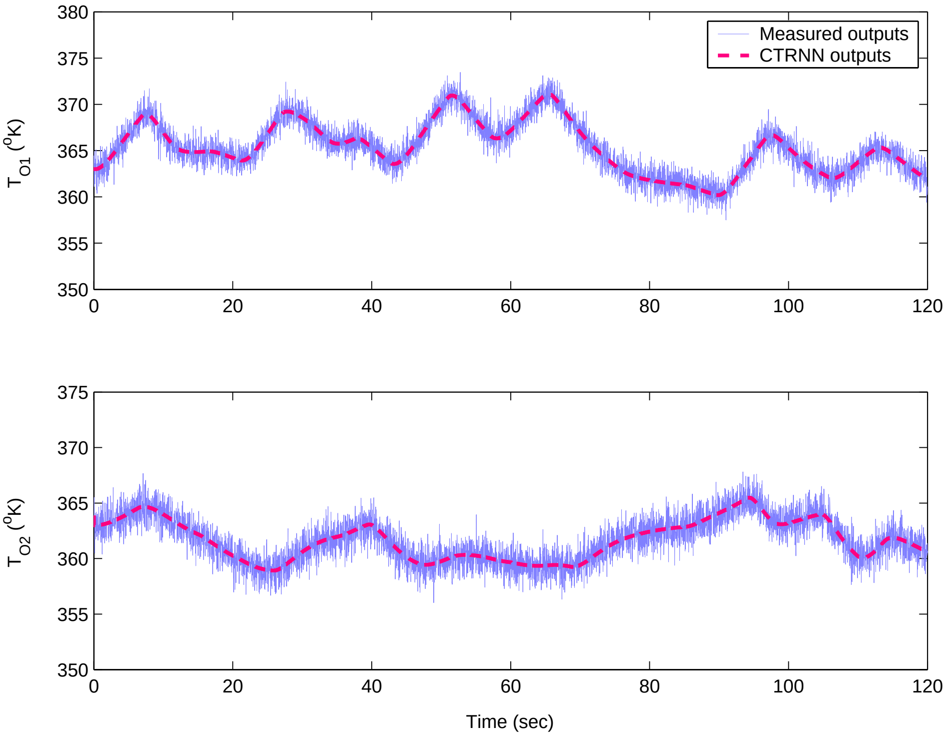

Figure 7.11: Validation data set, Outputs, $\triangle t = 0.02$ sec.

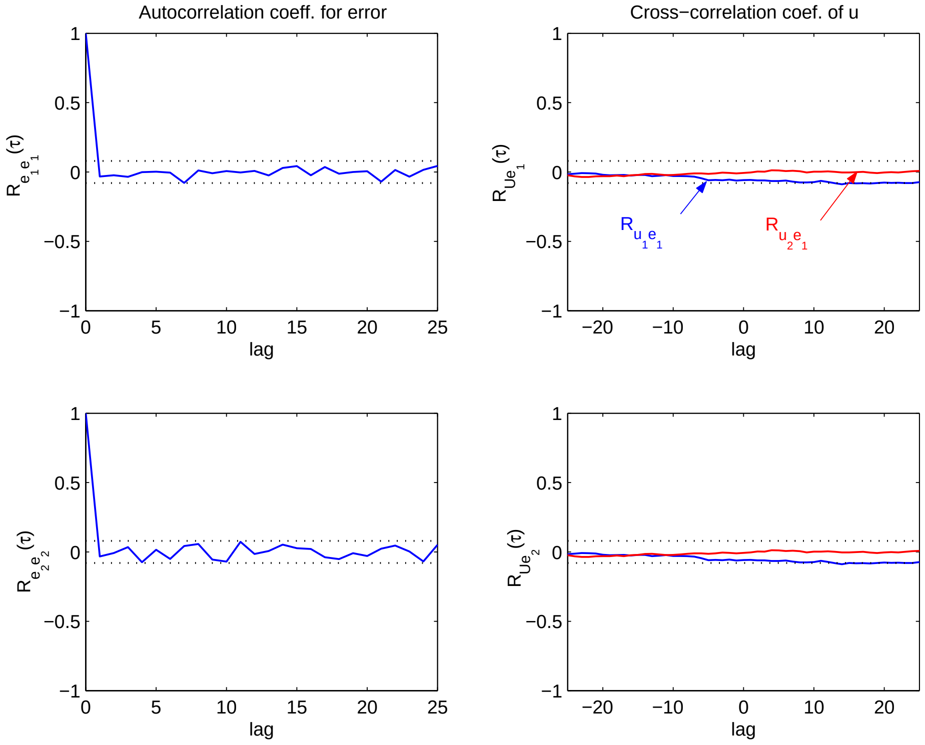

Figure 7.12: Two–CSTR model validation tests (validation data set), $\triangle t = 0.1$ sec.

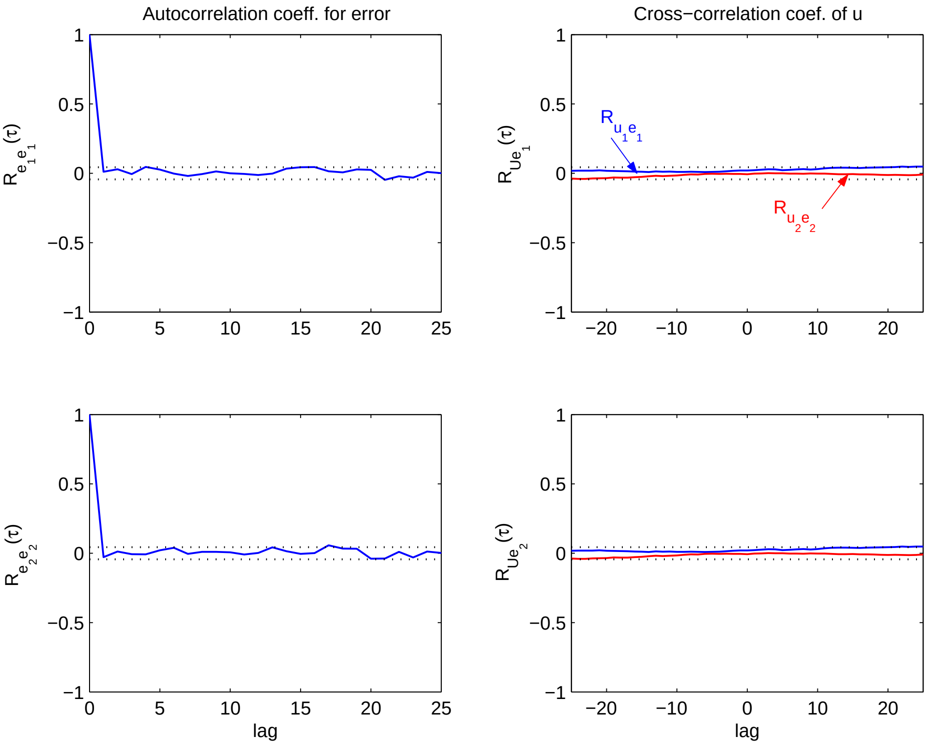

Figure 7.13: Two–CSTR/S2 model validation tests (validation data set), $\triangle t = 0.02$ sec.

network and its efficiency is compared with the traditional sensitivity approach (*i.e.* *ode23* MATLAB solver) as given in Table 7.3. The results show a big time saving with a high accuracy level using the new training algorithm.

7.5.2 Controller Design

The control problem is to maintain both tank temperatures T_{o1} and T_{o2} at the desired values in the presence of $\pm 10K^o$ cooling-water temperature fluctuations (T_{CW1} and T_{CW2}). Also, the controller should be able to control the process during a setpoint change. The two tests should be performed in the presence of actuator constraints which are; $0.05 \leq Q_{CW1} \leq 0.8$, and $0.05 \leq Q_{CW2} \leq 0.8$ respectively. The NMPC design parameters after online tuning have been chosen as follows; sampling time $T_s =$ 0.1 sec, $P = 5$ time steps , $M = 2$ time steps, $S_0 = I$, and $Q_0 = diag(100, 50)$, and for d_o model, $K_1 = diag(0.5, 0.5)$, $K_2 = diag(0.8, 1.5)$, $e_{ss,max} = -e_{ss,min} = [0.2 \;\; 0.2]$).

Simulation results during unmeasured disturbances rejection test ($\pm 10^o K$ step change in T_{cw1} and T_{cw2} at $t = 2$ sec) with the above configuration are shown in Figures 7.14–7.15 respectively. The results show a comparison between the system response when the difference between the plant and CTRNN output is added to the predicted output as an output disturbance, and the system response when the output integrated disturbances (*i.e.* d_o is added to the model output). Both controllers succeeded in rejecting the unmeasured non–zero mean disturbances without any input constraints violation or offset error, but the latter controller results in a shorter response time and smaller peak than the first controller does. Figure 7.16 show the system performance for the setpoint change test using the two NMPC strategies of the previous test. A good sepoint tracking is observed without input constraints violation.

To demonstrate the efficiency of the new NMPC algorithm; C1 (for first–order approximation of sensitivity equations with AD), and C2 (for sensitivity calculations using Taylor series expansion with AD), the total computation time (plant with controller simulation time until $t = 20$ sec) of two–CSTR/S2 system at unmeasured disturbance test (positive step change of T_{cw1} and T_{cw2}) is compared with two other NMPC controllers (see Table 7.4); C3 using MATLAB *ode23* solver plus perturbation to get sensitivity and C4 using AD to solve the differential equations but with perturbation to get sensitivity. Table 7.4 shows that the differential equation solver using AD reduce computational time by more than one half times (comparing C2 with C1 and C3 with C4) and ($n_y P \times n_u M$) sensitivity variables calculation using AD saves another order of magnitude in time (comparing C2 with C4 and C2 with C3). For differentiation using perturbation approaches, the computational time is very sensitive to the number of independent variables (C3 and C4) whilst for AD approaches, it is insensitive in this case study (C1 and C2).

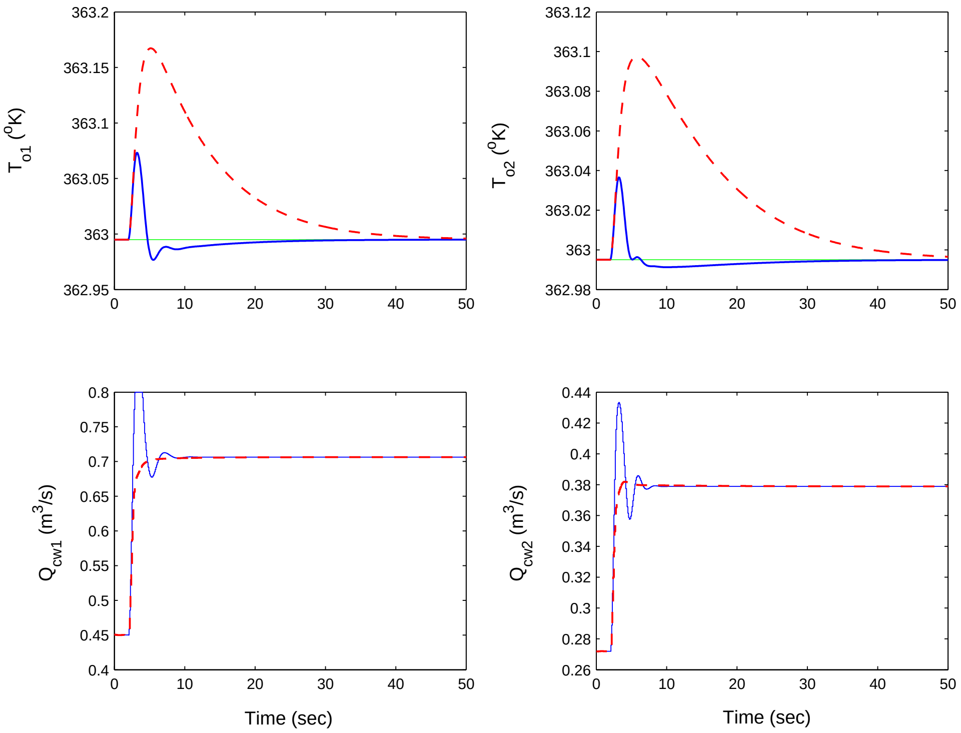

Figure 7.14: Two–CSTR/S2 response at unmeasured disturbance test (+10 ^oK step change in T_{cw1} and T_{cw2} at $t = 2$ sec) using; NMPC with output disturbance as the difference between the plant and the CTRNN model (dashed lines), NMPC with output integrated disturbance (solid lines).

Table 7.3: CTRNN Training for two–CSTR/S2 process, Computing Time and Accuracy Comparison

Traditional Sensitivity Approach			ADOL–C		
Tolerance	Time, sec	Error	Actual Order	Time, sec	Error
10^{-3}	40.61	1.5899	3	2.437	0.001
10^{-6}	162.281	0.0738	6	4.078	1.562e-7
10^{-8}	272.657	4.551e-4	8	5.391	2.606e-10
10^{-10}	316.375	1.864e-6	10	6.937	1.0729e-12

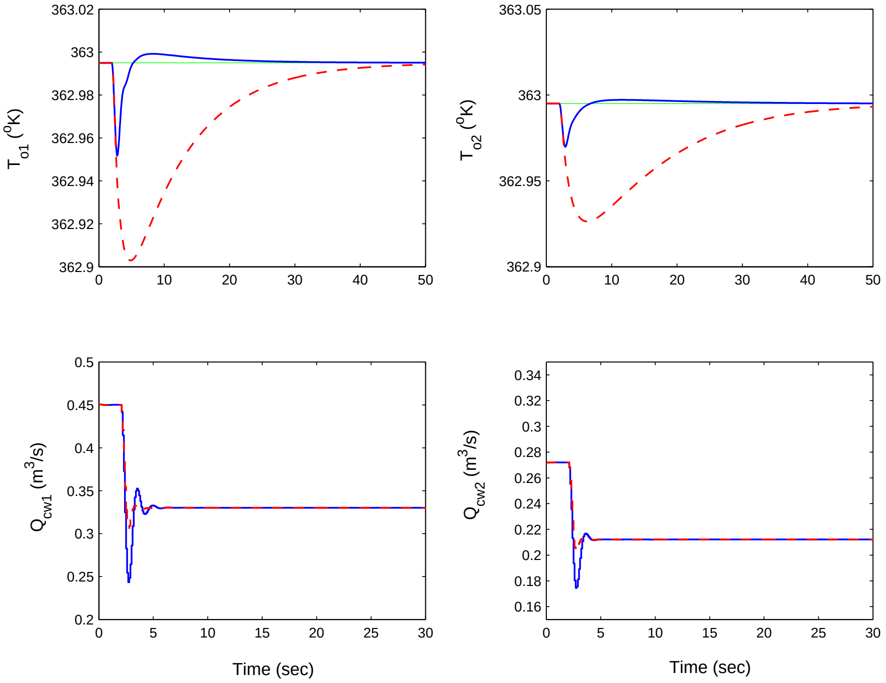

Figure 7.15: Two–CSTR/S2 performance at unmeasured disturbance test (-10 ^oK step change in T_{cw1} and T_{cw2} at $t = 2$ sec) using; NMPC with output disturbance as the difference between the plant and the CTRNN model (dashed lines), NMPC with output integrated disturbance (solid lines).

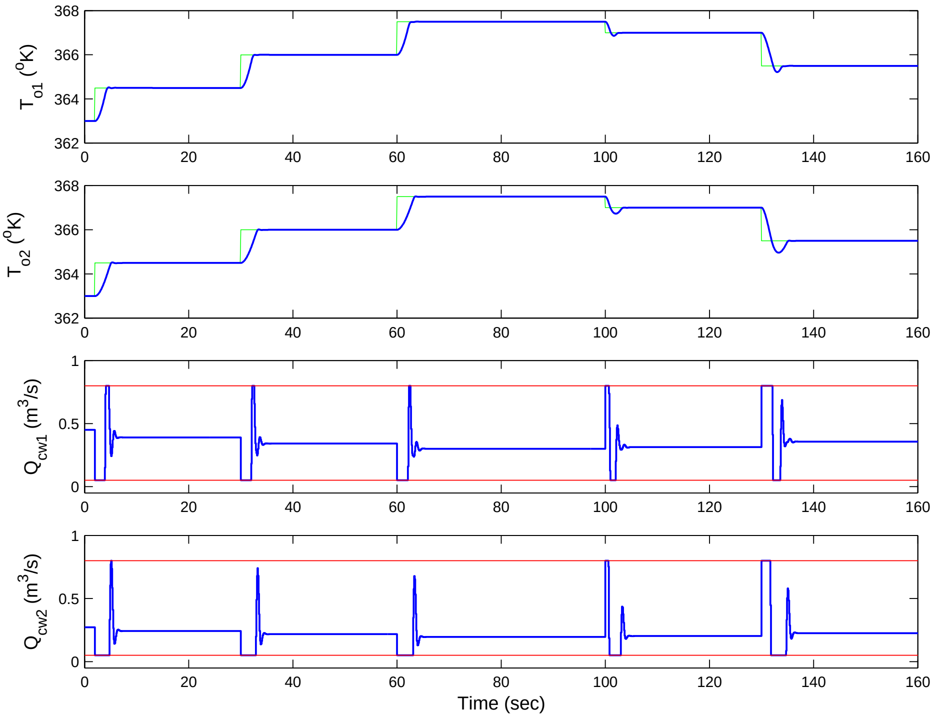

Figure 7.16: Two–CSTR/S2 performance during setpoint tracking test

Table 7.4: NMPC for two–CSTR/S2, Simulation Time (sec) Comparison

Controller Parameters	C1	C2	C3	C4
$P = 5, \quad M = 2$	16.25	**7.782**	26.047	9.078
$P = 10, \quad M = 2$	19.312	**8.563**	48.359	18.079
$P = 10, \quad M = 5$	23.984	**10.609**	102.891	33.219
$P = 50, \quad M = 10$	115.063	**43.765**	802.61	318.75

7.6 Summary

This chapter demonstrates the application of the proposed NMPC algorithm to the case of black box processes. The process is modelled first using a continuous–time artificial neural network. An algorithm has been proposed to train a continuous-time recurrent neural network to approximate a nonlinear dynamic systems given as a black box, so that the trained network can be used as the internal model for a nonlinear predictive controller. The new training algorithm is based on the efficient Levenberge Marquardt method combined with another efficient and accurate tool, automatic differentiation. Based on the identified neural network model, a NMPC controller has been constructed. The same strategy that was used in the network training was used to solve the online optimization problem of the predictive controller. The evaporator process and the two–CSTR/S2 process are chosen as a case studies to test the new system identification method and the NMPC algorithm with satisfactory results.

Chapter 8

MPC for the ALSTOM Gasifier Benchmark Problem

8.1 Introduction

The ALSTOM gasifier is chosen in this chapter as a new case study. The process has been issued by ALSTOM Power Technology Center as a benchmark challenge. Its importance comes from the fact that it is in the heart of the clean energy program for producing gas from underground coal deposits. Linear and nonlinear MPC algorithms are developed here to control the benchmark problem. The process model is provided in the form of a Simulink black–box in MATLAB. The internal model required for the predictive controller must be constructed via system identification as a first step. LMPC based on the state–space formulation is examined first to control the plant. Then, an extension to NMPC via Wiener model linearizion is considered. Of all three operating points of the plant, 0% load is identified as the most difficult case to control. Hence, a linear model of the plant at this load is adopted as the base model for prediction in the case of using the LMPC. Due to this choice, the control system comfortably achieves performance requirements at the most difficult load condition. The convex quadratic programming (QP) routine is used to solve the online optimization problem. Meanwhile, the case study shows that the model is also adequate for other load conditions to pass all tests specified in the benchmark problem.[1]

A further performance improvement is obtained when the LMPC algorithm is extended to NMPC one via model linearization. In this case, a partially nonlinear

[1] Most of material in this chapter is the subject of published papers e.g. [ACY04, ACY06, ASC05a, ASC05b].

state–space class Wiener model is used to identify the process model. The dynamic part of the Wiener model is chosen as the same as the linearized model at 0% load. In the static part of the model, a FFNN is created for a particular output channel, fuel gas pressure, to compensate its strong nonlinear behavior observed in open-loop simulation. By linearizing the neural network model at each optimization problem, the static nonlinear gain provides certain adaptation to the linear base model. The AD tool is used efficiently in the linearizion of the FFNN. Noticeable performance improvement is observed by comparing with pure linear model based predictive control. Both MPC algorithms use standard formulation and off-the-shelf software with a few tunable parameters. Thus, it is easy to implement and to tune to achieve satisfactory performance.

8.2 The ALSTOM Gasifier

8.2.1 Gasification

Power generation is responsible for a significant part of the total emission of solid, liquid and gaseous pollutants in Europe. Due to the predicted long–term high availability of solid fuels, in particular coal, compared to oil and natural gas, solid fuels will play an important part in future energy supply. Further, the demand for clean air and stringent environmental regulations are forcing consideration of alternative technologies, with high energy conversion efficiency and reduced pollutant emissions. However, power generation by conventional coal firing has a much greater environmental impact compared to natural gas based systems. Therefore, combustion technologies have been developed which aim at an environmentally advantageous use of coal in power generation plants [McS97]. As a result of this, Integrated Gasification Combined Cycle (IGCC) power plants, combining gasification with gas and steam cycles, are being developed around the world. The gasification plant considered in this study is based on the spouted fluidized bed gasification concept and can be considered as a reactor where coal is gasified with air and steam to produce low calorific value fuel gas, which then can be burnt in a suitably adapted gas turbine. In modern advanced power generating plants, gasification helps burning coal in a new and environmentally clean process.

As a part of the UK's Clean Coal Power Generation Group, ALSTOM has undertaken a detailed feasibility study on the development of a small-scale prototype integrated plant (PIP), based on the air-blown gasification cycle. The gasifier is one component

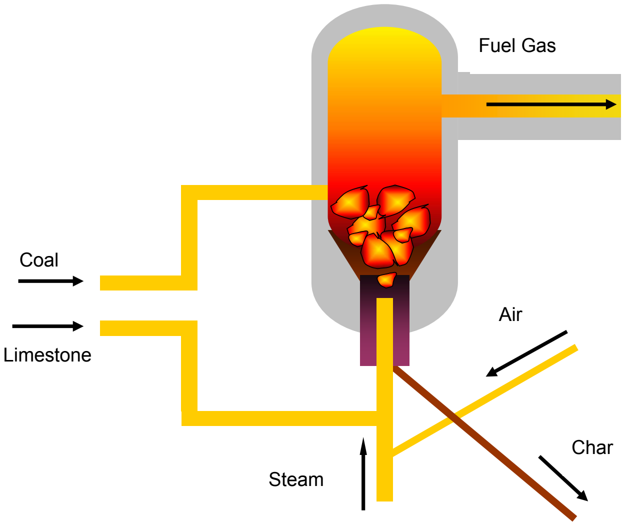

Figure 8.1: ALSTOM Gasifier

of the model which, as a highly coupled multi-variable system, has been found to be particularly difficult to control [DPM00]. For this reason, together with its associated control specification, operating constraints and various disturbance characteristics, ALSTOM coal gasifier has been issued by the ALSTOM Power Technology Center as a benchmark challenge in 1997 and a second round challenge in 2002.

The first round challenge included three linear models representing three operating points of the gasifier at 0%, 50% and 100% load conditions respectively. The challenge requires a controller to control the gasifier at three load conditions to satisfy input and output constraints in the presence of step and sinusoidal disturbances [DPM00]. An overview and a comparison of various control approaches submitted to the first round challenge are given in [Dix99].

None of the controllers proposed in the first round of challenge managed to meet all the performance criteria while satisfying the specified constraints. Using MPC to control the same plant has previously been studied at the first round of challenge (see ref. [RRS02]). The only MPC approach proposed to the first round challenge by Rice et al [RRS02] involved the use of a LMPC with an additional inner loop to stabilize the process. The inner loop controller is supervised by an outer loop to handle the process constraints. However, the attempt was not very successful in terms of satisfying all performance specifications.

The second round of the challenge issued in 2002 extended the original problem by providing participants with a nonlinear simulation model of the gasifier in MATLAB/SIMULINK [Dix02]. In addition to the original disturbance test, two extra tests: load change and coal quality disturbance tests were included. Recently, a group of control solutions for the benchmark problem were presented at Control-2004 Conference at Bath University, UK in September 2004. Most of controllers were reported as capable of controlling the system at disturbance and load change tests. However, this was not the case for the coal quality disturbance test because of the char flow rate saturation behavior.

8.2.2 The Gasifier Plant Details

The gasifier plant for the PIP shown in Figure 8.1 is based on the British Coal experimental gasifier, making use of the spouted fluidized-bed gasification concept and can be considered as a reactor where coal is gasified with air and steam. In simple terms the gasification process works as described below.

Pulverized coal mixed with limestone, which captures sulphur originating in the coal, is conveyed by pressurized air into the gasifier. The air and injected steam not only fluidize the solids in the gasifier but also react with the carbon and volatilize from the coal, producing a low calorific value fuel gas (approximately 4.6 MJ/kg or 12 per cent of that of natural gas). The remaining char (ash from coal, limestone and un–reacted carbon) is removed as bed material from the base of the gasifier as elutriated fines with the product gas. Under certain circumstances as much as 70 per cent of the total char off-take may leave the gasifier as elutriated fines [GBP04].

A schematic of the plant is shown in Figure 8.2. The gasifier is a 5×4, MIMO nonlinear plant. The controllable inputs are; flow rates of char extraction, coal, air, steam, and limenstone, and the output variables are; gas quality, bed-mass, pressure, and temperature. One of the inputs, limestone mass (WLS) is used to absorb sulphur in the coal and its flow rate is set to a fixed ratio of 1:10 against another input (WCOL). This leaves effectively 4 degrees of freedom for the control design. The plant inputs and outputs with their limits are given in Tables 8.1 and 8.2, respectively.

8.2.3 Black Box Model

The coal gasifier model was developed using the Advanced Continuous Simulation Language(ACSL) [Adv99]. Then it is transferred to MATLAB/SIMULINK to facilitate the design and evaluation of control law [DP04]. All the significant physical

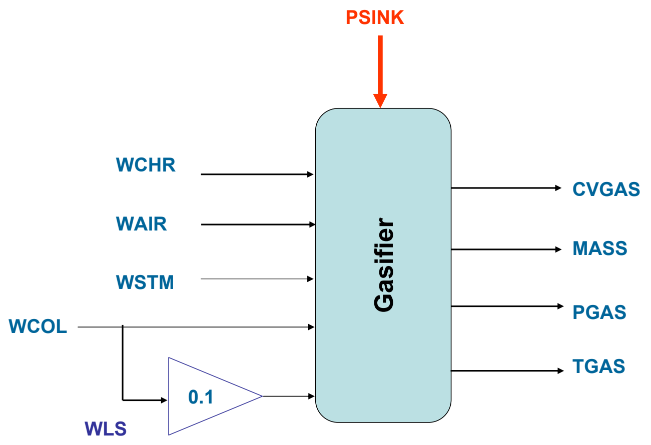

Figure 8.2: A simple diagram of the gasifier process

Table 8.1: Output variables and limits

		Allowed	Steady state values		
Outputs	Description	Fluctuations	100%	50%	0%
CVGAS (MJ/kg)	Fuel gas clarification	± 0.01	4.36	4.49	4.71
MASS (kg)	Bed mass	± 500	10000	10000	10000
PGAS (N/m^2)	Fuel gas pressure	$\pm 1 \times 10^4$	2.0×10^6	1.55×10^6	1.12×10^6
TGAS (K)	Fuel gas temperature	± 1.0	1223.2	1181.1	1115.1

Table 8.2: Input variables and limits

		Maximum	Peak	Steady state values		
Inputs	Description	Value	Rate	100%	50%	0%
WCHR (kg/s)	Char extraction	3.5	0.2 kg/s^2	0.90	0.89	0.50
WAIR (kg/s)	Air flowrate	20	1.0 kg/s^2	17.42	10.89	4.34
WCOL (kg/s)	Coal flowrate	10	0.2 kg/s^2	8.55	5.34	2.136
WSTM (kg/s)	Steam flowrate	6.0	1.0 kg/s^2	2.70	1.69	0.676
WLS (kg/s)	Limestone flowrate	1.0	0.02 kg/s^2	0.85	0.53	0.21
PSINK (N/m^2)	Sink pressure	–	–	18.5e5	14.8e5	11.1e5

effects are included in the model (e.g. drying processes, desulphurization process, pyrolysis process, gasification process, mass and heat balances) and it has been validated against measured time histories taken from the British Coal CTDD experimental test facility. More detail concerning the model can be found in [Dix02].

The foundation of the model is a black box C–code s–function, which is included within a SIMULINK block diagram (see Figure 8.3). Other nonlinear inputs for the

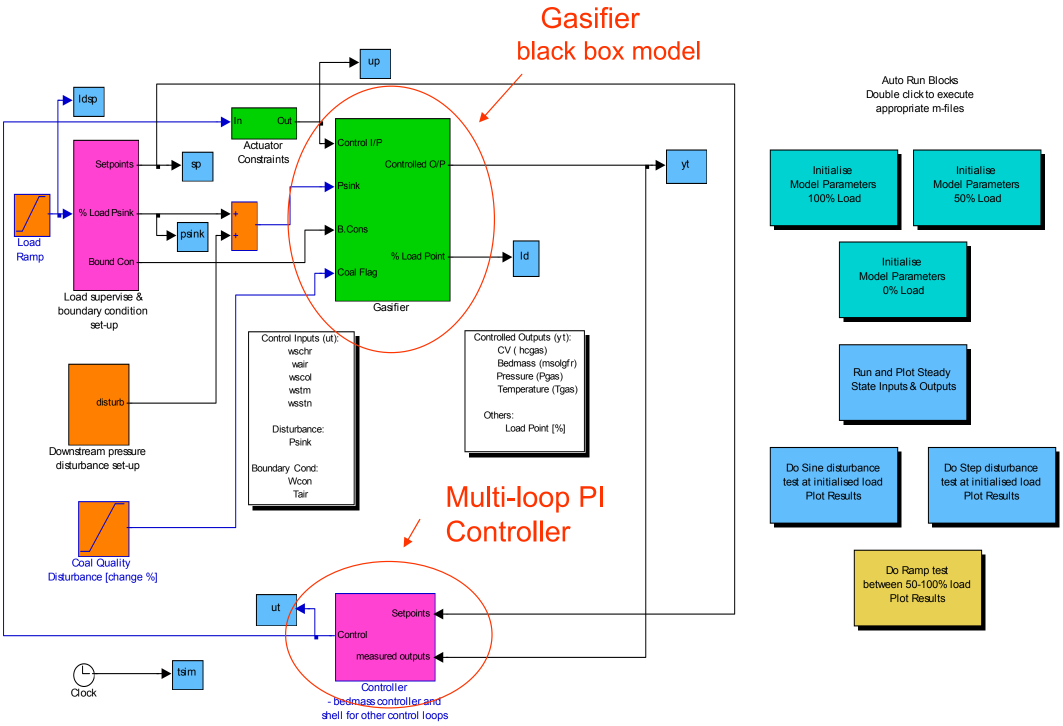

Figure 8.3: The baseline Simulink model of the gasifier (provided by ALSTOM company)

model include boundary condition (to allow manoeuvres to different operating points), a disturbances input, sink pressure (PSINK) which represents the pressure upstream of the gas turbine which would vary according to the position of the gas turbine fuel valve, and a coal quality input. At the various operating conditions the steady-state conditions for the nonlinear model are given in Tables 8.2 and 8.1 respectively.

The full model of the gasifier has 25 states and the aim of the benchmark challenge is to design a controller to work with the given nonlinear SIMULINK model as the gasifier plant, to satisfy the control performance.

Two models were provided for the challenge: one open–loop with a simple bed–mass control to avoid the system drifting from steady state, and the other, with a PI multi–loop controller which is provided to demonstrate a baseline performance as shown in Figure 8.3. The multi-loop PI controller was able to control the system in most of the required tests except in the case of sine wave disturbance test at 0% load since an upper constraint violation happened in PGAS output variable as shown in Figure 8.4.

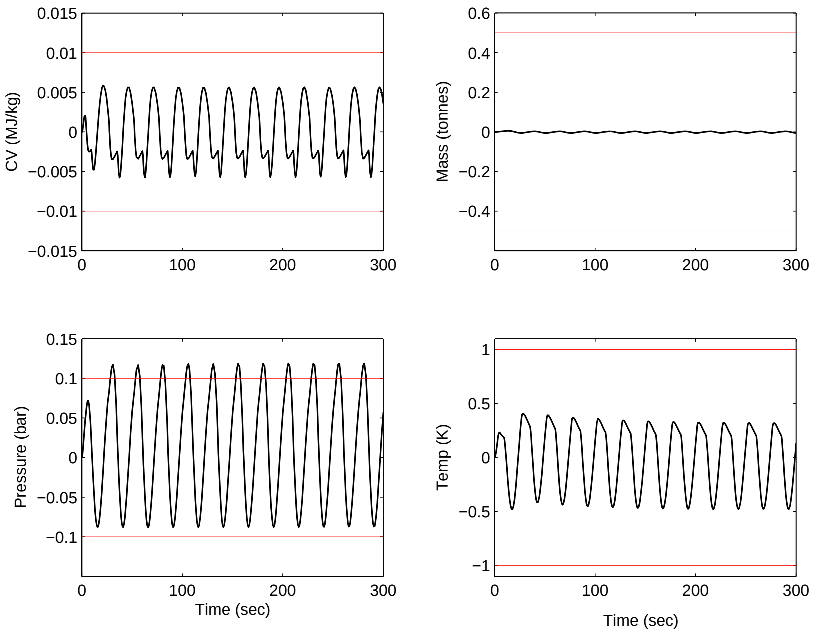

Figure 8.4: Output response to sinusoidal disturbance at 0% load using multi–loop PI controller.

8.3 Control Specification

The aim of this benchmark is to design a controller using the nonlinear black box model as the plant, to satisfy the following performance specification:

- All the system variable limits must not be exceeded when step or sine wave PSINK disturbance are applied.
- The robustness of the controller determined at one load condition is to be evaluated at other load conditions during the PSINK disturbance tests.
- The plant should be driven to a number of operating (load change test) and the stability of the plant over the working range should be satisfied.
- Control the system when the coal quality input changes by ±18%.

8.4 Linear MPC Attempts

Linear models with linear constraints and quadratic objective function results in a convex optimization problem easily solvable using QP. In this section, a linear MPC approach based on the state–space formulation is implemented to control the ALSTOM gasifier as a first trial.

8.4.1 LMPC formulation

Assume that the plant considered has manipulated input, $\tilde{u} \in \mathbb{R}^{n_u}$ and measured output, $\tilde{y} \in \mathbb{R}^{n_y}$, which have steady-state values, $\tilde{u}_0$ and $\tilde{y}_0$ at the nominal operating point respectively. Around the operating point, the dynamic behavior of the plant can be approximated by the following linear discrete-time state-space equations:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + d(k) \end{aligned} \tag{8.1}$$

where; k, stands for kth sampling time, $u(k) = \tilde{u}(k) - \tilde{u}_0$, and $y(k) = \tilde{y}(k) - \tilde{y}_0$, are deviation variables, $d(k)$: the virtual disturbance estimated at output, $x(k)$, the internal state of the model.

The model and plant are assumed to be coincident at the nominal operating point with $x(0) = 0$, $u(0) = 0$, $y(0) = 0$ and $d(0) = 0$. At the kth sampling time, with currently measured output, $y_m(k) = \tilde{y}(k) - \tilde{y}_0$ and current state, $x(k)$, the future output within the prediction horizon, P can be estimated from the future input (to be determined within the moving horizon, M), $u(k)$ as follows: assuming

$$d(k+i) = d_k = y_m(k) - Cx(k), \quad \text{for} \quad i = 1, \ldots, P \tag{8.2}$$

then

$$Y = \Phi U + \Psi x(k) + Ld_k \tag{8.3}$$

where:

$$Y = \begin{bmatrix} y^T(k+1) & \cdots & y^T(k+P) \end{bmatrix}^T \tag{8.4}$$

$$U = \begin{bmatrix} u^T(k) & \cdots & u^T(k+M-1) \end{bmatrix}^T \tag{8.5}$$

$$\Phi = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ CA^{P-1}B & CA^{P-2}B & \cdots & \sum_{i=M}^{P} CA^{P-i}B \end{bmatrix} \tag{8.6}$$

$$\Psi = \begin{bmatrix} CA \\ \vdots \\ CA^P \end{bmatrix} \tag{8.7}$$

$$L = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix} \tag{8.8}$$

Future input, U is determined to follow the output reference, $y_r(k)$, and the input reference $u_r(k) = G_0^{-1}(y_r(k) - d_k)$, where $G_0 = C(I - A)^{-1}B$. Define input and output reference vectors as

$$Y_r = \begin{bmatrix} y_r^T(k+1) & \cdots & y_r^T(k+P) \end{bmatrix}^T \tag{8.9}$$

$$U_r = \begin{bmatrix} u_r^T(k) & \cdots & u_r^T(k+M-1) \end{bmatrix}^T \tag{8.10}$$

Then, $U_r = \Lambda(Y_r - Ld_k)$, where

$$\Lambda \;=\; \underbrace{\begin{bmatrix} G_0^{-1} & 0 & \cdots & 0 \\ 0 & G_0^{-1} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & G_0^{-1} \end{bmatrix}}_{M}\underbrace{\begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}_{P-M} \tag{8.11}$$

The optimization problem is to minimize the performance cost:

$$\begin{aligned} J \;&=\; 0.5(Y - Y_r)^T Q (Y - Y_r) + 0.5(U - U_r)^T R (U - U_r) \\ \text{s.t.} \quad & \underline{u} \le u \le \overline{u} \\ & |u(k+1) - u(k)| \le \delta_u \end{aligned} \tag{8.12}$$

where, output and input weighting matrices, Q and R are positive definite and $\underline{u}$, $\overline{u}$ and δ_u are the lower, upper and maximum rate bounds of the input respectively.

Using the predictive equation (8.3), the optimization problem is equivalent to a standard quadratic programming (QP) problem:

$$\begin{aligned} J \;&=\; 0.5U^T S U + U^T(X_1 x(k) - X_2(Y_r - Ld_k)) \\ \text{s.t.} \quad & U \le \overline{U}, \quad -U \le -\underline{U} \\ & FU \le \Delta_u + Zu(k-1) \\ & -FU \le \Delta_u - Zu(k-1) \end{aligned} \tag{8.13}$$

where, $u(k-1)$ is the previous input, and other variables are defined as follows.

$$\begin{aligned}
S &= \Phi^T Q \Phi + R && (8.14)\\
X_1 &= \Phi^T Q \Psi && (8.15)\\
X_2 &= \Phi^T Q + R\Lambda && (8.16)\\
\overline{U} &= \begin{bmatrix} \overline{u}^T & \cdots & \overline{u}^T \end{bmatrix}^T, \quad \underline{U} = \begin{bmatrix} \underline{u}^T & \cdots & \underline{u}^T \end{bmatrix}^T && (8.17)\\
\Delta_u &= \begin{bmatrix} \delta_u^T & \cdots & \delta_u^T \end{bmatrix}^T && (8.18)\\
F &= \begin{bmatrix} I & 0 & \cdots & 0 & 0 \\ -I & I & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & -I & I \end{bmatrix} && (8.19)\\
Z &= \begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix}^T && (8.20)
\end{aligned}$$

Note, in the above formulation, output constraints are neglected to simplify the algorithm and to fully use the plant capability. The QP problem (8.13) is efficiently solvable by off-the-shelf software. The only tunable parameters in the above formulation are Q, R, P, M and the sampling time. Thus, the control strategy can be easily implemented and tuned to satisfy required performance.

In vector U, only the first n_u rows, corresponds to $u(k)$ are applied to the plant. The whole procedure is repeated at the next sampling instance.

For the unconstrained case, the optimal solution, corresponding to a state feedback control law, can be obtained analytically:

$$U = -K_x x(k) + K_y(y_r - d_k) \tag{8.21}$$

where $K_x = S^{-1}X_1$ and $K_y = S^{-1}X_2$. Let K be the first n_u rows of K_x, then the nominal stability (perfect model without input saturation) of the closed-loop can be checked by calculating the eigenvalue of the matrix, $A - BK$.

8.4.2 Controller Design

The first task to implement the control design using the above algorithm is to determine an internal model of equation (8.1). Three operating conditions are specified in the gasifier benchmark problem to represent 0%, 50% and 100% load conditions. All load conditions are subject to disturbance tests. Among these tests, it is observed that those under 0% load condition are most difficult to pass. That is understandable because a plant at a lower load condition (small throughput) normally exhibits larger time lagging and a larger gain. Therefore, a control system tuned under a higher load condition will tend to be unstable (or stability margin reduced) at a lower load condition. On the other hand, a controller designed for a lower load condition is more likely to work well without re-tuning at higher load conditions although performance might deteriorate when compared with a re-tuned controller at higher load conditions. Since the performance requirements at 50% and 100% load conditions are easier to achieve, it is decided to use the 0% load point as the nominal point to get the linearized state space model.

A linearized state space model is obtained from the nonlinear simulation model at 0% load condition. This linear model is then reduced to 16 states via pole-zero cancellation (using Control System Toolbox functions, *ssbal* and *minreal*). The 16-state model is then discretized with the sampling time selected as follows.

Normally, the sampling time should be less than one tenth of $2\pi/\omega_b$, where ω_b is the required bandwidth of the closed-loop. The benchmark requires to reject a sine disturbance with a period of 25 seconds (0.04 Hz). Therefore, the sampling time should be less than 2.5 seconds. On the other hand, the sampling time should not be too large so that in step disturbance tests, the output variables will not deviate from setpoints more than the specified limits before the controller can start to respond. Several open-loop tests for a step disturbance of PSINK at three load conditions are performed. The output response results are shown in Figure 8.5. The results show that, the worst response case is the 0% condition, where, without control, the pressure output can only stay within specified range for a period of 1.2 seconds. Hence, the sampling time is selected to be 1 second. This satisfies the requirements for both disturbance tests.

The above algorithm is implemented in MATLAB as a SIMULINK s-function to replace the control block in the nonlinear simulation model provided in the benchmark suit. The QP problem is solved by calling *quadprog* of the Optimization Toolbox at each sampling time. This is the major computation burden in the above algorithm and is solely determined by the control horizon, M. The prediction horizon, P has little effect on computation time, thus can be selected relatively large to benefit stability.

To tune M and P, initially let $P = M$. By varying M from 1 s to 12 s, a stable performance is obtained which satisfies all control specifications for 7 s $\leq M \leq$ 10 s. When $M \geq$ 10 s, the improvement on the system performance is negligible but computation time increases significantly. Therefor $M = 9$ s is selected, which gives a good per-

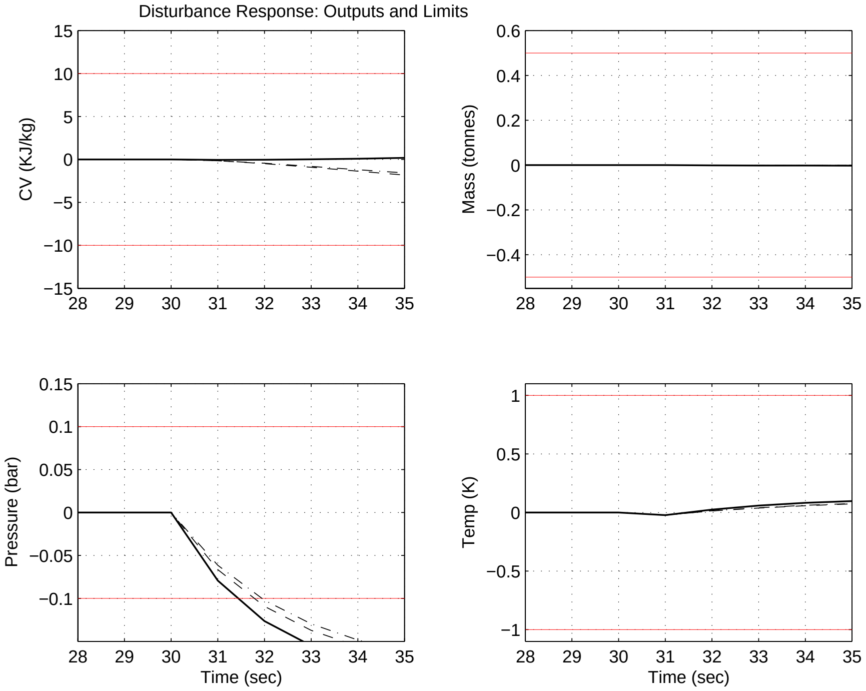

Figure 8.5: Open-loop output response to a step disturbance at 30s at 0% (solid), 50% (dashed) and 100% (dash-dotted) load conditions.

formance in all tests. To choose a suitable prediction horizon P, a reasonable range from the minimum value ($P = M = 9$ s) to $P = 25$ s has been tested. A stable response without any constraint violation is found within the range 15 s $\leq P \leq$ 20 s. No performance improvement can be observed when $P \geq 20$ s. Therefore $P = 20$ s (the maximum value of the range) is chosen to ensure both the system stability and satisfactory control performance are achieved within a reasonable computation time.

The weighting matrix, $Q = \text{diag}(Q_0, \cdots, Q_0)$, where Q_0 is diagonal and initially set to be the inverse of the output error bounds. After online tuning, the final values are:

$$Q_0 = \begin{bmatrix} 0.15 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 2.1 & 0 \\ 0 & 0 & 0 & 2 \times 10^6 \end{bmatrix} \tag{8.22}$$

Also, the input weighting matrix $R = \text{diag}(R_0, \cdots, R_0)$, where R_0 is diagonal and set to the following value after online tuning;

$$R_0 = \begin{bmatrix} 10^5 & 0 & 0 & 0 \\ 0 & 5 \times 10^3 & 0 & 0 \\ 0 & 0 & 5 \times 10^3 & 0 \\ 0 & 0 & 0 & 10^4 \end{bmatrix} \tag{8.23}$$

Using the above configuration, nominal stability is achieved at all three load conditions, *i.e.* the magnitudes of all eigenvalues of $A_i - B_iK$ are less than 1. Where, A_i and B_i are the discrete states and control matrices at different load conditions.

One of the advantages of MPC is that future setpoint change information is incorporable into the QP optimization problem to improve setpoint tracking performance. This is implemented in the gasifier controller.

8.4.3 Simulation Results

1- *PSINK Disturbance Tests*:

The following two disturbance tests are performed for three load conditions for 300 seconds:

1. step change in sink pressure of -0.2 bar at 30s;
2. 0.04 Hz sinusoidal variation in sink pressure of amplitude 0.2 bar beginning at 30s.

The maximum and minimum values as well as the peak rate change of the input variables of the two tests under different load conditions are shown in Table (8.4).

The maximum absolute error between output variables and their setpoints and the integral of absolute error (IAE) of these variables are calculated in Table (8.3). The corresponding plots of the normalized output errors at different load conditions are shown in Figure 8.6. The normalized error of the output variables is defined as;

$$e_y(t) := \frac{|\tilde{y}_m(t) - \tilde{y}_r(t)|}{e_{max}} \tag{8.24}$$

where, $e_{max} \in \mathbb{R}^{n_y}$ is the maximum value of the allowed output fluctuation around the nominal value for each plant output. All the values of e_y shown in Figure 8.6 are positive and less than one which means that there is no output constraints violation.

The corresponding plots of the normalized input variables at different load conditions are shown in Figure 8.7. Note, the input variables are normalized using;

$$u_n(t) := \frac{u(t)}{u_{max}} \tag{8.25}$$

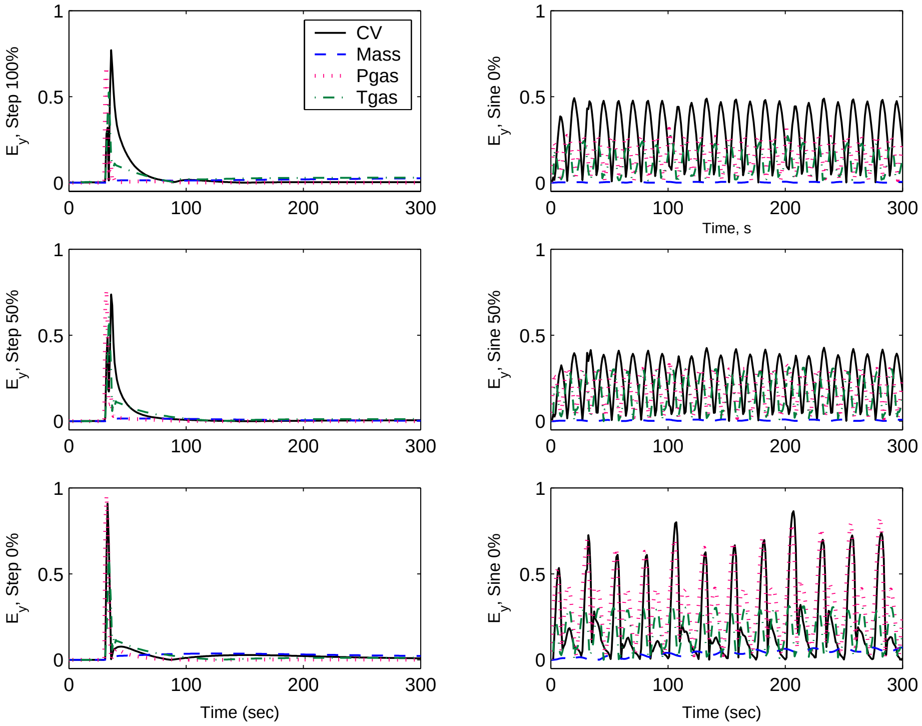

Figure 8.6: Gasifier output responses during PSINK disturbance test.

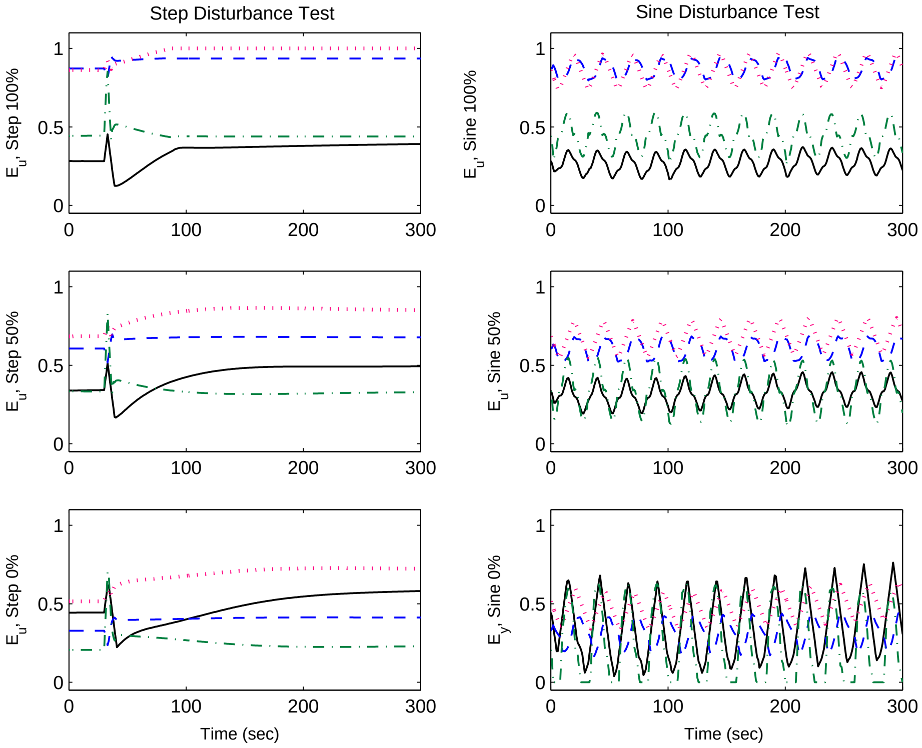

Figure 8.7: Gasifier input responses during PSINK disturbance test, WCHR (sold), WAIR (dash), WCOL (dot), WSTM (dot–dash).

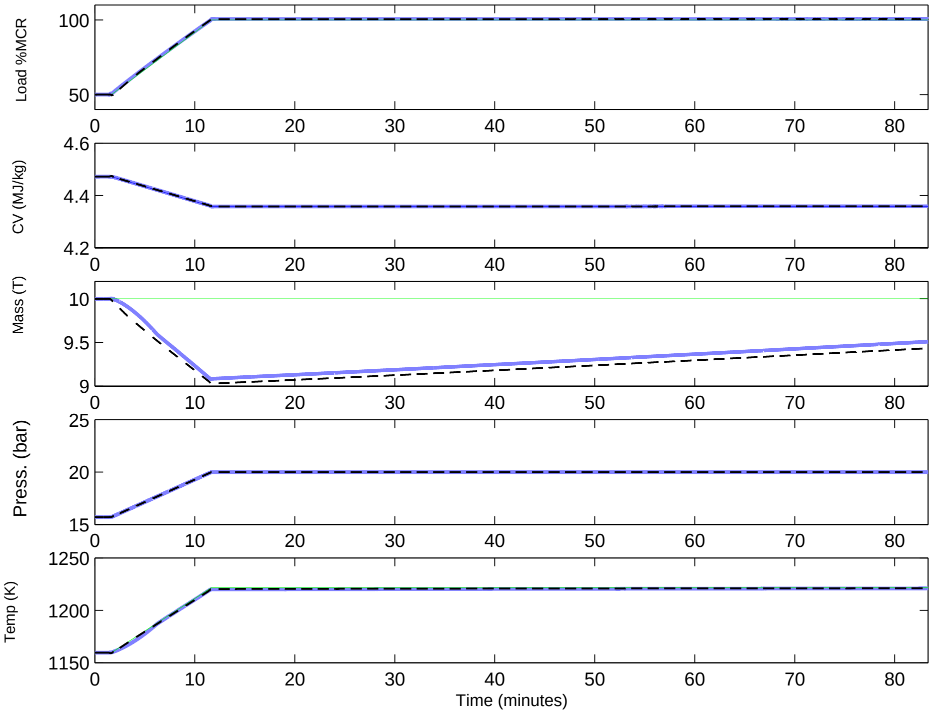

Figure 8.8: Output response to the load setpoint change using; (1) LMPC (dashed lines) (2) NMPC (solid lines).

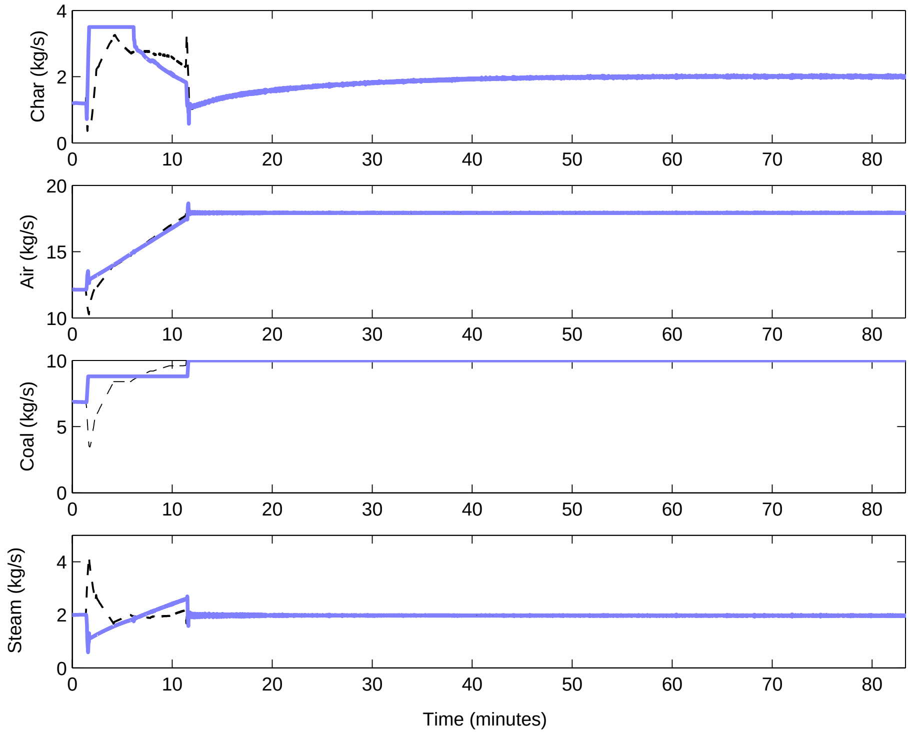

Figure 8.9: Input response to the load setpoint change using;(1) LMPC (dashed lines) (2) NMPC (solid lines).

where u_{max} is the maximum value of the input variable. No constraint violation was detected in the manipulated variables responses.

The output response plots show that the predictive controller based on a linear model identified at 0% load is able to maintain output variables within the limits specified for both tests. Particularly, the performance at 0% load is significantly improved from the one achieved by the PID configuration provided in the benchmark (see Figure 8.5). However, this improvement is traded with the price of performance deterioration at other load conditions. Thus, using a nonlinear model might be necessary in order to further improve performance at all load conditions.

2- *Load Change Test*:

In this test, the load is required to increase from 50% to 100% within time from 100s to 700s. The actual response is collected from simulation and compared with the demand in Figures 8.8 to 8.9. Significant improvement in the setpoint tracking performance can be observed from the results. This improvement is due to the advantage of predictive control to wisely use future setpoint information in the online optimization.

3- *Coal Quality Change Test*:

The benchmark problem includes a test of coal quality changes by ±18%. Physically, a positive coal quality change means an increase of energy per unit coal feed. To maintain the same level of load, it is expectable that coal feed and char outlet (ash) will decrease at steady state due to energy balance. Similarly, a negative coal quality change will increase coal feed and char outlet at steady state. Therefore, the feasible range of a coal quality change is restricted by the input constraints. If a coal quality change is beyond this feasible range, at steady-state some input saturations are inevitable and the control problem becomes infeasible, *i.e.* there is no controller which can achieve the performance specification.

Since an analytical model is not available, the feasible coal quality range is determined via simulation described as follows. Set the PID simulation model provided in the benchmark by removing all actuator constraints. Then repeatedly perform simulation until steady state by introducing a different coal quality change but without any other disturbance. By checking the steady-state input values against their constraints the following feasible coal quality ranges for different load conditions are identified.

100% load $-6.6\% \leq$ coal quality $\leq +7.1\%$

50% load $-14.1\% \leq$ coal quality $\leq +11.3\%$

0% load $-16\% \leq$ coal quality $\leq +18\%$

At the upper bounds of feasible coal quality ranges, WCHR reaches its lower bound at steady stated, whilst at the lower bounds of the feasible ranges WCHR saturates at its upper bound except at 100% load condition where WCOL, instead of WCHR, becomes saturated at its upper bound. The above results show that a performance deterioration is inevitable when a coal quality change is out of the above feasible range. This deterioration is independent of control design because of the inherent limitation imposed by the physical nature of the system.

Such performance deterioration is also observed for the MPC controller. It has been identified that the predictive controller can cope with coal quality change of ±18% for all standard disturbance tests for up to 600 seconds. For a simulation time longer than 600 seconds, output specifications are violated in sinusoidal tests under 0% load with a coal quality change of -18% and under 100% load with a coal quality change of +18%. A steady state test for a longer time (for example 15000 seconds) shows that the system cannot cope with a coal quality change either of -18% under 0% and 50% load conditions, or +18% under 50% and 100% load conditions.

In the presence of a sinusoidal disturbance test at 100% load, for positive coal quality change, WCHR tends to zero at $t \geq 200$ s as shown in Figure 8.11. When this happens for a sufficiently long time, the temperature starts to go up as more carbon has to be burned to balance the high coal quality inlet (Figure 8.10). Similarly, for sufficiently negative coal quality change, WCHR, and WCOL will be saturated at their upper bounds. The gasification process in this case is under–combusted and the outlet gas temperature will unavoidably drop.

8.5 NMPC via Linearization Approach

The LMPC above was able to attain all the required performance specifications within the input and output constraints at all load conditions. In this section, it is shown that the plant/model mismatch can be further reduced and an improvement to the whole system performance can be obtained if a partially developed nonlinear model is used instead of a pure linear model.

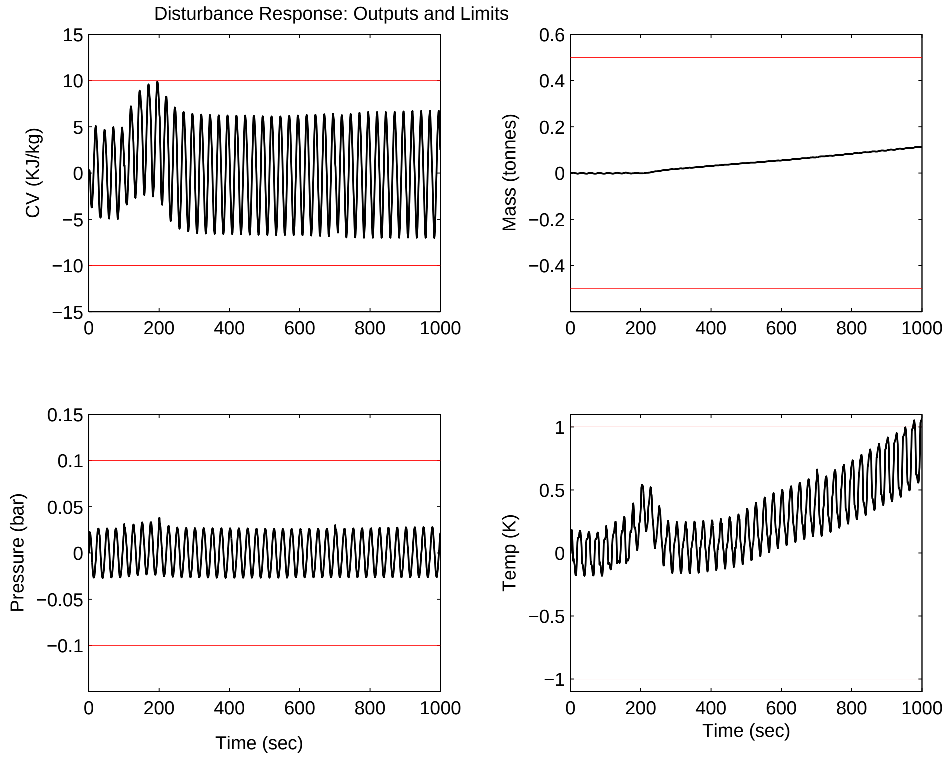

Figure 8.10: Output response for +18% step change in the coal quality at 100% load

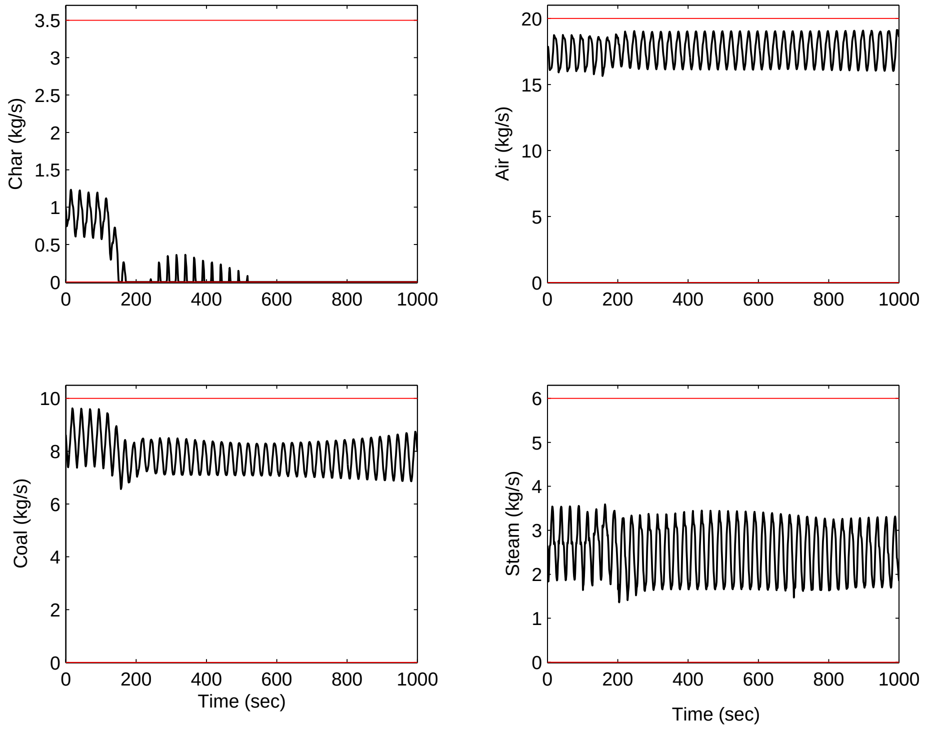

Figure 8.11: Input response for +18% step change in the coal quality at 100% load

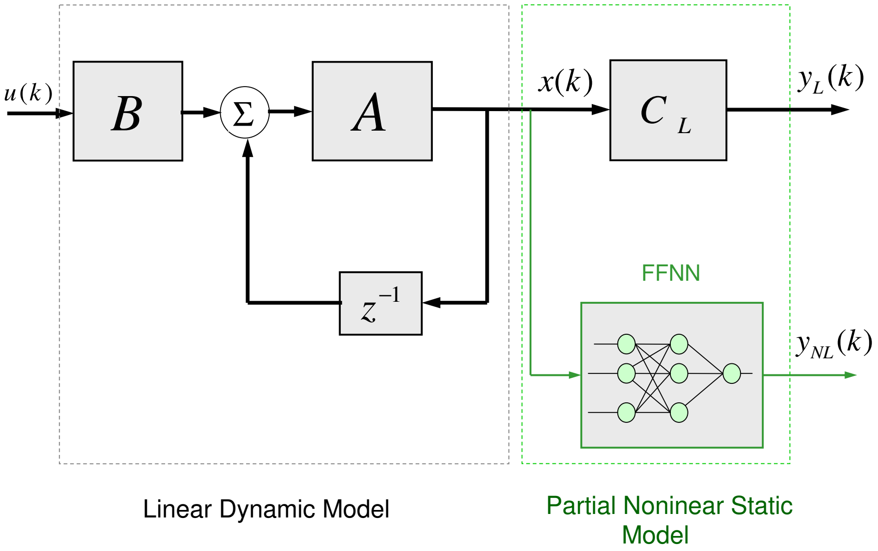

Figure 8.12: Nonlinear state–space type Wiener model for the gasifier process.

For extending linear MPC to NMPC, a model is required that can represent the salient nonlinearities but possibly without the complication associated with general nonlinear models. The Wiener model as mentioned earlier in chapter 2, corresponds to a process with linear dynamics but a nonlinear gain. It can adequately represent many of the nonlinearities commonly encountered in industrial processes [**?**, DFL94]. Due to the static nature of their nonlinearities, they can be effectively removed from the control problem, allowing the use of simpler linear programming algorithms. This model can be represented by a state-space model which as a general model form has been extensively used for control system analysis and design. In the case of ALSTOM gasifier, the previous linear (internal) model is extended to include some of the plant nonlinearities by developing a static nonlinear model in the form of Wiener class configuration as shown in Figure 8.12. Linear static gains are used for three outputs, CVGAS, MASS, TGAS, while, an artificial neural network model is created for the forth output PGAS.

The output selection was based on the open-loop step response comparison between the linear and nonlinear simulation model (see Figure 8.13). The results showed that the linear model can correctly capture the dynamic behavior in three of the four outputs for up to 30 sec (the practical range of the prediction horizon length) under all load conditions. However, the third output PGAS exhibits salient nonlinearities

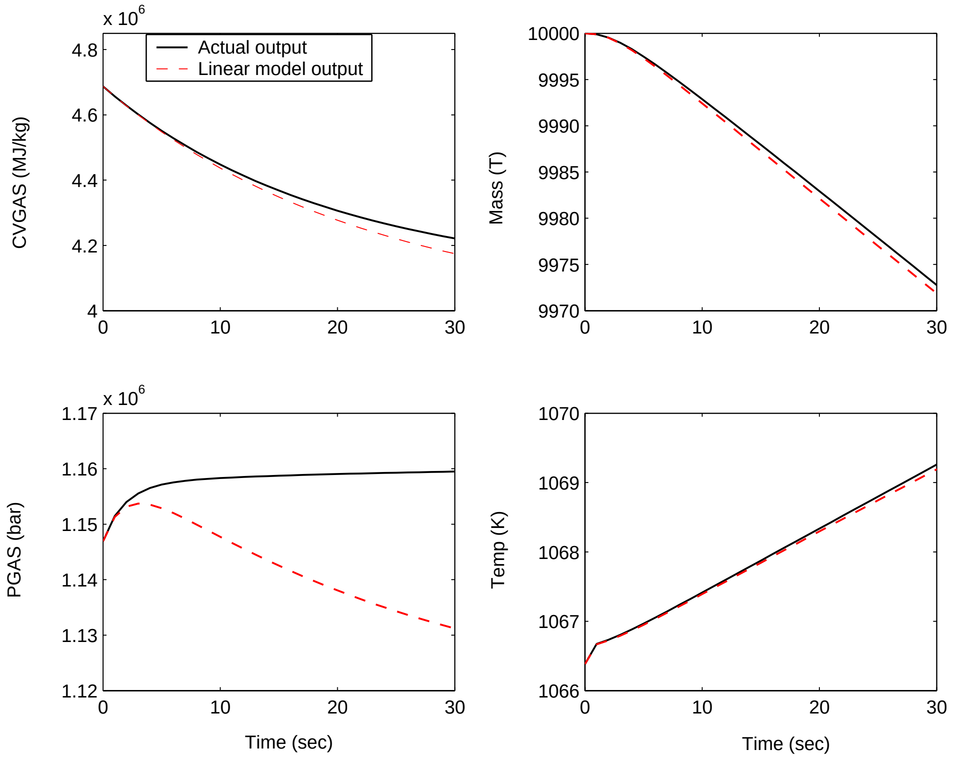

Figure 8.13: The gasifier open-loop response to a step change in WAIR at 0% load.

which cannot be predicted by the linear model. It is also observed that the effect of the unmeasured disturbance PSINK on the output variable PGAS is quite large, whilst the time constant of the response is very short compared to that of other outputs (see figure 8.13). Therefore, even for a short prediction horizon ($P \leq 20$ s), both transient and static responses of PGAS should be covered by a prediction model, while for other outputs only transient characteristics have dominant response. Hence, a Wiener model would not be very useful for the other variables unless a very long prediction horizon (about 10^4 s) is used which is clearly not practical.

Also, the neural network static gain of PGAS is trained to capture the static response at the three load conditions, so that, this step will reduce the performance deterioration at 50% and 100% load conditions caused by the system nonlinearity, and using a single dynamic model (which is linearized at 0% load) for all the operating points.

The PGAS nonlinear model was then linearized at every sampling instance to provide adaptation to the main linear controller. The partial nonlinearity compensated model leads to considerable performance improvement as shown later.

8.5.1 NMPC Formulation

In this case, around the operating point, the dynamic behavior of the plant can be approximated by the following nonlinear discrete-time state-space equations:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y_L(k) &= C_L x(k) + d_{L,k} \\ y_{NL}(k) &= f_{NN}(x(k)) + d_{NL,k} \end{aligned} \tag{8.26}$$

where u, x, and y are deviation variables as defined previously. In this model, the output variables are divided into two groups: y_L(k) outputs vector corresponding to the linear variables CVGAS, MASS, and TGAS, and $y_{NL}(k)$ corresponding to nonlinear output, PGAS. The matrix C_L represents the linear static gain, while f_{NN} is the nonlinear function modelled by a neural network. The terms $d_{L,k}$ and $d_{NL,k}$ are the virtual disturbances estimated from the plant measurement to correct the internal model prediction at the next optimization problem, which are given as;

$$\begin{aligned} d_L(k+i) &= d_{L,k} = y_m(k) - C_L x(k), \quad \text{for} \quad i = 1, \ldots, P \\ d_{NL}(k+i) &= d_{NL,k} = y_m(k) - f_{NN}(x(k)), \quad \text{for} \quad i = 1, \ldots, P \end{aligned} \tag{8.27}$$

The model and plant is assumed to be coincident at the nominal operating point with $x(0) = 0$, $u(0) = 0$, $y(0) = 0$ and $d_{NL}(0) = 0$.

The matrices A, B, and C_L are obtained by linearizing the nonlinear plant model at 0% load condition. The neural network model consists of two hidden layers and one output layer. The transfer functions of both hidden layers are the nonlinear function *sigmoid* type while a *linear* transfer function is used for the output layer. So that, the mathematical form of the function f_{NN} can be represented in a vector form as :

$$\begin{aligned} O_1(k) &= \sigma_s(W_1 x(k) + b_1) \\ O_2(k) &= \sigma_s(W_2 O_1(k) + b_2) \\ O_3(k) &:= y_{NL}(k) = W_3 O_2(k) + b_3 \end{aligned} \tag{8.28}$$

where $O_1(k), O_2(k)$ and $O_3(k)$ are the output values of each layer at time instant k. The values W_1, W_2, and W_3 are the weight parameters while b_1, b_2, and b_3 are the bias parameters. The function $\sigma_s(\cdot)$ is the sigmoid–*tanh* function which is defined previously in equation (7.5).

Because the model in equation (8.26) is nonlinear, the problem is no longer quadratic and a nonlinear programming optimizer will be needed to solve the problem, which can not guarantee to find the global minimum of the cost function every time. In order to obtained a convex solution by using efficient QP algorithm, and to establish a link between it and the PGAS Wiener model, a local linearization of the neural network model around the current states is implemented as follows.

Future predictions of output based on current measurement, $y_{NL}(k)$ can be approximated by the first two terms of the Taylor series expansion, for $i = 1, \cdots, P$:

$$\begin{aligned} y_{NL}(k+i) &\approx f_{NN}(x(k)) + \left.\frac{\partial f_{NN}}{\partial x}\right|_{x=x(k)} (x(k+i) - x(k)) && (8.29) \\ &= \rho(k) + C_{NL}(k)\ x(k+i) && (8.30) \end{aligned}$$

where:

$$\begin{aligned} \rho(k) &:= \rho_k = f_{NN}(x(k)) - \left.\frac{\partial f_{NN}(x)}{\partial x}\right|_{x=x(k)} x(k) && (8.31) \\ C_{NL}(k) &:= C_{NL,k} = \left.\frac{\partial f_{NN}(x)}{\partial x}\right|_{x=x(k)},\ \text{for } k \in [1, P] && (8.32) \end{aligned}$$

The value of the function $f_{NN}(x)$ and its partial derivative $\partial f_{NN}(x)/\partial x$ are calculated simultaneously and accurately from the neural network structure in equation (8.28) using the forward mode of AD tool, MAD Toolbox in MATLAB.

By replacing y_{NL} in equation (8.26) by the approximated value in equation (8.29), the term ρ_k is absorbed into $d_{NL}(k)$. This results in the following time-varying linear state-space model to be used in predictive control:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \quad k \in [1, P] && (8.33) \\ y(k) &= Cx(k) + d(k) \end{aligned}$$

where:

$$y(k) = \begin{bmatrix} y_L(k) \\ y_{NL}(k) \end{bmatrix}, \quad C = \begin{bmatrix} C_L \\ C_{NL,k} \end{bmatrix}, \quad d_k = \begin{bmatrix} d_{L,k} \\ d_{NL,k} \end{bmatrix}$$

The time–varying model in equation (8.33) is used to predict the future dynamic behavior of the plant while the same steps of the cost function and the other matrices required for the QP optimization problem from the previous LMPC algorithm are used here too.

The neural network model is trained and tested using the `train` and `sim` functions of the MATLAB/Neural Network Toolbox while the MATLAB MAD Toolbox [AMR00] is used as the AD tool for the local linearization of the PGAS nonlinear model.

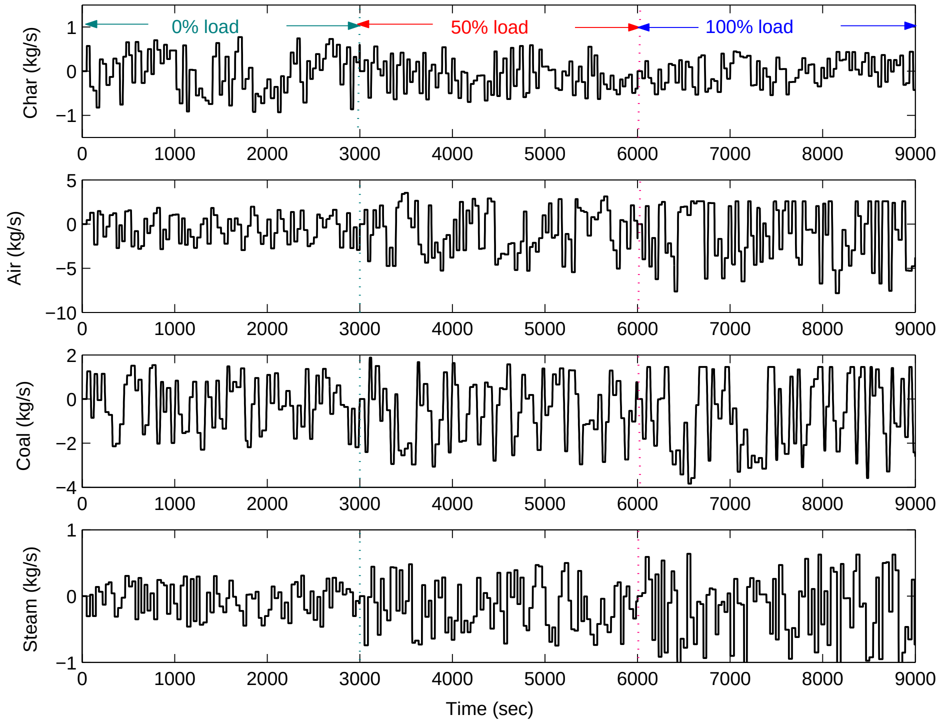

Figure 8.14: Input data for training set

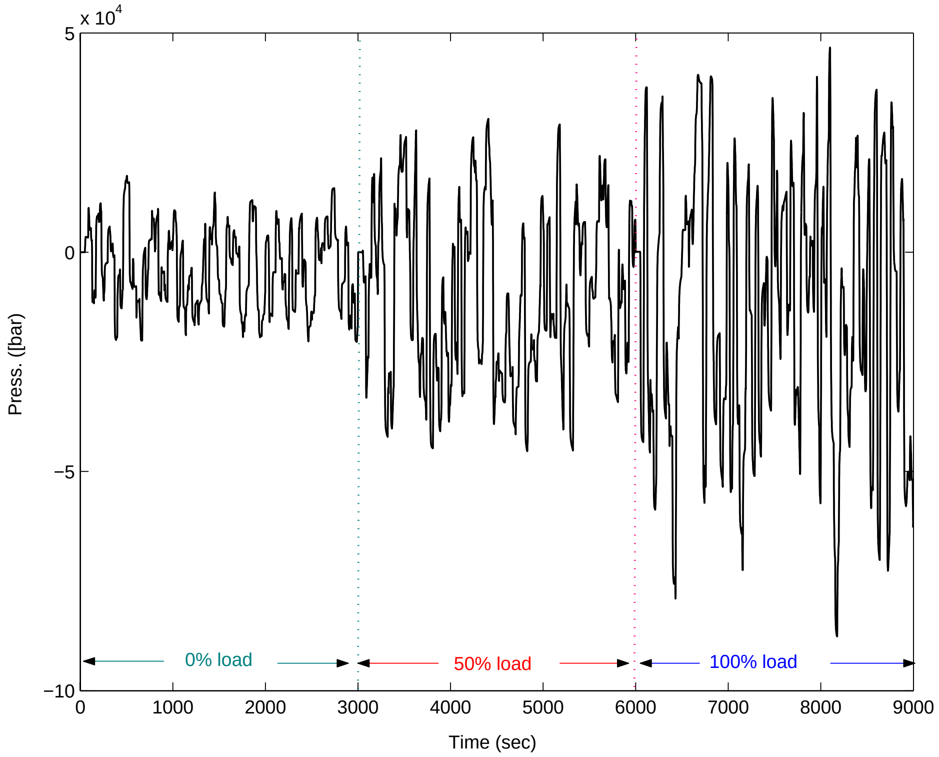

Figure 8.15: Output data for training set

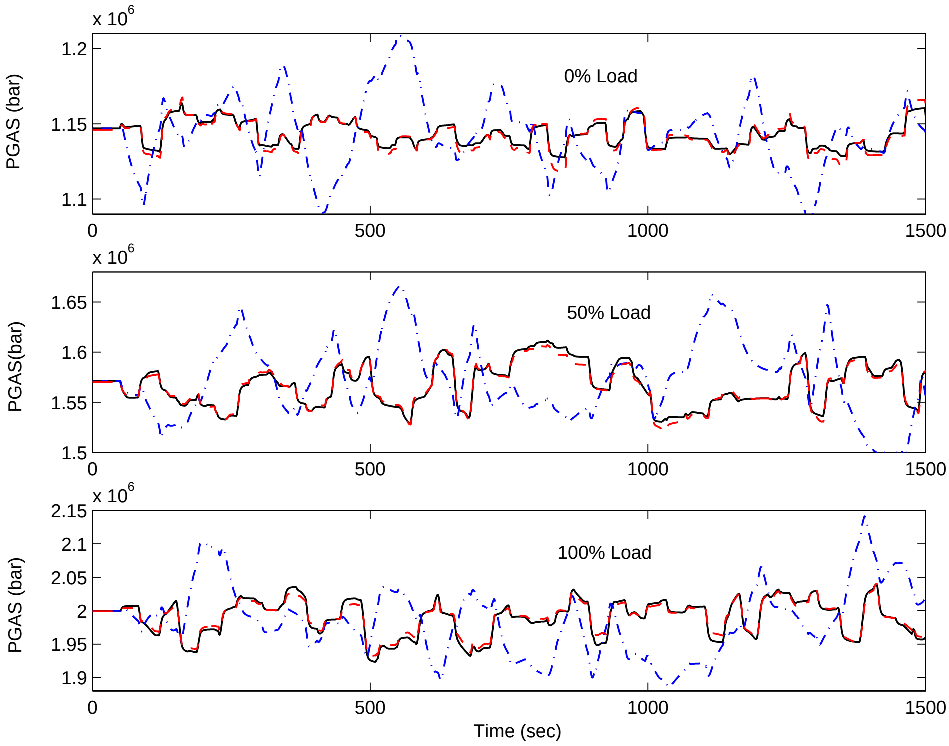

Figure 8.16: Gas pressure (PGAS) output response; actual (solid lines); linear model (dot–dashed lines); Wiener model (dashed lines).

8.5.2 Simulation Results

***PGAS Wiener model identification*:**

The same dynamic matrices A and B of the previous internal model in the LMPC section are used in the linear dynamic part of the PGAS Wiener model. For the nonlinear static part of the model, the number of nodes in the first and second hidden layers of the neural network was 16 and 10 respectively with one node in the output layer. Data were generated through applying a zero mean normalized sequence of random pulses with their periods and amplitudes corresponding to the maximum and minimum expected variations and frequency in response to individual input change under different load conditions. The input and output data sets over different loads were then linked together and used in training and validation of the ANN (see Figures. 8.14 to 8.15). The performance of the trained PGAS Wiener model at the three load conditions are given in Figures 8.17 to 8.16, which show the model is capable of capturing most of the characteristic behavior of the plant.

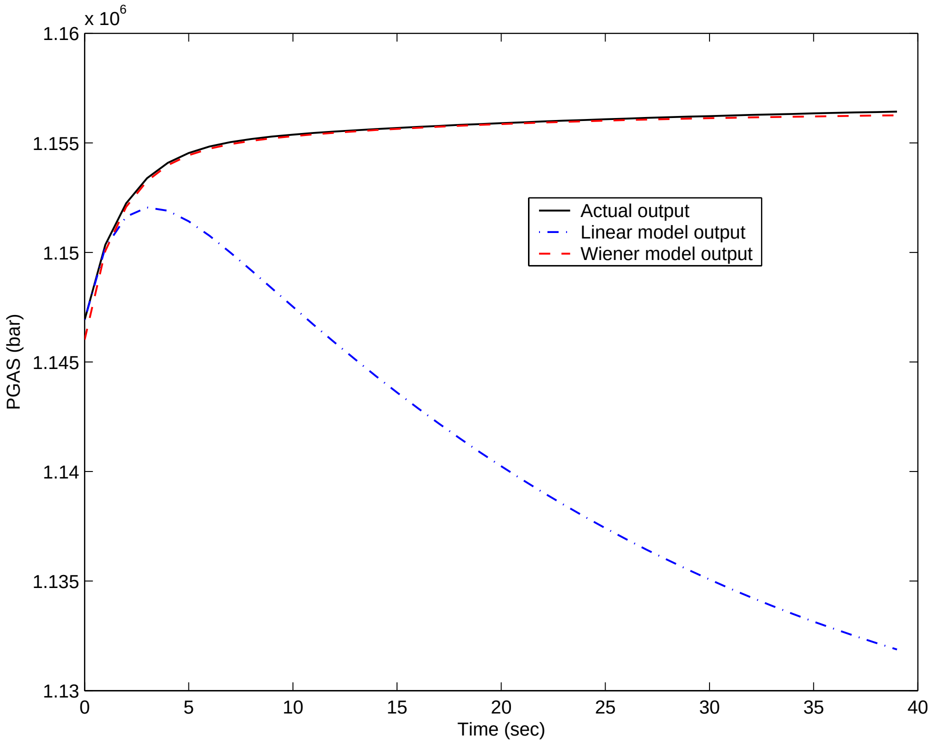

Figure 8.17: Open-loop step response of PGAS

2- ***NMPC parameters tuning*:**

In this case, $M = 7$ s, $P = 17$ s are chosen to get a stable and good control performance without any constraints violation.

The weighting matrices Q_0, and R_0 are set to the following values after online tuning;

$$Q_0 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 150 & 0 & 0 \\ 0 & 0 & 3.5 & 0 \\ 0 & 0 & 0 & 2 \times 10^6 \end{bmatrix}, \quad R_0 = \begin{bmatrix} 10^4 & 0 & 0 & 0 \\ 0 & 5 \times 10^2 & 0 & 0 \\ 0 & 0 & 5 \times 10^2 & 0 \\ 0 & 0 & 0 & 10^3 \end{bmatrix}$$

Using the above configuration, nominal stability was achieved at all three load conditions. That is the magnitudes of all eigenvalues of $A - BK$ are less than 1.

3- ***PSINK Disturbance Test*:**

All the results to follow are compared with the linear MPC. The maximum and minimum values as well as the peak rate change of the input variables of two disturbance tests under different load conditions are shown in Table 8.4. The maximum absolute error between output variables and their setpoints and the integral of absolute error (IAE) of these variables are calculated in Table 8.3. Figs. 8.18 to 8.23 show the system performance at 50% and 0% load conditions. In the step disturbance test, the results are plotted for $t \leq 100$ s to present the control performance in more detail. After this time period, all the output responses remained constant. The results in Table 8.4 and 8.3 however are calculated until t=300 s.

For 0% load sinusoidal test, the results with extra simulation time (until t=600 s) are provided to confirm the satisfactory performance of output in meeting the given specifications. The results show that both controllers are capable of maintaining the output variables within the limits for the tests specified by ALSTOM. In the case of the step disturbance test, an improvement in the whole system performance was observed using the NMPC approach. In fact, all output variables have benefited from using a more accurate PGAS internal model. Due to multivariable interactions, the improvement in other output variables sometimes is even larger than that in PGAS itself (see Figures 8.18 and 8.20). This is explained as follows. The response of PGAS, particularly to disturbance PSINK is much faster than other output variables (Figure 8.5). The improvement of nonlinear model is mainly in long term prediction (Figure 8.13). Hence, it has more effect on slow-response variables rather than PGAS, which is a fast-response variable. Moreover, the maximum drop of PGAS in the step disturbance test is the response to disturbance before the controller can take action, hence is not able to be reduced by changing the internal model only.

However, in the sine disturbance test, the results of 50% and 100% load conditions using the NMPC become less favorable compared to the LMPC results, but stay within the allowed range. Further improvements in the CVGAS and TGAS responses at 0% load are observed in the same test (see Tables 8.3). The remedy for this behavior could be in further training of the FFNN to handle sinusoidal inputs. It is recalled here that the training data is collected for all the three load conditions and not only that for 0% load. A fully nonlinear model that can handle all load conditions might also prove useful for solving this behavior.

4- *Load Change Test*:

The plant response in this test is compared with the results of LMPC controller. For both controllers, good setpoint tracking performance is obtained (see Figs. 8.9 to 8.8). The output results show approximately similar behaviors for the two controllers, with a small improvement in the MASS response when using the NMPC approach. Also, the manipulated variables response is smoother in this case as shown in Figure 8.9.

8.6 Summary

Two predictive controllers have been developed to control the ALSTOM gasifier benchmark process. The first controller was LMPC, then a NMPC approach was developed by modifying the LMPC algorithm to include a partially nonlinear internal model. In the linear predictive controller case, a linear state space model identified at 0% load condition is chosen as the internal model while a QP routine is used to solve the online optimization problem. The controller is able to achieve all required performance specifications within input and output constraints. The linear controller was then extended to a nonlinear one in order to improve the system performance further. In this case, the QP routine is used as the optimizer and a time-varying linear state-space model is adopted for process response prediction. A nonlinear model represented by a feed forward neural network is created for one of the four outputs (PGAS) while a linear model corresponding to 0% load condition is adopted for the other output variables. This selection is based on open loop response comparison between the linear model and plant responses. A static nonlinear model of PGAS is identified using FFNN. The data generated by the plant model at the three operating points is used to train and validate the neural network. To regain the convex feature of the optimization problem, the ANN model was linearized at every sampling time to update the linear model used for optimization. Thus, the internal model is in effect a linear time-varying model. The new controller meets all the required performance specifications within given input and output constraints during sink pressure disturbance and load change tests and the results show a significant improvement in the system performance compared with the results obtained when only linear time-invariant model is used. The final conclusion of this work is that sometimes a simple controller might be able to control a complex plant. The gasifier control using fully nonlinear plant and nonlinear optimizer is not used here because of the limitation of the time scale and it will be our future work.

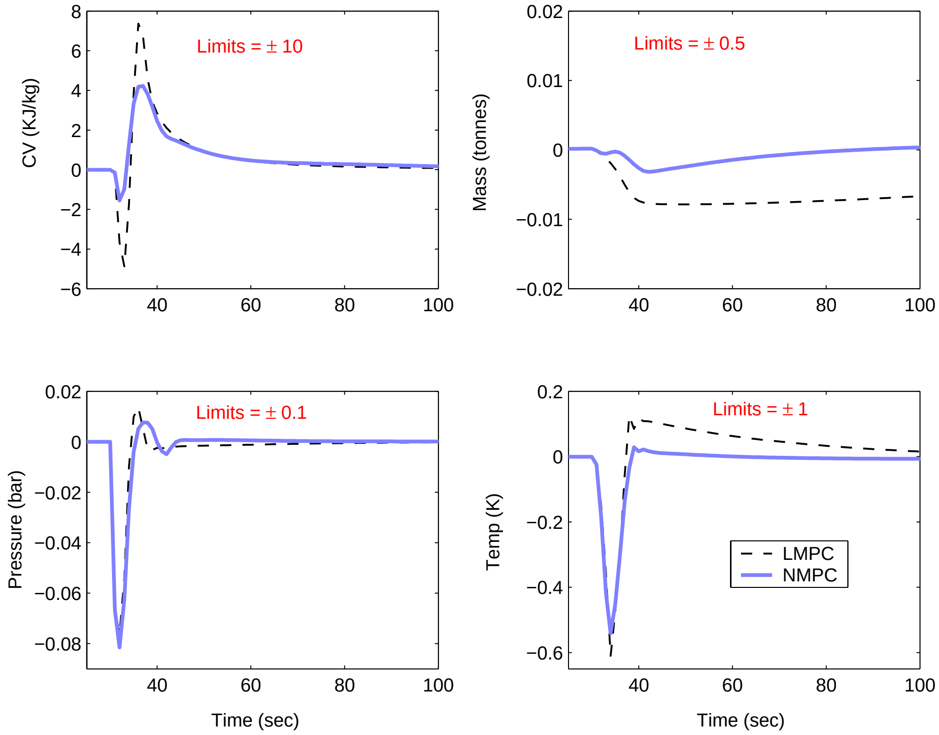

Figure 8.18: Output response at 50% load condition

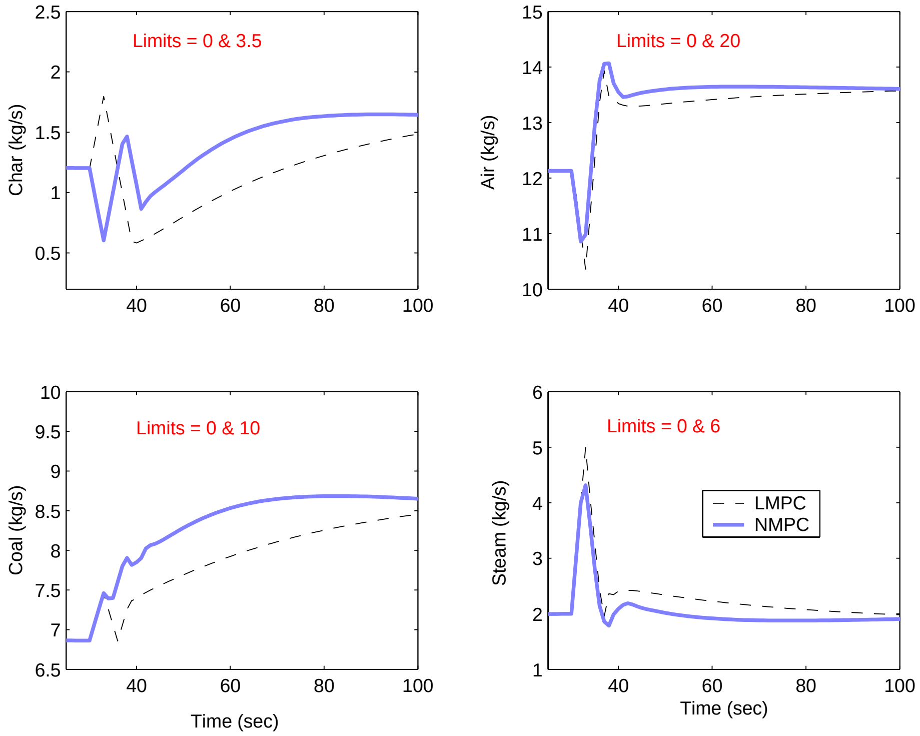

Figure 8.19: Input response at 50% load condition

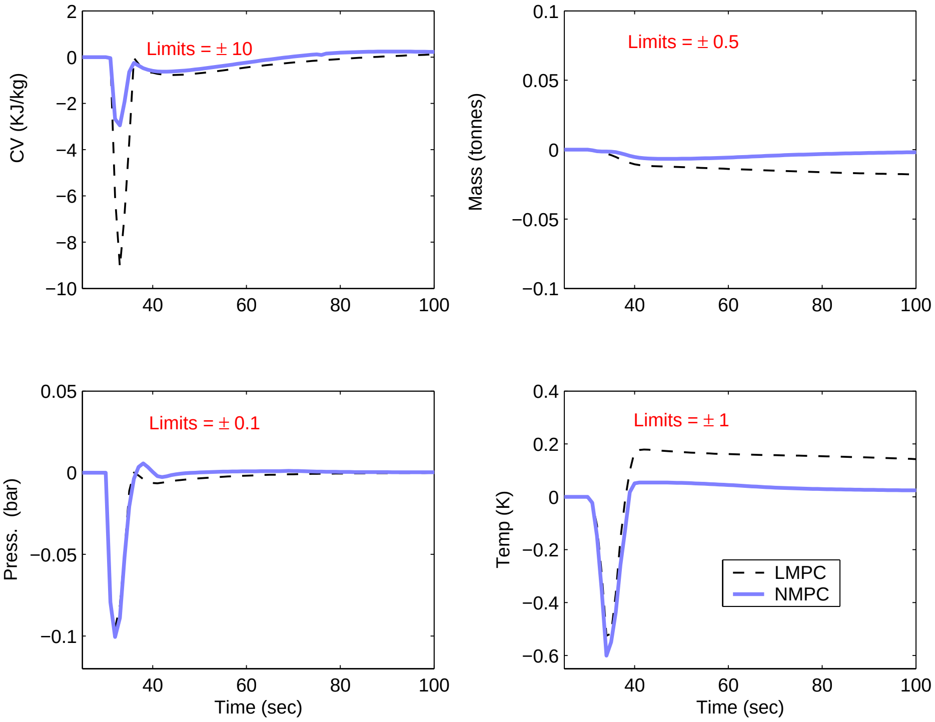

Figure 8.20: Output response at 0% load condition

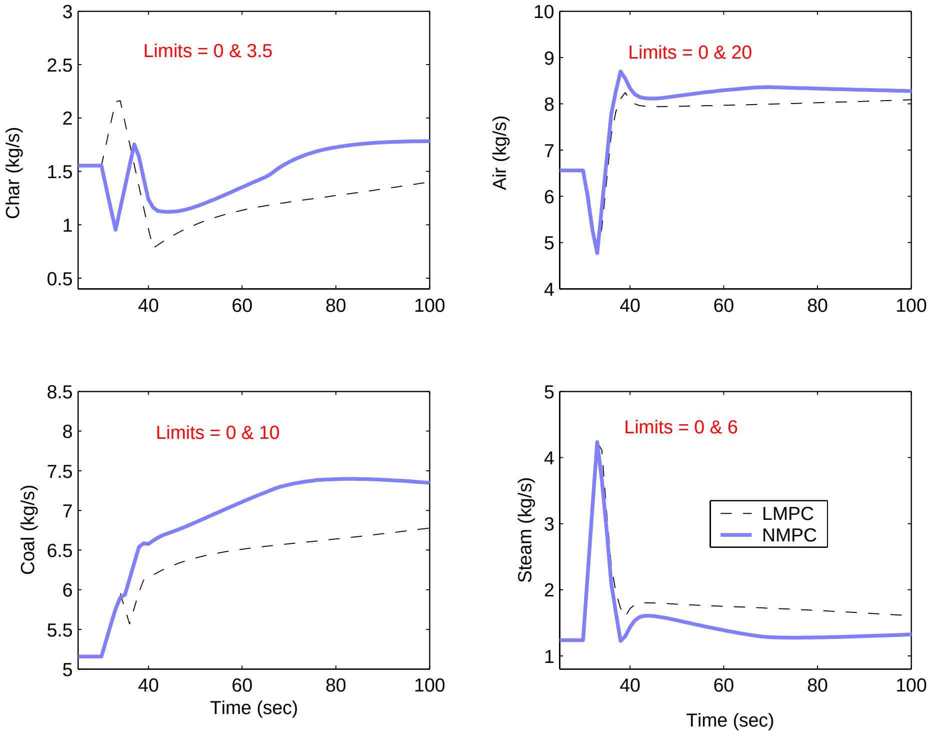

Figure 8.21: Input response at 0% load condition

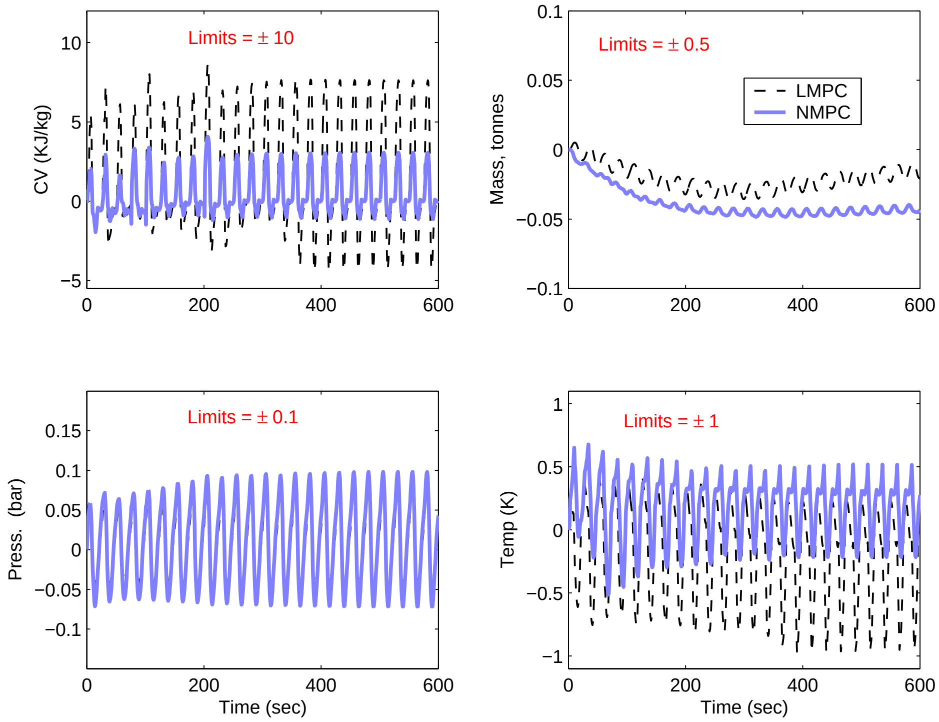

Figure 8.22: Output response at 0% load condition

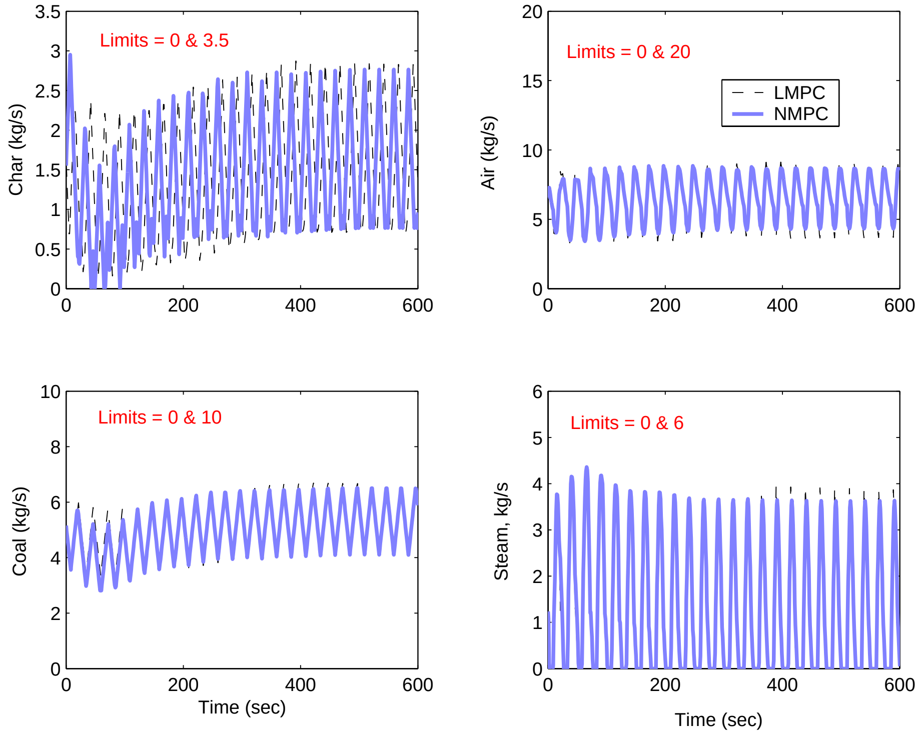

Figure 8.23: Input response at 0% load condition

Table 8.3: Output results

Step, 100% load	Maximum Absolute Error		IAE	
Output	LMPC	NMPC	LMPC	NMPC
CVGAS	7.685	5.6659	90.975	84.568
MASS	12.915	10.4678	2328	1642
PGAS	0.0674	0.0735	0.251	0.2761
TGAS	0.529	0.5104	10.348	7.259
Step, 50% load	Maximum Absolute Error		IAE	
Output	LMPC	NMPC	LMPC	NMPC
CVGAS	7.370	4.2325	75.002	66.515
MASS	8.046	3.3512	1210	125.717
PGAS	0.076	0.0816	0.308	0.3177
TGAS	0.610	0.5406	7.304	3.937
Step, 0% load	Maximum Absolute Error		IAE	
Output	LMPC	NMPC	LMPC	NMPC
CVGAS	9.084	2.9471	86.041	49.2873
MASS	18.504	6.6339	4050	557.149
PGAS	0.095	0.1006	0.458	0.4490
TGAS	0.525	0.6005	29.347	8.1648
Sine, 100% load	Maximum Absolute Error		IAE	
Output	LMPC	NMPC	LMPC	NMPC
CVGAS	5.1753	5.5628	890.19	898.96
MASS	2.3263	5.1133	318.004	865.246
PGAS	0.0308	0.03615	5.0379	5.241
TGAS	0.1819	0.3802	36.899	51.421
Sine, 50% load	Maximum Absolute Error		IAE	
Output	LMPC	NMPC	LMPC	NMPC
CVGAS	4.3678	4.7659	725.15	764.8269
MASS	4.3017	8.0425	784.93	1406
PGAS	0.03259	0.04015	5.698	6.471
TGAS	0.22628	0.47371	49.3625	68.679
Sine, 0% load	Maximum Absolute Error		IAE	
Output	LMPC	NMPC	LMPC	NMPC
CVGAS	7.8471	3.6711	646.07	256.71
MASS	33.918	45.694	5699	9853
PGAS	0.0962	0.0960	10.515	13.58
TGAS	0.7749	0.6574	103.57	85.558

Table 8.4: Input results

Step, 100% load	Maximum		Minimum		Peak Rate	
Input	LMPC	NMPC	LMPC	NMPC	LMPC	NMPC
WCHR	1.585	1.3664	0.435	0.5498	0.2	0.2
WAIR	19.071	19.3215	16.136	16.158	1.0	1.0
WCOL	10.00	10.00	8.619	8.6153	0.2	0.2
WSTM	5.109	4.6969	2.531	2.3675	1.0	1.0
Step, 50% load	Maximum		Minimum		Peak Rate	
Input	LMPC	NMPC	LMPC	NMPC	LMPC	NMPC
WCHR	1.796	1.6831	0.583	0.6026	0.2	0.2
WAIR	13.934	14.0667	10.360	10.8571	1.0	1.0
WCOL	8.653	8.6843	6.845	6.839	0.2	0.2
WSTM	4.992	4.3168	1.897	1.7868	1.0	1.0
Step, 0% load	Maximum		Minimum		Peak Rate	
WCHR	2.1534	2.0469	0.2274	0.9536	0.2	0.2
WAIR	8.5508	8.7148	4.7147	4.7708	1.0	1.0
WCOL	7.7216	7.6251	5.1574	5.1398	0.2	0.2
WSTM	4.2366	4.2365	1.0404	1.1226	1.0	1.0
Sine, 100% load	Maximum		Minimum		Peak Rate	
WCHR	1.4640	1.725	0.3492	0.0742	0.2	0.2
WAIR	18.923	19.049	15.747	15.628	0.577	0.671
WCOL	9.7537	9.7444	7.2684	7.0443	0.2	0.2
WSTM	3.6452	3.7488	1.5192	1.5676	0.603	0.610
Sine, 50% load	Maximum		Minimum		Peak Rate	
WCHR	1.9503	1.8531	0.11527	0.21605	0.2	0.2
WAIR	13.764	14.014	10.331	9.9657	0.623	0.7461
WCOL	8.1287	8.1573	7.2684	7.0443	0.2	0.2
WSTM	3.3492	3.548	0.6553	0.3868	0.678	0.5866
Sine, 0% load	Maximum		Minimum		Peak Rate	
WCHR	2.3547	2.8559	0.12867	0	0.2	0.2
WAIR	8.905	8.9219	3.2579	3.4465	0.623	0.7461
WCOL	6.1213	6.4590	3.2334	2.860	0.2	0.2
WSTM	3.8088	4.3271	0	0	0.678	0.5866

Chapter 9

Conclusion and Recommendations

9.1 General Conclusions

This section complements the various comments given as appropriately on the results as they were obtained. In general the following points can be stated.

1. Two new algorithms for computationally efficient nonlinear model predictive control are developed in this thesis. Based on a nonlinear least square optimization problem, efficient algorithms to calculate the gradient information to solve the NLP are derived.
2. An approach is developed for the dynamic sensitivity calculation which are required to solve the online optimization problem of NMPC. A first–order approximation is introduced to simplify the dynamic sensitivity equations by using AD tool so that the computation efficiency is improved.
3. The existing approach of using high–order Taylor series expansion and AD to solve the model ODEs together with the sensitivity equations developed in [Cao05], is further developed to included state estimate stage and integrated disturbances models for offset free response and applied to a new test case.
4. An efficient algorithm has been proposed to train continuous-time recurrent neural networks to approximate nonlinear dynamic systems so that the trained network can be used as the internal model for a nonlinear predictive controller. The new training algorithm is based on the efficient Levenberge Marquardt method combined with an efficient and accurate tool of automatic differentiation. The dynamic sensitivity equations and the ODEs of the recurrent neural

network are solved accurately and simultaneously using high–order Taylor series and AD. Higher efficiency to solve sensitivity equations with a higher accuracy are obtained using the new algorithm compared with a traditional method. The trained networks work well with different model orders showing the capability to approximate a multivariable nonlinear plants. Different sampling time can be used with the trained model without the need to re-train the networks. The results show that, the choice of the network order is important to obtain good model fitting and stable performance.

5. Based on the identified neural network model, a NMPC controller is developed. The same strategy used in the network training has been used to solve the online optimization problem of the predictive controller. The capability of the new nonlinear identification algorithm and NMPC algorithm are tested in both the evaporator and the two–CSTR case studies with good results.

6. The controllers are demonstrated to be robust and stable for the various nonlinear MIMO processes studied in this thesis. The structure used makes it easy to incorporate variable constraints as well as large changes in the setpoint of operation of the process under control. The controller worked well with model mismatch experiments and with the presence of measurement noise, random nonzero mean or sustained unmeasured disturbances. Using a nonlinear least square optimizer in the controller is proved very effective and there has been no instances of unfeasible solutions in any of the case studies reported. It is not necessary for the case studies to included any terminal penalties on the optimizer.

7. The proposed controllers are shown to work well with first–principle as well as black–box process models. The black–box model can be used to replace the first–principle models in NMPC if the latter is too complicated to be incorporated in the controller.

8. The use of the automatic differentiation tool has remarkably reduced the computation time as shown by the results. The use of AD tool reduces the total calculation time by reducing the number of function evaluations and by producing values for the function and its partial derivatives that are of very high accuracy leading to more conclusive optimizing searches.

9. The predictive controller is formulated for state–space models. It is therefore necessary to have an accurate estimate of the states at every time step. The states are estimated from plant measurements and the main controller is augmented by an extended Kalman filter to evaluate the states at the start of every

equation solving step. This is necessary only for processes that contain hidden states. The work of the extended Kalman filter was made more practical and efficient using the AD tool on a locally linearized process model.

10. The controller was also augmented with a nonlinear feedback error integral function. This addition is needed to eliminate unwanted offset in the output resulting from plant/model mismatch due to the presence of sustained unmeasured disturbances, random disturbances of nonzero mean, measurement noise, and modelling errors. This new error integrator saturates at high offsets and large set point changes until these are brought to a small level by the controller then it works linearly on the remaining offset. It has been observed that it is not possible to get offset free control in many cases without the help of this error integrator. Furthermore, the EKF can be removed in some cases when this integrator is used and replaced with a simple state estimator. The offset remover algorithm worked by looking for a match to the unmeasured disturbance and shifting the model outputs or inputs to new levels. The output disturbance integrator is more effective than the input disturbance integrator as there are sometimes not enough disturbance inputs to cover all the offset sources. Nonlinear integration is used here to improve the transient behavior of the process outputs during offset error rejection time.

11. The controller for the first case study, the evaporator did not need any stated estimation as there were no unmeasured states in this case. The controller for the two–CSTR plant in the two control configurations S1 and S2 on the other hand needed EKF as the states are not all measurable. In both cases, two controllers were designed with good success to handle the first–principle and the black box model generated using the CTRNN for these case studies.

12. A simpler predictive controller has been developed to control the ALSTOM gasifier benchmark process. LMPC employing a GPC strategy modified to include a partial nonlinear internal model. A nonlinear class Wiener model is used to identify one of the process output variables (PGAS) which has strong nonlinearity while a linear model at 0% load condition is adopted for the other output variables. This nonlinear model for PGAS was linearized at every sampling time to update the linear model used for optimization. Thus, the internal model becomes a linear time-varying model. This is a novel approach which shows that it is not always needed to turn the internal model to full nonlinearity if a linear version can be made to work on some of the variables as in this case study. The new controller meets all the required performance specifications within given input and output constraints during various tests and the results

show a significant improvement in the system performance compared with the results obtained when only a linear time-invariant model is used.

9.2 Recommendations for Future Work

At the start of this project there were no publications connecting model predictive control and automatic differentiation and few in nonlinear model predictive control. The interest has grown very rapidly in the subject and more ideas are being generated in the literature on how to improve the controller and to make it faster. Building on the experience gained in this project and the new research by others, one can put forward the following points for future advancement in the subject;

1. The present work relied fully on ready packages connected together for achieving the present results. One useful procedure to save time is therefore to be able to combine the various routines in one which will no doubt cut some of the call time between the various procedures. One important saving can be made for example if the generated machine code for the AD of a particular problem can be compiled and saved to be used as a single procedure rather than calling the whole AD software every time it is needed.

2. The continuous–time recurrent neural network proved to be a very effective nonlinear modelling tool, and with the present method of training, it is recommended to be used in more applications. Due to lack of time, the third case study could have been modelled using this type of network. Not that the present procedure is not defective, but such representation is likely to make the model more effective for similar problems with perhaps stronger nonlinearities.

3. There has been a number of research monographs advocating a terminal penalty or/and terminal region that are used to lead the optimizer to ensure stability and feasibility in highly nonlinear problems. Although the formulation can be included in the controller routines in this work, this is not done as no cases of instability or infeasibility were met. This is not to say that such need will not appear in some other applications. It is therefore recommended that new case studies of varying degrees of complexity are attempted to asses the need for such terminal regions and corresponding terminal penalty in the objective functions.

4. The use of an effective modelling tool as the CTRNN makes it attractive to study the question of how much simplification is allowed in the model to produce

effective NMPC. It is clear that a large time saving can be made if the model structure is simpler, which might not represent the plant completely, but can nevertheless produce an effective controller.

5. There is a possibility of splitting the controller into outer and inner control loops both of the predictive control type. The main use of this type is in cases where there are short time as well as long time objectives for the process. In this case the inside loop will take care of the short term objective which could be done using a simpler and faster loop, and the outer loop to take care of the longer time objective which could use a more sophisticated model but with more time available for doing the more demanding calculations.

6. The proposed offset removal approaches have two parameters that need to be tuned. In this thesis online tuning is used for this task. It will be better if an optimal off-line solution is attempted to find the best values of these parameters.

7. The strategy of using high–order Taylor series and AD tool for solve simultaneously the model ODEs and dynamic sensitivity equation solution used in the proposed CTRNN training and the NMPC algorithm can be extended to solve the NLP for the more advanced state estimate, *i.e.* moving horizon state estimate, which can be then used with the proposed NMPC algorithm instead of EKF for cases when simple state updates become inadequate.

Bibliography

[AB03] Aufderheide B. and Bequette B. W., “Extension of dynamic matrix control to multiple models”, *Computers Chem. Eng.*, vol. 27, pp. 1079–1096, 2003.

[ACY04] Al Seyab R. K., Cao Y., and Yang S. H., “A case study of predictive control for the ALSTOM gasifier problem”, *Control 2004 Conference*, Bath University, 2004.

[ACY06] Al Seyab R. K., Cao Y., and Yang S. H., “A case study of predictive control for the ALSTOM gasifier problem”, *IEE Proceedings – Control and Applications*, vol. 153(3), pp. 293–301, 2006.

[Adv99] “Advanced Continuous Simulation Language”, *Mitchell and Gauthier Associates Inc., Concord, Massachusetts*, 1999 .

[AKC96] Al–Duwaish H., Karim M. N., and Chandrasekar V., “Use of multilayer feed forward neural networks in identification and control of Wiener model”, *IEE, Proc. Control Theory Appl.*, vol. 143(3), pp. 255–258, 1996.

[ANK01] Armstrong B., Neevel D., and Kusik T., “New results in NPID control: tracking, integral control, friction, compensation, and experimental results”, *IEEE Trans. Contr. Syst. Techn.*, vol. 9(2), pp. 399–406, 2001.

[ANL] A collection of automatic differentiation tools. http://www–c.mcs.anl.gov/Projects/autodiff/AD–Tools/index.html

[AMR00] Shaun Forth, AMOR Group “MAD a MATLAB automatic differentiation Toolbox”, *Engineering System Department, Cranfield University (RMCS Shrivenham)*,UK, 2001.

[ASC05a] Al Seyab R. K. and Cao Y., “ Nonlinear model predictive control for the ALSTOM gasifier benchmark problem”, *IFAC 16th Congress*, 2005.

[ASC05b] Al Seyab R. K. and Cao Y., " Nonlinear model predictive control for the ALSTOM gasifier benchmark problem", Accepted for publication in: *Journal of Process Control*, 2005.

[ASC05c] Al Seyab R. K. and Cao Y., "Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation", Submitted for publication in: *IEEE TNN Special Issue on Feedback Control*, Septmeber, 2005.

[ASC06] Al Seyab R. K. and Cao Y., "Differential recurrent neural networks based predictive control", Accepted in: *9th International Symposium on Process System Engineering and 16th European Symposium on Computer Aided Process Engineering (ESCAPE/PSE2006) Conference*, to be held at 9–14 July 2006, Germany.

[AVA01] Abrahantes M. A., Vazquez, and Agamennoni O. E., "Approximate models for nonlinear dynamical systems and their generalization properties", *Math. Comput. Modell.*, vol. 33, pp. 965–986, 2001.

[Beq91a] Bequette B. W., "Nonlinear predictive control using multirate sampling", *Can. J. Chem. Eng.*, vol. 69, pp. 136–151,1991.

[Beq91b] Bequette B. W., "Nonlinear control of chemical processes: A review", *Ind. Eng. Chem. Res.*, vol. 30, pp. 1391–1413,1991.

[BGH92] Bischof C., Griewank A., Hovland P., "ADIFOR: generating derivative codes from Fortran programs", *Scientific Programming*, vol. 1(1), pp. 1–29, 1992.

[BGW90] Bitmead R. R., Gevers M., Wertz V., "Adaptive optimal control–the thinking man's GPC", *Prentice–Hall, Englewood Cliffs*, 1990.

[Bie98] Biegler L. T. "Advance in nonlinear programming concepts for process control", *Journal of Process Control*, vol. 8(5), pp. 301–311, 1998.

[Bin01] Binder T., Blank L., Bock H., Bulirsch R., Dahmen W., Diehl M., Kronseder T., Marquardt W., Scholoder J., Stryk O., "Introduction to model based optimization of chemical processes on moving horizons", in: M. Grötschel, S. Krumke, J. Rambau (Eds.), "Online optimization of large scale systems: state of art", Springer, pp. 295–340, 2001.

[Bis96] Bischof C., Jones W., Mauer A., Samareh J., "Experiences with the application of the ADIC automatic differentiation tool to the CSCMDO 3–D volume grid generation code", *Proceedings of the 34th AIAA Aerospace*

Sciences Meeting, AIAA 96–0716, American Institute of Aeronautics and Astronomics, 1996.

[BM90] Behat N. and McAvoy T. J., "Use of neural nets for dynamic modeling and control of chemical process systems", *Computers Chem. Engng.*, vol. 14(415), pp. 573–583, 1990.

[BQ01] Badgwell T. A. and Qin S. J., "Review of nonlinear model predictive control applications, chapter 1", In: Kouvaritakis B. and Cannon M., Nonlinear predictive control: theory and practice, *IEE Control Engineering Series 61, The Institution of Electrical Engineers*, London, 2001.

[BS89] Brengle D. D. and Seider W. D., "Multistep nonlinear predictive controller", *Ind. Eng. Chem. Res.*, vol. 28, pp. 1812–1822, 1989.

[BLR02] Bucker H. M., Lang B., Rasch A., Bisch C. H., "Computing sensitivity of electrostatic potential by automatic differentiation", *Computer Physics Communications*, vol. 147, pp. 720–723, 2002.

[BVS86] Billings S. A. and Voon W. S. F., "Correlation based model validity tests for nonlinear models", *Int. J. Control*, vol. 44, pp. 235–244, 1986.

[CA98] Chen H. and Allgower F., "A Quasi–infinite horizon nonlinear model predictive control scheme with guaranteed stability", *Automatica*, vol. 34, pp. 1205–1217, 1998.

[CAF03] Cervantes A. L., Agamennoni O. A., Figueroa J. L., "A nonlinear model predictive control systems based on Wiener piecewise linear models", *J. Proc. Contr.*, vol. 13(7), pp. 655–666, 2003.

[Cao95] Cao Y., "Control structure selection for chemical process using input–output controllability analysis", *PhD Thesis, University of Exeter*, 1995.

[Cao05] Cao Y.,"A formulation of nonlinear model predictive control using automatic differentiation", *Journal of Process Control*, vol. 15, pp. 851–858, 2005.

[CAS03] Cao Y., and Al Seyab R. K., "Nonlinear model predictive control using automatic differentiation", *European Control Conference (ECC 2003)*, Cambridge, UK, 2003, in CDROM.

[CB96] Cao Y. and Biss D., "An extension of singular value analysis for assessing manipulated variable constraints", *J. Process Control*, vol. 6(1), pp. 37–48, 1996.

[CBG] Chen W. H., Ballance D. J., and Gawthrop P. J., "Nonlinear PID predictive controller", *CSC Report: CSC–99010, Center for Systems and Control*, Dep. Electronics and Electical Eng., Univ. of Glasgow, UK, 1999.

[CJ87] Chen C. Y. and Joeseph B., "On-line optimization using a two–phase approach: an application study", *Ind. Eng. Chem. Res.*, vol. 26, pp. 1924–1930, 1987.

[CH88] Cutler C. R. and Hawkins R. B., "Application of a large predictive multivariable controller to a hydrocracker second stage reactor", *Proc. Automatic Control Conf.*, Atlanta, GA; pp. 284-291, 1988.

[ChC98] Chen C. T., "Linear system theory and design", *Oxford University Press, New York, third edition*, 1998.

[ChG88] Char B. W., Geddes K. O., Gonnet G. H., Monagan M. B., Watt S. M., "MAPLE reference manual", *Watcom Publications*, Waterloo, Ontario, Canada, 1988.

[ChJ98] Chen J., "Systematic derivations of model predictive control based on artificial neural networks", *Chemical Eng. Communications*, vol. 164, pp. 35–39, 1998.

[Chr92] Christianson B., "Reverse accumulation and accurate rounding error estimators for Taylor series", *Optimization Methods and Software*, vol. 1, pp. 81–94, 1992.

[ChS94] Chinchalkar S., "The application of automatic differentiation to problem in engineering", *Computer Methods in Applied Mechanics and Engineering*, vol. 118, pp. 197–207, 1994.

[ChW04] Chen W. S., Bakshi B. R., Goel P., Ungarala S., "Bayesian estimation via sequential Monte Carlo sampling: unconstrained nonlinear dynamic systems", *Ind. Eng. Chem. Res.*, vol. 43, pp. 4012–4025, 2004.

[Clg58] Clegg J. C., "A nonlinear integrator for servomechanisms", 1958. In cite: Sheng A., Satoshi N., Yamaguchi S., Itakura H., "A new nonlinear integrator with positive phase shifts", *IEICE Trans. Fundamentals*, vol. E81–A(1), pp. 197–201, 1998.

[CMT87] Clarke D. W., Mohtadi C., and Tuffs P. S., "Generalized predictive control–1, the basic algorithm", *Automatica*, vol. 23, pp. 137-148, 1987.

[Com94] Compbell S. L., “Utilization of automatic differentiation in control algorithm”, *IEEE Trans. on Automatic Control*, vol. 39(5), pp. 1047–1052, 1994.

[CR98] Campbell J. C. and Rawlings J. B., “Predictive control of sheet and film–forming processes”, *AIChE J.*, vol. 44, pp. 1713–1723, 1998.

[CS85] Caracotsios M. and Stewart W. E., “Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations”, *Comput. Chem. Eng.*, vol. 9(4), pp. 359–365, 1985.

[CuR80] Cutler C. R. and Ramaker B. L., “Dynamic matrix control: a computer control algorithm”, *Proc. Automatic Control Conf.*, San Francisco, CA, paper Wp5-B, 1980.

[CV97] Coleman T. F., Verma A., “ADMIT–1: Automatic Differentiation and MATLAB Interface Toolbox”, Technical Report CTC97TR271, 3/97, Cornell Theory Center, Dept. Comput. Science, Cornell University, 1997, available via http://simon.cs.cornell.edu/home/verma/AD/ADMIT.ps.

[CV98] Coleman T. F., Verma A., “ADMAT: An Automatic differentiation Toolbox for MATLAB” *Computer Science Department and Center for Applied Mathematics*, Cornell University, Ithaca, NY 14850, 1998.

[CVB00] Castro M. C., Vieira R. C., and Biscaia E. C., “Automatic differentiation tools in the dynamic simulation of chemical engineering processes”, *Barzilian J. of Chem. Eng.*, vol. 17(4), 2000.

[CW91] Cluett W. R. and Wang L., “Modelling and robust controller design using step response data”, *Chem. Eng. Science*, vol. 46(8), pp. 2065–2077, 1991.

[CXS94] Chu X. and Seborg D. E., “Nonlinear model predictive control based on Hammerstein models”, *Proc. Int. Symp. on Process Systems Engineering*, Seoul, Korea, pp. 995, 1994.

[CY04] Cao Y., and Yang Z., “Multiobjective process controllability analysis”, *Computer and Chemical Eng.*, vol. 28(1–2), pp. 83–90, 2004.

[DAS02] Dojouad R., Audiffren N., Sportisse B., “A sensitivity analysis study for RADM2 mechanism using automatic differentiation”, *Atmospheric Environment*, vol. 37, pp. 3029–3038, 2002.

[DB98] Demuth H. and Beal M., “Neural Network Toolbox: Users’s Guide, version 3.0”, *The MathWorks*, Inc., Natick, MA, 1998.

[Den77] Dennis Tr. J.E., “Nonlinear least squares”, In Jacobs D., editor, “State of the art in numerical analysis”, pp. 269–312, York, England, 1977, Academic Press.

[DER95] Draeger A., Engell S., and Ranke H., “Model predictive control using neural networks”, IEEE Control Systems, vol. 5, pp. 61–66, 1995.

[DFL94] Dumont G., Fu Y., and Lu G., “Nonlinear adaptive generalized predictive control and applications ”, in: Advances in model-based predictive control, *Oxford University Press*, Oxford, 1994.

[DiB02] Diehl M., Bock H. G., Schloder J. P., Findeisen R., Nagy Z., and Allgower F., “Real-time optimization and nonlinear model predictive control of processes governed by differential–algebraic equations”, *J. Process Control*, vol. 12, pp. 577-585, 2002.

[Dix99] Dixon R., “Advanced gasifier control”, *IEE Computing and Control Eng. J.*, vol. 10(3), pp. 93–96, 1999.

[Dix02] Dixon R., “ALSTOM benchmark challenge II: control of a non-linear gasifier model”, available from: http://www.iee.org/OnComms/PN/controlauto/Specification_v2.pdf, 2002.

[DKW95] Delgado A., Kambhampati C., Warwick K., “Dynamic recurrent neural network for system identification and control”, *IEE Proc. Control Theory Appl.*, vol. 142(4), pp. 307-314. 1995.

[DP04] Dixon R., Pike A., “Introduction to the 2^{nd} ALSTOM benchmark challenge on gasifier control”, *Control 2004, Bath, UK*, 2004.

[DPM00] Dixon R., Pike A., and M. Donne, “The ALSTOM benchmark challenge on gasifier control”, *Proc. Instn. Mech. Engrs. Part I, J. of Systems and Control Eng.*, vol. 214, pp. 389–394, 2000.

[EB99] Eberhard P. and Bischof C., “Automatic differentiation of numerical integration algorithms”, *Mathematics of Computation*, vol. 68(226), pp. 717–731, 1999.

[ECF02] Elizonda D., Cappelare B., Faure C., “Automatic versus manual model differentiation to compute sensitivities and solve nonlinear inverse problems”, *Computers and Geoseciences*, vol. 28, pp. 309–326, 2002.

[EMP86] Economou C. G., Morari M., and Plasson B. O., "Internal model control. 5. extension to nonlinear systems", *Ind. Eng. Chem. Process Des. Dev.*, vol. 25, pp. 403–411, 1986.

[ErG97] Eriksson J., Gulliksson M., Lindstrom P., Wedin P. A., "Regularization tools for training large feed–forward neural networks using automatic differentiation", *Optimization Methods and Software*, vol. 10, pp. 49–69, 1997.

[ER90] Eaton J. W. and Rawling J. B.,"Feedback control of nonlinear processes using online optimization techniques", *Comp. Chem. Engng*, vol. 14(4–5), pp. 469–479,1990.

[FiD01] Findeisen R., Diehl M., Nagy Z., Allgower F., Bock H. G., Schloder J. P., "Computational feasibility and performance of nonlinear model predictive control schemes", In: *Proceedings of European Control Conference (ECC 2001)*, 2001.

[Fin97] Findeisen R., "Suboptimal nonlinear model predictive control", *Msc. thesis, University of Wisconsin–Madison*, 1997.

[FnI03] Findeisen R., Imsland L., Allgower F., Foss A., "State and output feedback nonlinear model predictive control: an overview", *European Journal of Control*, vol. 9, pp. 190–206, 2003.

[FN93] Funahashi K. L. and Nakamura Y., "Approximation of dynamical systems by continues time recurrent neural networks", *Neural Networks*, vol. 6, pp. 183–192, 1993.

[FP98] Faure C. and Papegay Y., "Odyssee user's guide, version 1.7. Technical Report 0224", *INRIA, Unite de Recherche, INRIA, Sophia Antipolis, 2004 Route des Lucioles, B. P. 93, 06902, Sophia Antipolis Cedex*, France, 1998, available from: http://www.inria.fr/safir/SAM/Odyssee/odyssee.html.

[GA93] Grewal M. S., Andrews A. P., "Kalman filtering theory and practice", *Prentice-Hall, New Jersey*, 1993.

[GsA88] Giuseppe S. B. and Alexander S., "A variable–structure controller", *IEEE Trans*, vol. AC–33, 1988. In cite: Sheng A., Satoshi N., Yamaguchi S., Itakura H., "A new nonlinear integrator with positive phase shifts", *IEICE Trans. Fundamentals*, vol. E81–A(1), pp. 197–201, 1998.

[Gar84] Garcia C. E., "Quadratic dynamic matrix control of nonlinear processes, an application of batch reaction process", paper presented at 1984, *AIChE Annual Meeting*, San Fransisco, CA, 1984.

[GBP04] Gatley S. L., Bates D. G., Postlethwaite I., "H_∞ control and anti–windup compensation of the nonlinear ALSTOM gasifier model" *Control 2004, University of Bath*, UK, 2004.

[GDJ96] Griewank A., David J., Jean U., "ADOL-C: a package for the automatic differentiation of algorithms written in C/C++", *ACM*, vol. 22(2), pp. 131–167, 1996.

[GeJ00] Gerksic S., Juricic D., Strmcnik S., and Matko D., Wiener model based nonlinear predictive control, *Int. J. Sys. Science*, vol. 31, pp. 189–202, 2000.

[GeK82] Gelb A., Kasper J. F., Nash R. A., Price C. F., and Sutherland A. A., "Applied optimal estimation", *Cambridge, MA: MIT Press.*, 1982.

[Gie97] Giering R., "Tangent linear and adjoint model compiler: user manual version 1.2, TAMAC version 4.8", *Center for Global Change Sciences, Dept. Earth, Atomospheric, and Planetary Science, MIT*, Cambridge, MA 02139, USA, 1997.

[GJB04] Gomez J. C, Juntan A., and Baeyens E., "Wiener model identification and predictive control of a pH neutralisation process", *IEE Process Control Theory Applications*, vol. 151(3), pp. 329–338, 2004.

[GKW00] Garaces F., Kambhampati C., and Warwick K., "Dynamic recurrent neural networks for feedback linearization of a multivariable nonlinear evaporator systems", *Proceedings DYCONS'99*, World Scientific, 1999.

[GL00] Gregorcic G. and Lightbody G., "A comparison of multiple model and pole–placement self–tunning for the control of highly nonlinear processes", *In proceeding of the Irish Signals and Systems Conference*, pp. 303–311, 2000.

[GM82] Garcia C. E. and Morari M., "Internal model control: 1, unifying review and some new results", *Ind. Eng. Chem. Process Des. Dev.*, vol. 21, pp/ 308–323, 1982.

[Gol89] Goldberge D. E., "Genetic algorithms in search, optimization and machine learning", *Reading, MA:Addision–Wesley*, 1989.

[Gri89] Griewank A., "On automatic differentiation", In Iri M. and Tanabe K. (Eds.), "Mathematical programming: recent developments and applications", *Kluwer Academic Publishers*, pp. 83–108, 1989.

[Gri92] Griewank A., "Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation", *Optimization Methods and Software*, vol. 1, pp. 35–54, 1992.

[Gri95] Griewank A., "ODE solving via automatic differentiation and rational prediction", *in: Griffiths D., Watson G. (Eds.), Numerical Analysis 1995, vol. 344 of Pitman Research Nots in Mathematics Series, Addison–Wesley., Reading, MA*, 1995.

[Gri00] Griewank A., "Evaluating derivatives", *SIAM, Philadelphia, PA*, 2000.

[GW04] Griesse R., Walther A., "Evaluating gradients in optimal control: continuous adjoint versus automatic differentiation", *J. Optimization Theory and Applications*, vol. 122(1), pp. 63–86, 2004.

[GZ92] Gattu G. and Zafiriou E., "Nonlinear quadratic dynamic matrix control with state estimation", *Indus. Eng. Chem. Res.*, vol. 31, pp. 1096–1104, 1992.

[Hen98] Henson M. A., "Nonlinear model predictive control : current states and future directions", *Computer and Chem. Eng.*, vol. 23, pp. 187–202, 1998.

[HH93] Hush D. R. and Horne B. G., "Progress in supervised neural networks", *IEEE Sig. Process, Mag.*, vol. 1, pp. 8–39, 1993.

[HL90] Huberman B. A. and Lumer E., "Dynamics of adaptive systems", *IEEE Tranc. on Circuits and Systems*, vol. 37(4), 1990.

[HPB94] de Hennin S., Perkins J.D., and Batron G.W., "Structural decision in on–line optimization", *Proceeding of the Fifth International Symposium on Process System Engineering*, Kyongiu, South Korea, pp. 297–302, 1994.

[HR99] Hu Q. P., and Rangaiah G. P., "Adaptive internal model control of nonlinear processes", *Chem. Eng. Science*, vol. 54, pp. 1205–1220, 1999.

[HS97] Henson M. A. and Seborg D. E., "Nonlinear process control", *Prentice–Hall PTR*, New Jersey, 1997.

[HuG02] Huzmezan M., Gough W. A., Doumont G. A., Kovac S., "Time delay integrating systems: a challenge for process control industries, a practical solution", *Control Eng. Practic*, vol. 10, pp. 1153–1161, 2002.

[HuS93] Hunt K. J., Sbarbaro D., Zbikowski R., Gawthrop P. J., "Neural networks for control systems–a survey", *Automatica*, vol. 28(6), pp. 1083–1112, 1992.

[ImF03] Imsland L. Findeisen R., Bullinger E., Allgower F., Foss B. A., “A note on stability, roubstness, and performance of output feedback nonlinear model predictive control”, *J. Process Control*, vol. 13, pp. 633–644, 2003.

[IF95] Iyer N. M., Farell A. E., “Adaptive input-output linearizing control of a continuous stirred tank reactor”, *Comput. Chem. Eng.*, vol. 19, pp. 575-579, 1995.

[Ino03] Inou M., “An application of automatic differentiation to optimal control of fluid force around a circular cylinder”, *Kawahara Lab*, vol. 1, pp. 1–5, 2003.

[Iri97] Iri M., “Roles of automatic differentiation in nonlinear analysis and high–quality computation”, *Nonlinear Analysis, Theory, Methods and Applications*, vol. 30(7), pp. 4317–4328, 1997.

[Jaz04] Jazayeri–Rad H., “The nonlinear model predictive control of a chemical plant using multiple neural networks”, *Neural Computer and Application*, vol. 13, pp. 2-15, 2004.

[JD93] Jubien C. M., and Dimopoulos J. D., “Recurrent neural networks in system identification”, *IEEE Int. Symposium on Circuits And Systems (ISCAS ’93)*, vol. 4 , pp. 2458–2461, May, 1993.

[JF95] Johansen T. A. and Foss B. A., “Empirical modelling of a heat transfer process using local models and interpolation”, *American Control Conf.*, Seattle, Wa., pp. 3654–3658, 1995.

[JG01] Jiang F. and Gao Z., “An application of nonlinear PID control to a class of Truck ABS problem”, Presented at: *2001 IEEE Conference on Decision and Control*, 2001.

[JJM87] Jang S., Joseph M, and Mukai H., “Control of constrained multivariable nonlinear process using a two–phase approach”, *Ind. Eng. Chem. Res.*, vol. 26, pp. 2106–2114, 1987.

[JJM89] Joseph B., Jang S. S., Mukai H., “Integrated model based control of multivariable nonlinear systems”, In: McAvoy T. J., Arkun Y., Zafiriou, E., Eds., “model based process control”, Pergamon Press, New York, 1989.

[JNG95] Jin L., Nikiforuk P. , Gupta M., “Approximation of discrete–time state–space trajectories using dynamic recurrent neural networks”, *IEEE Transactions on Automatic Control*, vol. 40(7), pp. 1266-1270, 1995.

[Joh95] Johansen T. A., “Operating regime based process modelling and identification”, *PhD Thesis, Dep. of Eng., Cybernetics, University of Trondheim*, Norway, 1994.

[KaC00] Kambhampati C., Craddock R. J., Tham M., Warwick K., “Inverse model control using recurrent networks”, *Mathematics and Computers in Simulation*, vol. (51), pp. 181-199, 2000.

[Kal60] Kalman R. E., “A new approach to linear filtering and prediction problem”, *Trans. ASME, Ser.*, D82, pp. 35–45, 1961.

[KaMC96] Kambhampati C., Manchanda S., Delgado A., Green G., Warwick K., Tham M., “The relative order and inverses of recurrent networks”, *Automatica*, vol. 32(1), pp. 117-123, 1996.

[Kar77] Karybakas C. A., “Nonlinear integrator with zero phase shift”, *IEEE Trans.*, vol. IECI, 1977. In cite: Sheng A., Satoshi N., Yamaguchi S., Itakura H., “A new nonlinear integrator with positive phase shifts”, *IEICE Trans. Fundamentals*, vol. E81–A(1), pp. 197–201, 1998.

[KC01] Kouvaritakis B. and Cannon M., “Nonlinear predictive control: theory and practice, chapter 1”, *IEE Control Engineering Series 61, The Institution of Electrical Engineers*, London, 2001.

[KF03] Kim J. G. and Finsterle S., “Application of automatic differentiation in TOUGH2”, *Proceedings, TOUGH Symposium 2003*,California,pp. 1–5, 2003.

[KGE88] Keerthi S. S., and Gilbert E. G., “Optimal infinite–horizon feedback laws for a general class of constrained discrete–time systems: stability and moving–horizon approximations”, J. Theory Appl., vol.57(2), pp. 265–293, 1988.

[KGW00] Kambhampati C., Garces F., Warwick K., “Approximation of non–autonomous dynamic systems by continuous time recurrent neural networks”, *Proceeding of the IEEE–INNS–ENNS International Joint Conference on Neural Networks IJCNN 2000*, vol. 1, pp. 64–69, 2000.

[KH97] Kurtz M. J. and Henson M. A., “Input–output linearizing control of constrained nonlinear processes”, *J. Process Control*, vol. 7, pp. 3–17, 1997.

[KP91] Korenberg M. J. and Paarmann L. D., “Orthogonal approaches to time–series analysis and system identification”, *IEEE Signal Proc. Mag.*, vol. 7, pp. 29-43, 1991.

[KR03] Khadir M. T. and Ringwood J. V., “Linear and nonlinear model predictive control design for a milk pasteurization plant”, *Control Intell. Sys.*, vol. 31, pp. 1–8, 2003.

[Kri80] Krikelis N. J., “State feedback integral control with ‘intelligent’ integrators”, *Int. J. Control*, vol. 32(3), pp. 465–473, 1980.

[Lew86] Lewis F. L., “Optimal estimation”, *NewYork: Wiley*, 1986.

[LGM92] Lee J. H., Gelormino M. S., and Morari M., “Model predictive control of multi–rate sampled data systems: A state–space approach”, *Inter. J. Control*, vol. 55, pp. 153–191., 1992.

[LP99] Li S. and Petzold L. R., “Design of new DASPK for sensitivity analysis”, *UCSB Department of Computer Science Technical Report*, 1999.

[LK99] Leonard J. A. and Kramer M. A., “Improvement of the back–propagation algorithm for trainig neural networks”, *Chemical Eng.*, vol. 14, pp. 337–341, 1990.

[LP94] Lee J. and Park S., “Robust nonlinear predictive control using a disturbance estimation”, *Chem. Eng. Comm.*, vol. 128, pp. 43–64, 1994.

[LR94] Lee J. H., and Ricker N. L., “Extended Kalman filter based nonlinear model predictive control”, *Ind. Eng. Chem. Res.*, vol. 33, pp. 1530–1541., 1994.

[LW93] Lindskog P. and Wahlberg B., “Applications of Kautz models in system identification”, *in Preprints 12th IFAC World Cong.*, pp. 5:309–312, July 1993.

[LY94] Lee J. H. and Yu Z., “Tuning of model predictive controllers for robust performance”, *Computer Chemical Eng.*, vol. 18(1), pp. 15–37, 1994.

[LzP02] Lazar, and Pastravanu, “A neural predictive controller for nonlinear systems”, *Mathematics and Computers in Simulation*, vol. 60, pp. 315–324, 2002.

[Mac02] Maciejowski J. M., “Predictive control with constraints”, *Prentice Hall*, Harlow, England, 2002.

[MaD96] Maner B. R., Doyle F. J., Ogunnaike B. A., and Pearson R. K., “Nonlinear model predictive control of a simulated multivariable polymerization reactor using second–order Volterra models”, *Automatica*, vol. 32, pp. 1285–1301, 1996.

[Mar63] Marquardt D., “An Algorithm for least–squares estimation of nonlinear parameters”*SIAM J. App. Math.*, vol. 11, pp. 431–441, 1963.

[MaR00] Mayne D. Q., Rawlings J. B., Rao C. V., Scokarent P.O. M., “Constrained model predictive control: stability and optimality”, *Automatica*, vol. 26(6), pp. 789–814, 2000.

[Mat91] Mathews V. J., ”Adaptive polynomial filters”, *IEEE Signal Proc. Mag.*, no. 7, pp. 10–26, 1991.

[May82] Maybeck P. S., “Stochastic models, estimation, and control”, 2, London: Academic Press, 1982.

[MBF04] Martinsen F., Biegler L. T., and Foss B. A., “A new optimization algorithm with application to nonlinear MPC”, *J. Proc. Cont.*, (in press), 2004.

[McS97] McMullan J. T., Sloan E. P., “Clean coal technologies, emission control technology for power and process plant”, *IMechE*, vol. 1, chapter 3, pp. 95–107, 1997.

[ML99] Morari M. and Lee H., “Model predictive control: past, present, and future”, *Computer and Chem. Eng.*, vol. 23; pp. 667–682, 1999.

[MLL86] Morshedi A. M., Lin H. Y., Luecke R. H., “Rapid computation of the Jacobian matrix for optimization of nonlinear dynamic processes”, *Computers and Chemical Engineering*, vol. 10(4), pp. 367–376, 1986.

[MM90] Mayne D. Q. and Michalska H., “Receding horizon control of nonlinear systems”, *IEEE Trans. Automat. Control.*, vol. 35(7), pp. 814–824, 1990.

[MM93] Michalska H., and Mayne D. Q., “Robust receding horizon control of constrained nonlinear systems”, *IEEE Trans. Automatic Control*, vol. 38(11), pp. 1623–1633, 1993.

[Mor86] Morshedi A. M., “Universal dynamic matrix control”, In: *Chemical Process Control III*, Morari M., McAvoy T. J., Eds., Elsevier: New York, pp. 547–577, 1986.

[MR97] Meadows E. S. and Rawlings J. B., “Model predictive control”, In: Michael A. Henson and Dale E. Seborg, editors, *Nonlinear Process Control*, chapter 5, pp. 233–310, Prentice Hall, 1997.

[MSW90] Miller W. T., Sutton R. S., and Werbos P. J., “Neural networks for control”, MIT Press, Cambridge, MA, 1990.

[MuB02] Muske K. R. and Badgwell T. A., "Disturbance modelling for offset–free linear model predictive control", *J. Process Control*, vol. 12, pp. 617–632, 2002.

[MuR93] Muske K. E. and Rawlings J. B., "Model predictive control with linear models", *AIChE J.*, vol. 39(2), pp. 262, 1993.

[NBC95] Narendra K. S., Balakrishnan J., and Ciliz M. K., "Adaptation and learning using multiple models, switching and tuning", *IEEE Control System*, vol. 15(3), pp. 37–51, 1995.

[NKR97] Narayanan N. R., Krishnaswamy P. R., and Rangaiah G. P., "An adaptive internal model control strategy for pH neutralization", *Chem. Eng. Scince*, vol. 52, pp. 3067–3074, 1997.

[NL89] Newell R. B. and P. L. Lee, "Applied process control–a case study", *Prentice Hall, Englewood Cliffs, NJ*, 1989.

[NP90] Narendra K. S. and Parthasarathy K., "Identification and control of dynamical systems using neural networks", *IEEE Trans. Neural Networks*, vol. 1(1), pp. 4–26, 1990.

[NP91] Narendra K. S. and Parthasarathy K., "Gradient methods for the optimization of dynamical systems containing neural networks", *IEEE Trans, Neural Networks*, vol. 12(2), pp. 252–262, 1991.

[NPR98] Norquay S. J., Palzoglu A., and Romagnoli J. A., "Model predictive control based on Wiener models", *Chem. Eng. Science*, vol. 53, pp. 75-84, 1998.

[Nor01] Nørgaard M., "State estimation for nonlinear systems: KALMTOOL version 2 for use with MATLAB", *Technical Report IMM-REP-2000-6, Technical Univ. of Denmark*, 2001.

[NPR00] Nørgaard M., Poulsen N. K. and Ravn O., "New developments in state estimation for nonlinear systems", *Automatica*, vol. 36, pp. 1627–1638, 2000.

[Pan03] Pannocchia G., "Roubst disturbance modelling for model predictive control with application to multivariable ill–conditioned processes", *J. of Process Control*, vol. 13, pp. 693–701,2003.

[PBV03] Prasad V., and Bequette B. W., "Nonlinear system identification and model reduction using artificial neural networks", *Computers and Chemical Engineering*, vol. 27, pp. 1741-1754, 2003.

[PE93] Patwardhan A. A. and Edgar T. F., “Nonlinear model predictive control of packed distillation column”, *Ind. Eng. Chem. Res.*, vol. 32, pp. 2345–2356, 1993.

[PeH89] Peterson T., Hernandez E., Arkun Y., Schork F. J., “Nonlinear predictive control of a semi–batch polymerization reactor by extended DMC”, *In Proceedings of the 1989 American Control Conference* , pp. 1534–1539, July, 1989.

[PeB95] Pearlmutter B. A., “Gradient calculations for dynamic recurrent neural networks: A survey”, *IEEE Trans. on Neural Networks*, 1995.

[PeR94] Pearson R. K., “Nonlinear input/output modelling”, *Preprints IFAC Symposium ADCHEM’94*, Koyoto Research Park, Kyoto, Japan, 1994.

[PeR03] Pearson R. K., “Selecting nonlinear model structures for computer control: review”, *Journal of Process Control*, vol. 13, pp. 1-26, 2003.

[PiS00] Piche S., Sayyar-Rodsari B., Johnson D., and Gerules M., “Nonlinear model predictive control using neural networks”. *IEEE control system magazine*, vol. 20(3), pp. 53–62, 2000.

[PR03] Pannocchia G. and Rawlings J. B., “Disturbance models for offset–free model predictive control”, *AIChE J.*, vol. 49, pp. 426–437, 2003.

[PRE90] Patwardhan A. A., Rawlings J. B., and Edgar T. F., “Nonlinear model predictive control”, *Chem. Eng. Comm.*, vol. 87, pp. 123–141, 1990.

[PSR97] Palizban H. A., Safavi A. A., and Romagnoli J. A., “A practical multi–model approach for controlling nonlinear process”, *CONTROL’97*, Iasted–Acta Press, pp. 169–174, 1997.

[QB97] Qin S. J. and Badgwell T. A., “An overview of industrial model predictive control technology”, In Kantor J. C., Garcia C. E., and Carnahan B., editors, *Fifth International Conference on Chemical Process Control–CPC V*, pp. 232–256, American Institute of Chem. Eng., 1997.

[QB03] Qin S. J. and Badgwell T. A., “A survey of industrial model predictive control technology”, *Control Eng. Practice*, vol. 11, pp. 733–264, 2003.

[Ral81] Rall L. B., “Automatic differentiation: techniques and applications” *Lecture Notes in Computer Science, Berlin:Springer–Verlag*, vol. 120, , 1981.

[RAB03] Rao R. R., Aufderheide B., and Bequette B. W., “Experimental studies on multiple–model predictive control for automated regulation of hemodynamic variables”, *IEEE Transactions Biomedical Engineering*, vol. 50, pp. 277–288, 2003.

[RbR03] Robenack K. and Reinschke K. J., “Nonlinear observer design using automatic differentiation”, In Proceeding: *Modelling Identification and Control (MIC 2003)*, Innsbruck, Austria, 2003.

[Rich89] Rich L. C., “An implementation of automatic differentiation”, *Master’s Thesis*, Department of Mathematics, Temple University, Philadelphia, 1989.

[Ric90] Ricker N. L., “Model predictive control with state estimation”, *Ind. Eng. Chem. Res.*, vol. 29, pp. 374–382, 1990.

[RiR78] Richalet J., Railt A., Testud J. L., Papou J., “Model predictive heuristic control: application to industrial processes”, *Automatica*, vol. 14(413), 1978.

[RH92] Rich L. C. and Hill D. R., “Automatic differentiation in MATLAB”, *Applied Numerical Mathematics*, vol. 9, pp. 33–43, 1992.

[RO00] Rodrigues M. A. and Odloak D., “Output feedback MPC with guaranteed robust stability”, *J. of Process Control*, vol. 10, pp. 557–572, 2000.

[RM93] Rawlings J. B. and Muske K. R., “The stability of constrained receding horizon control”, *IEEE Trans. Automatic Control*, vol. 38(10), pp. 1512–1516, 1993.

[RMM94] Rawlings J. B., Meadows E. S. and Muske K. R., “Nonlinear model predictive control: a tutorial and survey”, ADCHEM ’94 Proceedings, Kyoto, Japan, 1994.

[RRS02] Rice M., Rossiter J., and Schurmans J., “An advanced predictive control approach to the ALSTOM gasifier problem,” *Proc. Instn Mech. Engrs. Part I, Journal of Systems and Control Engineering*, vol. 214, pp. 405–413, 2002.

[RST02] Rangaiah G. P., Saha P., and Tade M. O., “Nonlinear model predictive control of an industrial four–stage evaporator system via simulation”, *Chem. Eng. J.*, vol. 87, pp. 285–299, 2002.

[Rug87] Rugh W. J., “Design of nonlinear PID controllers”, *AIChE J.*, vol. 33(10), pp. 1738–1741, 1987.

[RuH86] Rumelhart D. E., Hinton G. E., and Williams R. J., “Learning internal representiations by error propagation”, *in Prallel Distributed Processing, D. E. Rumelhart and J. L. MeClelland, Eds.*, Cambridge MA: MIT Press, 1986.

[RV04] Robenack K., Vogel O., “Computation of state and input trajectories for flat systems using automatic differentiation”, *Automatica*, vol. 40, pp. 459–464, 2004.

[RWR98] Rao C. V., Wright J. B., and Rawllings J. B., “ Application of interior-point methods to model predictive control”, *J. Optim. Theory Appl.*, vol. 99, pp. 723–757, 1998.

[SaA01] Santos L. O., Afonso P. A., Castro J. A., Olivera N. M., and Biegler L. T., “On –line implementation of nonlinear MPC; an experimental case study”. *Control Engineering Practice*, vol. 9(9), pp. 847-857, 2001.

[Sah99] Saha P., “Model predictive control of chemical processes using nonlinear Laguerre models”, *Ph. D. thesis, Process Control, Dep. of Chemical Engineering, Indian institute of Technology*, Madras, 1999.

[Sah04] Saha P., KrishnanS. H., Rao V. S., Patwardhan S. C., “Modelling and predictive control of MIMO nonlinear systems using Wiener–Laguerre models”, *Chem. Eng. Comm., Taylor and Francis Group*, vol. 191, pp. 1083–1119, 2004.

[SB91] Sistue P. B. and Bequette B. W., “Nonlinear predictive control of uncertain processes: application to CSTR”, *AICh Journal*, vol. 37(11), pp. 1711–1723, 1991.

[ScM04] Schlegel M., Marquardt W., Ehrig R., Nowak U., “Sensitivity analysis of linearly implicit differential–algebraic systems by one–step extrapolation”, *Applied Numerical Mathematics*, vol. 48, pp. 83–102, 2004.

[Ser98] Seraji H., “A new class of nonlinear PID controllers with robotic applications”, *J. Robot. Syst.*, vol. 15(3), pp. 161–181, 1998.

[SGB93] Sistue P. B., Gopinath R. S., and Bequette B. W., “Computational issues in nonlinear predictive control”, *Comput. Chem. Eng.*, vol. 17, pp. 361–367, 1993.

[ShD93] Shukla N. V., Deshpande P. B., Kumar V. R., Kulkarni B. D., “Enhancing the robustness of internal–model based nonlinear pH controller”, *Chem. Eng. Science*, vol. 48, pp. 913–920, 1993.

[SI89] Sastry S. S. and Isidori A., “Adaptive control of linearizable systems”, *IEEE Trans. Autom. Contr.*, vol. 34, pp. 1123–1131, 1989.

[SJ97] Sbarbaro D. and Johansen T. A., “Multiple local Laguerre models for modeling nonlinear dynamics systems of the Wiener class”, *IEE Proc. Control Theory Appl.*, vol. 144(5), pp. 375–380, 1997.

[SK99] Silva R. G. and Kwong W. H., “Nonlinear model predictive control of chemical processes”, *Brazilian J. of Chem. Eng.*, vol. 16(1), 1999.

[ShP00] Shengtai L. and Petzold L., “Software and algorithm for sensitivity analysis of large–scale differential algebra systems”, *J. of Computational and Applied Mathematics*, vol. 125, pp. 131–145, 2000.

[SH03] Serban R., Hindmarch A. C., “CVODES: An ODE solver with sensitivity analysis capabilities”, *Technical Report UCRL-JP-200039, Lawrence Livermore National Laboratory, U.S. Department of Energy*, 2003.

[SnA96] Sentoni G. B., Agamennoni O., Dessages A., Romagnoli J., “Approximate models for nonlinear process control”, *AIChE J.*, vol. 42(8), pp. 2240–2250, 1996.

[SBG98] Sentoni G. B., Biegler L. T., Guiver J. B., and Zhao H., “State–space nonlinear process modelling”, *AIChE J.*, vol. 44(10), pp. 2229–2239, 1998.

[SMc97] Su H. T. and McAvoy (1997), “Artificial neural networks for nonlinear process identification and control”, In: Henson, M. A. & Seborg, D.E. (Eds). “Nonlinear Process Control”, Chap. 7, pp. 371–428, Englewood Cliffs, NJ: Prentice–Hall.

[Son98] Sontag E., “A learning result for continues–time recurrent neural networks”, *Syst. Contr. Letters*, vol. 34, pp. 151–158, 1998.

[SP01] Serban R. and Petzold L. R., “COOPT–a software package for optimal control of large–scale differential–algebraic equation systems”, *Mathematics and Computers in Simulation*, vol. (56), pp. 187–203, 2001.

[SPF00] Stamatiadis S., Prosmiti R., Farantos S. C., “AUTO–DERIV: tool for automatic differentiation of a FORTRAN code”, *Computer Physics Communications*, vol. 127, pp. 343–355, 2000.

[SrA97] Sriniwas G. R. and Arkun Y. , “A global solution to the nonlinear predictive control algorithms using polynomial ARX models”, *Comput. Chem. Eng.*, vol. 21, pp. 431–439, 1997.

[SS98] Sheng A., Satoshi N., Yamaguchi S., Itakura H., “A new nonlinear integrator with positive phase shifts”, *IEICE Trans. Fundamentals*, vol. E81–A(1), pp. 197–201, 1998.

[StH99] Storen S., Hertzberg T., “Obtaining sensitivity information in dynamic optimization problems solved by the sequential approach”, *Computer and Chemical Engineering*, vol. 23, pp. 807–819, 1999.

[StP93] Stephanopoulos G., Park S., “Bioreactor state estimation”, In: K. Schugerl (Ed.), Biotechnology, *Measuring, Modeling and Control*, vol. 4, pp. 225C249, 1992.

[UC03] Ungarala S. and Z. Z. Chen, “Bayesian data rectification of nonlinear systems with Markov chains in cell space, *Proc. ACC*, pp. 4857-4862, Denver, CO, 2003.

[TC96] Tan Y., and Cauwenberghe A., “Nonlinear one step ahead control using neural networks: control strategy and stability design”, *Automatica*, vol. 32(12), pp. 1701–1706, 1996.

[TB98] Tolsma, J. E. and Berton P. I, “On computational differentiation” *Computers Chemical Eng.*, vol. 22(4/5), pp. 475–490, 1998.

[Ten02] Tenny M. J., “Computational strategies for nonlinear model predictive control”, *PhD. Thesis, University of Wisconsin–Madison*, 2002.

[THL00] Tan K. K., Huang S. N., and Lee T. H., “Development of a GPC–based PID controller for unstable systems with deadtime”, *ISA Transactions*, vol. 39, pp. 57–70, 2000.

[TRW04] Tenny M. J., Rawling J. B., and Wright S. J., “Closed–loop behavior of nonlinear model predictive control”, *AIChE J.*, vol. 50(9), pp. 2142, 2004.

[TSM95] Temeng H., Schenelle P., and McAvoy T., “Model predictive control of an industrial packed reactors using neural networks”, *J. Process Control*, vol. 5(1), pp. 19–28, 1995.

[Veh04] Vehreschild A., “ADiMAT AD in MATLAB”, *Institute for Scientific Computing RWTH Aachen University*, 2004.

[Ver98] Verma A., “Structured automatic differentiation”, *PhD Thesis*, Cornell University, 1998.

[Ver00] Verma A.,“An introduction to automatic differentiation”, *Current Science*, vol. 78, pp. 804–807, 2000.

[Wah94] Wahlberg B., ”System identification using Kautz models”, *IEEE Trans. Automatic Control*, vol. 39(6), pp. 2333–2359, 1994.

[WB04] Welch G., Bishop G., “An introduction to the Kalman filter”, http://www.cs.unc.edu/welch/media/pdf/kalman intro.pdf, 2004.

[Wen64] Wengert R. E., “A simple automatic derivative evaluation program” *Comm. ACM*, vol. 7, pp. 463–464, 1964.

[Wil64] Wilkins R. D., “Investigation of a new analytic method for numerical derivative evaluation” *Comm. ACM*, vol. 7, pp. 465–471, 1964.

[YG02] Yu D. L. and Gomm J. B., “Implementation of neural network predictive control to a multivariable chemical reactor”, *Control Engineering Practice*, 2002.

[YZ90] Yong W., and Zhengpin C., “An intelligent integrator”, *Information and Control*, No. 6, 1990. In cite: Sheng A., Satoshi N., Yamaguchi S., Itakura H., “A new nonlinear integrator with positive phase shifts”, *IEICE Trans. Fundamentals*, vol. E81–A(1), pp. 197–201, 1998.

[XiW01] Xiang L., Weitao Z., Zhijang S., Jixin Q., “Applying extended automatic differentiation technigue to process system optimization problems”, *Proceedings of the American Control Conference, Arlington VA*, pp. 4079–4084, 2001.

[XZJ04] Xiang L., Zhijang S., Jixin Q., “Module–oriented automatic differentiation in chemical process system optimization”, *Computers and Chemical Eng.*, vol. 28, pp. 1551–1561, 2004.

[ZB02] Zhu Y. and Butoyi F., “Case studies on closed-loop identification for MOPC”, *Control Engineering Practice*, vol. 10, pp. 403–417, 2002.

[ZGS98] Zhao H., Guiver J., and Sentoni G., “An identification approach to nonlinear state space model for industrial multivariable model predictive control”, *Proceeding of the American Control Conference, Philadelphia, Pennsylvania*, 1998.

[ZhX03] Zhao Z., Xia X., Wang J., Gu J., Jin Y., “Nonlinear dynamic matrix control based on multiple operating models”, *J. of process Control*, (in Press), vol. 13, pp. 41–56(16),2003.

[ZV98] Zamarreno J. M. and Vega P., “State–space neural network, properties and application”, *Neural Networks*, vol. 11, pp. 1099–1112, 1998.

[ZM99] Zhang J., and Morris J., “Recurrent neuro–fuzzy networks for nonlinear process modeling”, *IEEE Trans. on Neural Networks*, vol. 10(2), pp. 313–326, 1999.

Appendix A

First–Principle Model of Evaporator Process

The first–principle model of the process is given by the following differential equations;

$$\frac{dL_2}{dt} = \frac{F_1 - F_4 - F_2}{20} \tag{A.1}$$

$$\frac{dX_2}{dt} = \frac{F_1X_1 - F_2X_2}{20} \tag{A.2}$$

$$\frac{dP_2}{dt} = \frac{F_4 - F_5}{4} \tag{A.3}$$

$$T_2 = 0.561P_2 + 0.3126X_2 + 48.43 \tag{A.4}$$

$$T_3 = 0.507P_2 + 55.0 \tag{A.5}$$

$$F_4 = \frac{Q_{100} - F_1C_P(T_2 - T_1)}{38.5} \tag{A.6}$$

$$T_{100} = 0.1538P_{100} + 90.0 \tag{A.7}$$

$$Q_{100} = 0.16(F_1 + F_3)(T_{100} - T_2) \tag{A.8}$$

$$F_{100} = Q_{100}/36.6 \tag{A.9}$$

$$Q_{200} = \frac{13.68F_{200}C_P(T_3 - T_{200})}{2C_PF_{200} + 6.84} \tag{A.10}$$

$$T_{201} = T_{200} + \frac{Q_{200}}{F_{200}C_P} \tag{A.11}$$

$$F_5 = Q_{100}/38.5 \tag{A.12}$$

Table A.1: Steady-state values of the evaporator system

Variables	Description	Value	
F_1	Feed flowrate	10	kg/min
F_2	Product flowrate	2.0	kg/min
F_3	Circulating flowrate	50	kg/min
F_4	Vapor flowrate	8.0	kg/min
F_5	Condensate flowrate	8	kg/min
X_1	Feed composition	5.0	%
X_2	Product composition	25	%
T_1	Feed temperature	40.0	%
T_2	Product temperature	84.6	^{o}C
T_3	Vapor temperature	80.6	^{o}C
L_2	Separator level	1.0	m
P_2	Operating pressure	50.5	kPa
F_{100}	Steam flowrate	9.3	kg/min
T_{100}	Steam temperature	119.9	^{o}C
P_{100}	Steam pressure	194.7	kPa
Q_{100}	Heat duty	339	kW
F_{200}	Cooling water flowrate	208	kg/min
T_{200}	Inlet C. W. temperature	25.0	^{o}C
T_{201}	Outlet C. W. temperature	46.1	^{o}C
Q_{200}	Condenser duty	307	kW

Appendix B

First–Principle Model of the Two CSTR Process

The first–principle model of the two–cstr process is given by the following differential equations [Cao95];

$$
\begin{aligned}
\frac{dx_1}{dt} &= -K_1 x_1 + Q_{I1}\frac{(C_{I1}-x_1)}{V_1} && \text{(B.1)}\\
\frac{dx_2}{dt} &= \Delta H K_1 x_1 + Q_{I1}\frac{(T_{I1}-x_2)}{V_1} - U_{a1}\frac{(x_2-x_3)}{V_1} && \text{(B.2)}\\
\frac{dx_3}{dt} &= \frac{1}{V_{J1}}(Q_{CW1}(T_{CW1}-x_3) + U_{a1}(x_2-x_3)) && \text{(B.3)}\\
\frac{dx_4}{dt} &= -K_2 x_4 + K_{V1}\sqrt{V_1}\frac{(x_1-x_4)}{V_2} + Q_{I2}\frac{(C_{I1}-x_4)}{V_2} && \text{(B.4)}\\
\frac{dx_5}{dt} &= \Delta H K_2 x_4 + K_{V1}\sqrt{V_1}\frac{(x_2-x_5)}{V_2} + Q_{I2}\frac{(T_{I2}-x_5)}{V_2} - U_{a2}\frac{(x_5-x_6)}{V_2} && \text{(B.5)}\\
\frac{dx_6}{dt} &= \frac{1}{V_{J2}}(Q_{CW2}(T_{CW2}-x_6) + U_{a2}(x_5-x_6)) && \text{(B.6)}
\end{aligned}
$$

where;

$$
\begin{aligned}
K_1 &= K_0 e^{-E/Rx_2} && \text{(B.7)}\\
K_2 &= K_0 e^{-E/Rx_5} && \text{(B.8)}
\end{aligned}
$$

Table B.1: Physical and process constants of the two–CSTR process

Constant	Value	Units
V_1	4.489	m^3
V_2	5.493	m^3
K_{V1}	0.16	$m^{3/2}s^{-1}$
K_{V2}	0.256	$m^{3/2}s^{-1}$
$U_{a1},\ U_{a2}$	0.35	m^3s^{-1}
E/R	6000	K
ΔH	5	$m^3 \cdot K \cdot mol^{-1}$
K_0	2.7×10^8	s^{-1}
$V_{J1},\ V_{J2}$	1	m^3

Table B.2: Two–CSTR process steady-state variables value

Variable	Value	Units
x_1	0.084	mol/m^3
x_2	362.995	^oK
x_3	327.560	^oK
x_4	0.053	mol/m^3
x_5	362.995	^oK
x_6	335.447	^oK
Q_{I1}	0.339	m^3s^{-1}
Q_{I2}	0.261	m^3s^{-1}
Q_{CW1}	0.45	m^3s^{-1}
Q_{CW2}	0.272	m^3s^{-1}
$T_{I1}, T_{I2}, T_{CW1}, T_{CW2}$	300	^oK
C_{I1}, C_{I2}	20	mol/m^3