To the Graduate Council:

I am submitting herewith a dissertation written by Suzanne D. Roat entitled "The Application of Neural Networks and System Cultivation with a Nonlinear Optimal Control Algorithm for the Chemical Process Industry." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Chemical Engineering.

Charles F. Moore

Dr. C. F. Moore, Major Professor

We have read this dissertation
and recommend its acceptance:

Edwin L. LeCuyer

Robert M. Counce

Deane D. Bruns

Ernest F. Vogel

Accepted for the Council:

Carmichael

Vice Provost
and Dean of The Graduate School

THE APPLICATION OF NEURAL NETWORKS AND SYSTEM CULTIVATION WITH A NONLINEAR OPTIMAL CONTROL ALGORITHM FOR THE CHEMICAL PROCESS INDUSTRY

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Suzanne D. Roat

May 1991

ACKNOWLEDGMENTS

The author would like to express her sincere appreciation to a number of professors, colleagues, and friends that provided the inspiration and motivation to complete this dissertation. First, the author would like to express deep gratitude to Dr. Charlie Moore for directing this work as well as providing moral and emotional support during the culmination of this work. Second, the author is indebted to Dr. Jim Downs and Dr. Ernie Vogel of the Tennessee Eastman Company for providing the idea for this work, helping to keep the research going in the right direction, and serving on the committee approving this work. Third, the author wishes to thank her fellow graduate students for help, guidance, and support received during the years of research work. Namely, the author is indebted to Jim Hackney, David Canter, and Andy Farell for services rendered beyond the call of duty that made completing this not only possible, but more enjoyable. Fourth, the author wishes to thank her undergraduate mentor, Dr. Steve Melsheimer of Clemson University for providing the inspiration to continue with this work, questioning her judgement along the way, lending a critical ear when the need arised, and generally just being there during the crises. Fifth, the author wishes to express her gratitude to the other members of the committee approving this work. Namely, Dr. Elden DePorter of the Industrial Engineering Department who taught the author all she knows about optimization. Also, the author appreciates the service provided by Dr. Duane

Bruns and Dr. Pete Counce in reviewing the work.

The author wishes to acknowledge her mother and father for annoying her with weekly phone calls asking about the progress of the research and just to say they cared about her work. Without the calls, the work would not be complete. Also, a special acknowledgment goes to the author's roommate, Liz Deming. Without her unbiased opinions and moral support, the author would have given up long ago. A great deal of admiration goes to her as she pursues her PhD for which the author's own struggle to achieve the same goal pales in comparison.

Finally, the author wishes to acknowledge the financial support provided by the Measurement and Control Engineering Center at the University of Tennessee.

ABSTRACT

Plants in the chemical process industry have become more difficult to control as they have increased in complexity and come to produce a wider variety of products. Within the last two decades, the chemical process industry has attempted to deal with some of the more difficult control problems through model based control schemes. This dissertation develops a nonlinear model predictive control scheme and addresses some of the practical issues related to using feedback in such schemes.

This work examines several areas of continued research with model predictive control (MPC) schemes: feedback analysis, process signal filtering, and unmeasured state variable estimation. The work presented in this dissertation deals with these areas using a technique called feedback cultivation. Feedback cultivation extracts temporal (time-related) information from a historical database to reduce the amount of data to be examined and to enhance the dynamic characteristics present in the historical data.

Other tools used in conjunction with feedback cultivation include neural networks and adaptive filtering. Neural networks are one form of pattern recognition that has seen increased use recently. Adaptive signal filters remove excessive noise from the process signal measurements.

The results presented in this work demonstrate that the feedback cultivation algorithm successfully incorporates feedback information into the nonlinear optimal control algorithm. Using the neural network,

process-to-model mismatch and unmeasured disturbances can be diagnosed and treated accordingly. In the case of process-to-model mismatch, a model parameter re-identification algorithm updates the inaccurate model parameter. The correction for unmeasured disturbances involves determining new values for model output variable bias terms using the model parameter re-identication algorithm. The model parameter re-identification algorithm involves the solution of a modified form of the nonlinear optimal control problem.

The results presented in this work involve applications of the nonlinear optimal control algorithm with feedback cultivation on a simple first-order process with a first-order model and on a continuous stirred tank reactor with a first principle nonlinear model.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

The availability of fast process control computers opens up the possibility of using increasingly sophisticated control strategies. Over the last two decades much of the development of advanced computer based strategies attempts to employ model predictive control schemes that use computer models of processes and plants and mathematical optimization techniques to determine appropriate control actions. This dissertation develops a nonlinear model predictive control scheme designed to handle complex processes or groups of processes in the chemical process industry.

The chemical process industry must deal with many constraints in the form of equipment limitations, safety factors, economic limits, and many others. Model predictive control schemes offer the advantage of dealing directly with any process constraints. Traditional control schemes typically deal with constraints in an *ad hoc* fashion with limit switches, high and low selectors, and anti-reset-windup devices.

Three aspects of model predictive control (MPC) schemes represent areas of continued research: feedback analysis, unmeasured state variable estimation, and process signal filtering.

1. Process feedback contains information about unmeasured process

disturbances, process-to-model mismatch, and the degree of signal noise in process measurements.

2. The model of the process requires initial conditions for the state variables that may need to be estimated because the actual process variables cannot be measured.

3. Noisy signals degrade the operation of MPC schemes. Process signal filtering may be necessary to reduce the amount of noise in the measurement signals before the measurements can be successfully used in feedback analysis or state variable estimation schemes.

To develop a more generally applicable MPC algorithm, this dissertation addresses items one and three extensively and item two to a lesser degree.

The contribution of this work in the field of model predictive control involves the development of an algorithm designed to incorporate feedback into a nonlinear optimal control algorithm. This feedback handling algorithm uses a recently developed perspective called system cultivation.

System cultivation, a recently developed collection of tools, extracts useful information from a historical database. Recording of process feedback signals during a period of process operation generates a historical database of information about the process and the MPC controller. The information contained in the historical database relates to the three problems all controllers must face: process signal noise, process-to-model mismatch, and unmeasured process disturbances. System cultivation tools extract subtle but

important relationships from the historical database to detect and diagnose system problems. This study uses two system cultivation tools: feedback cultivation and neural networks.

Feedback cultivation extracts temporal (time-related) information from the historical database to reduce the amount of data to be examined and to enhance the dynamic characteristics present in the data. Feedback cultivation employs a number of summary and statistical variables to describe time windows of data as single points in a reduced database. A pattern recognition technique scans the reduced database to determine what problems exist in the process.

Neural networks are one form of pattern recognition that has seen increased use recently. In the form of computer programs, neural networks attempt to mimic the transmission and storage of information used by the human brain and nervous system. In this work, a back-propagation perceptron neural network classifies the data in the reduced database and automatically detects unmeasured disturbances, process-to-model mismatch, and excessive signal noise. The neural network eliminates the need to have an operator examine the reduced database to detect process or controller problems.

Corrective action can be taken by the computer program after the source of a process or controller problem (in the form of process-to-model mismatch) has been diagnosed by the system cultivation tool. Bias terms in the model equations account for unmeasured process disturbances. A

nonlinear optimization scheme determines the optimum value of the bias terms. The same nonlinear optimization scheme also identifies inaccurate model parameters. A neural network trained as a parameter identifier can perform the model identification in the same manner as the nonlinear optimization scheme. Excessive signal noise is removed using process signal filtering.

In this work, adaptive signal filters are used to filter the process signals to remove excessive noise. Adaptive signal filters resemble frequently used multiple-order Z-transform filters except that a steepest descent optimizaton scheme determines the weighting factors. The weighting factors can be continuously updated, or updated only when the system cultivation tools determine that excessive signal noise is present in the measured signals.

The body of this dissertation is divided into six chapters. Chapter 2 reviews the history of model predictive control techniques as they apply to the chemical process industry. Chapter 3 explains the system cultivation tools including feedback cultivation, neural networks, and signal filtering. Chapter 4 describes the nonlinear model predictive control scheme used in this work. Chapters 5 and 6 give the results of applying the nonlinear model predictive control scheme to a simple first-order process and model example, and a more complex stirred-tank reactor, respectively. Chapter 7 presents conclusions and recommendations derived from this work.

CHAPTER 2

Model Predictive Control

A model predictive control (MPC) algorithm uses a model of the process to predict the future outputs of the process. The MPC algorithm optimizes future control actions to minimize the deviations of the predicted process outputs from specified target trajectories. The main difference between MPC algorithms and traditional proportional-integral-derivative (PID) mode controllers lies in the representation of the process, i.e. the model of the process, in the control algorithm. PID algorithms utilize an implicit model of the process, whereas MPC algorithms utilize explicit models.

The earliest suggestions for the development of MPC algorithms appeared in 1962 when Zadeh and Whalen recognized the connection between minimum time optimal control and linear programming [50]. Then, in 1963, Propoi proposed the moving horizon approach that is now the core of all MPC algorithms [50]. The moving horizon approach describes how the model determines the predictions of the future process outputs.

MPC algorithms have been used in various forms in the chemical process industry for over a decade [57,51,53,27,19,59,14,15,13,26,24,25,35,7]. The most widely known MPC algorithm, Dynamic Matrix Control (DMC), uses as the model a matrix of step response coefficients relating process

outputs to control actions. Other variations of MPC include Quadratic DMC (QDMC), a constrained version of DMC solved using quadratic programming; Model Algorithmic Control (MAC), which uses an impulse response model of the process; and Linear DMC (LDMC), also a constrained version of DMC solved with linear programming techniques. Attempts to generalize the basic DMC algorithm have resulted in Internal Model Control (IMC), Generalized Predictive Control (GPC), Universal DMC (UDMC), and Generic Matrix Control (GMC). The differences, similarities, and important features of each of these MPC algorithms and others will be explained in the following sections.

The history of MPC applications in the chemical process industry begins with Richalet, who in 1978 successfully applied a Model Predictive Heuristic Control (MPHC) algorithm, which is similar to Model Algorithmic Control, to a chemical process [43,58]. In 1979, Cutler and Ramaker [19] and Prett and Gillette [57], all of Shell Development Corporation, applied a DMC algorithm to the control of a fluid catalytic cracker. Other notable applications of MPC algorithms include an application of a quadratic programming form of IMC to an evaporation process in a Kraft pulp mill by Ricker *et al.* [60] and an application of a DMC algorithm to a hydrocracker preflash distillation column in a refinery by Cutler *et al.* [18].

Balchen and Mummé [2] point out that MPC algorithms of the form used by Richalet, Cutler, Ramaker, Prett, Gillette and others are generalizations of the Smith predictor [66,49]. The Smith predictor was

developed by Smith [66] to compensate for process deadtime. In the Smith predictor, a simple, exponential model of the process is used to compensate for the process dynamics due to large process deadtimes. More general MPC algorithms use more elaborate models to compensate for process dynamics due to process disturbances as well as process deadtimes.

Mehra *et al.* [43] reviewed applications of MPC in the chemical process industry up to 1982. Since then, applications have become more frequent and complex mostly due to commercially available MPC software. This software is available from automatic control consulting firms such as DMC Corporation, founded by Cutler, and Setpoint Corporation (now a licensed DMC Corp. dealer), among others. Grosdidier *et al.* describe the IDCOM-M controller package, a multivariable MPC software package available from Setpoint Corporation [30].

MPC algorithms address the most serious shortcoming of traditional single-loop, PID-type controllers: the ability to handle process operating constraints. The chemical process industry must deal with an endless variety of constraints: equipment limitations, safety factors, economic limits, product specifications, environmental concerns, and many others. Typically, constraints are dealt with in an *ad hoc* fashion with limit switches and relays, high and low selectors, and anti-reset-windup devices. Because MPC algorithms are formulated as optimizations, the process constraints can easily be incorporated directly into the algorithm. The optimization formulation for MPC schemes will be explained in more detail in the following sections.

Because MPC algorithms deal directly with process constraints, they are particularly well-suited for applications in the chemical process industry.

Another feature of MPC algorithms that makes them attractive for use in the chemical process industry is the ease with which feedforward information can be incorporated into the algorithm. If measurements of process disturbance variables are available, they can be taken into account by the MPC algorithm in the determination of the future control moves. This increases the robustness of MPC algorithms in the face of process disturbances. Examples of disturbance variables that could be measured and used in the MPC algorithm include process feed stream temperatures, feed stream flow rates, feed stream concentrations, ambient temperatures and pressures, cooling or heating medium temperatures, and others.

2.1 MPC Models

As mentioned before, the main difference between MPC algorithms and traditional PID-type control algorithms is the model representation of the process in the controller. PID-type control algorithms contain an implicit model of the process, whereas MPC algorithms contain explicit models of processes. In other words, the MPC model of the process exists in a form that can be visually analyzed. Using explicit models has several advantages. First, the models can be specifically derived to match the process. Second, controller and process problems can be easier to diagnose and correct with

explicit models. Jorgensen [40] points out that a useful model must describe the possible process dynamics in the desired process operating regions and in the frequency range of interest.

Two types of models can be used by MPC algorithms: non-parametric and parametric. Non-parametric models involve vectors and matrices of coefficients determined to represent the relationship between the process state variables and the control variables or process disturbance variables. Parametric models consist of differential and algebraic equations that contain parameters that can be adjusted to fit the model to the process.

2.1.1 Non-parametric Models

MPC algorithms such as DMC, QDMC, LDMC, MAC, and others use non-parametric models to predict the process output and determine the best control actions. The dynamic matrix used as the model in DMC, QDMC, and LDMC is derived from the step-response coefficients relating process outputs to the control actions or disturbance variables. Figure 2.1 shows a typical step-response curve of the process output, y, after a step change has been made in the control action, u. Also indicated in Figure 2.1 are the values of the step-response coefficients a_1, a_2, ..., a_N. The dynamic matrix is constructed from these coefficients as shown in Figure 2.2. The dynamic matrix serves as the model in DMC and related MPC schemes. MAC and IMC algorithms use impulse-response coefficients to model the process.

Non-parametric models have several advantages. The resulting

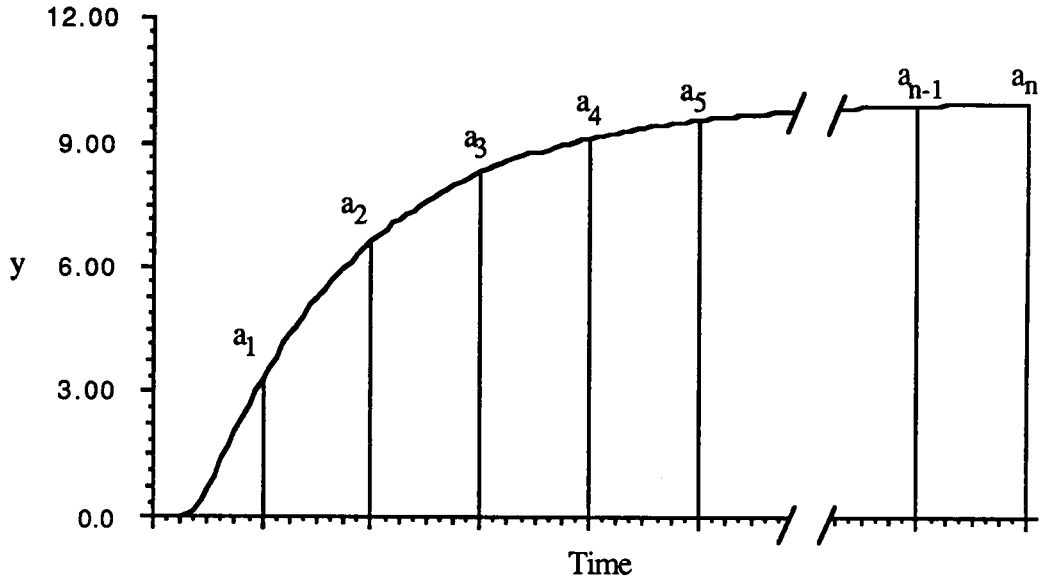


Figure 2.1: Step Response Curve

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ a_2 & a_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_M & a_{M-1} & \cdots & a_1 \\ \vdots & \vdots & \ddots & \vdots \\ a_P & a_{P-1} & \cdots & a_{P-M} \end{bmatrix}$$

where P = the prediction horizon

M = the control horizon

Figure 2.2: Sample dynamic matrix

matrices simplify the solution for the control moves, thus making DMC, QDMC, LDMC, and other related MPC algorithms computationally appealing. Nonlinearities of the process are incorporated into the dynamic matrix coefficients because actual process data is used to determine the coefficients. Any process deadtime can be clearly represented in the dynamic matrix by shifting the coefficients of the model in each column of the matrix down and adding more zeros to the tops of the columns.

The disadvantages of non-parametric models are significant. The coefficients in the vectors and matrices are derived from fixed process operating conditions. If the process operating conditions change, the model no longer adequately represents the process and may cause serious controller problems. The criteria specified by Jorgensen [40] and others for the model to describe the possible dynamics of the process are not met.

Another disadvantage of a non-parametric model is that the size of the dynamic matrix may become intractable. Sufficient coefficients need to be used to simulate the process until steady state is reached – usually between one and two process time constants. Highly nonlinear processes require close enough spacing of the coefficients to capture all the intermediate dynamics of the process output response. Short intervals and long process time constants work to create very large dynamic matrices, which increase the solution time for the control moves. Processes with large deadtimes also increase the size of the dynamic matrix. Multiple-input multiple-output (MIMO) systems increase the complexity of the dynamic matrix greatly. Figure 2.3 shows the

$$\mathbf{A} = \begin{bmatrix}
a_{11_1} & 0 & \cdots & 0 & a_{12_1} & 0 & \cdots & 0 \\
a_{11_2} & a_{11_1} & \cdots & 0 & a_{12_2} & a_{12_1} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_{11_M} & a_{11_{M-1}} & \cdots & a_{11_1} & a_{12_M} & a_{12_{M-1}} & \cdots & a_{12_1} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_{11_P} & a_{11_{P-1}} & \cdots & a_{11_{P-M}} & a_{12_P} & a_{12_{P-1}} & \cdots & a_{12_{P-M}} \\
a_{21_1} & 0 & \cdots & 0 & a_{22_1} & 0 & \cdots & 0 \\
a_{21_2} & a_{21_1} & \cdots & 0 & a_{22_2} & a_{22_1} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_{21_M} & a_{21_{M-1}} & \cdots & a_{21_1} & a_{22_M} & a_{22_{M-1}} & \cdots & a_{22_1} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_{21_P} & a_{21_{P-1}} & \cdots & a_{21_{P-M}} & a_{22_P} & a_{22_{P-1}} & \cdots & a_{22_{P-M}}
\end{bmatrix}$$

Figure 2.3: 2×2 Dynamic Matrix

structure of a 2-input 2-output dynamic matrix. The upper left block of the 2×2 dynamic matrix contains the step-response coefficients describing the effect of input u_1 on output y_1. The lower left block of coefficients describes the effect of input u_1 on output y_2. The upper and lower right blocks contain the coefficients describing the effect of input u_2 on outputs y_1 and y_2, respectively. For a 2×2 system, four step tests of the process must be performed. A 3-input 3-output system requires nine process step tests, although some of the tests can be run concurrently. Using the DMC algorithm becomes more prohibitive as the number of output variables to be controlled and the number of controlling input variables increases.

2.1.2 Parametric Models

Some MPC algorithms use state-space equations or differential and algebraic equations to represent the process. The most general MPC algorithm uses a nonlinear model of the process consisting of equations based on material, energy, and momentum balances. An example of a parametric model for a stirred-tank reactor is given in Figure 2.4. The parameters include variables such as heat transfer coefficients, heat capacities, heats of reaction, and other physically meaningful values. The resulting control algorithm is a nonlinear optimal control problem solved with nonlinear programming techniques. This work focuses on the nonlinear optimal control problem form of MPC. This is discussed in detail in Chapter 4.

Using MPC algorithms with parametric models avoids some of the

Mass Balance on Reactant A:

$$\frac{\partial C_a}{\partial t} = \frac{F}{V}(C_{ai} - C_a) - k_1 C_a^2$$

Energy Balance on Reactor:

$$\frac{\partial T}{\partial t} = \frac{F}{V}(T_i - T) - \frac{\Delta H_r}{\rho c_p} k_1 C_a^2 - \frac{UA}{V \rho c_p}(T - T_c)$$

Jacket Energy Balance:

$$\frac{\partial T_c}{\partial t} = \frac{UA}{V_c \rho_c c_{pc}}(T - T_c) - \frac{F_c}{V_c}(T_c - T_{ci})$$

Reaction Rate Coefficient:

$$k_1 = k_0 e^{\frac{E}{R(T+273.16)}}$$

Equal % Control Valve:

$$F_c = F_{c,max} \gamma^{-m} \quad 0 \le m \le 1$$

C_a = concentration of the reactant in reactor, kgmole / m^3

C_{ai} = inlet concentration of reactant, kgmole / m^3

T = reactor temperature, ^{0}C

T_c = jacket temperature, ^{0}C

T_i = inlet temperature, ^{0}C

T_{ci} = inlet coolant temperature, ^{0}C

b = transmitter signal (0 to 1)

F = feed flow rate, m^3 / s

V = reactor volume, m^3

k_1 = reaction rate coefficient, m^3 / kgmole s

ΔH_r = heat of reaction, J / kgmole

ρ = density of reactor contents, kgmole / m^3

c_p = reactant heat capacity, J / kgmole ^{0}C

U = overall heat transfer coefficient, J / s m^2 ^{0}C

A = heat transfer area, m^2

V_c = jacket volume, m^3

ρ_c = density of coolant, kg / m^3

c_{pc} = specific heat of coolant, J / kg ^{0}C

F_c = coolant flow rate, m^3 / s

m = controller output signal (0 to 1)

$F_{c,max}$ = maximum flow through control valve, m^3 / s

γ = valve rangeability parameter

k_0 = Arrhenius frequency parameter, m^3 / s kgmole

E = activation energy, J / kgmole

R = ideal gas law constant, 8314.39 J / kgmole K

Figure 2.4: Nonlinear parametric model of a stirred-tank reactor

shortcomings of non-parametric models. Parametric models can deal with multiple operating conditions of the process as long as the parameters in the model represent the values of the parameters in the process. In the case of MIMO processes, the model does not become intractable. One or two equations for each process variable may be added to the model, and the number of control actions searched for in the nonlinear optimization increases. Incorporating process disturbance measurements into the model is much easier with parametric models than with non-parametric models.

Although parametric models have many advantages, they also have several disadvantages. The complexity of the solution algorithm for the control actions is greater than most non-parametric solution methods. The most general case requires nonlinear programming techniques that can be difficult to set up and computationally intensive. Another problem, addressed in detail in Chapter 3, is the question of dealing with the measurements of the process state variables.

2.1.3 Neural Network Models

Neural networks represent an area of recent research interest. Neural networks attempt to mimic the storage and recall of information used in the brain. A complete description of neural networks will be given in Chapter 3, but they are mentioned here because of the possibility of using them as chemical process modelers.

Neural networks must be trained to produce a desired output given a

specific input. If a body of information about the operation of a process is available, a neural network could be trained to predict the process output if given the inputs determined by an MPC algorithm. So far, no attempts have been made to try neural network modeling with MPC algorithms. Bhat *et al.* [5,4] and von Westerholt *et al.* [67] have used neural networks to model highly nonlinear and complex chemical processes accurately. The neural network models could be used as part of an MPC algorithm to control these highly nonlinear and complex processes.

2.2 Modes of Operation

MPC algorithms have three basic modes of operation: prediction, correction, and control move determination. The prediction step uses the model of the process to determine the future values of the process outputs. The correction step compares the measured value of the process output to the value predicted by the model and corrects the predictions to account for the difference between the two values. The control move determination step performs the optimization, or more generally the model inversion, to calculate the future control moves needed to minimize or eliminate deviations of process output variables from target trajectories.

2.2.1 Prediction

The type of model used in the MPC algorithm determines how the prediction step is performed. For example, a DMC or related algorithm sums the effects of the past inputs and the future inputs on the process outputs to determine the predicted future value of the output:

$$y_p(k+l) = \sum_{i=l+1}^{M} a_i \Delta u(k+l-i) + \sum_{i=1}^{l} a_i \Delta u(k+l-i) \qquad (2.1)$$

where a_i, $i = l+1$ to M, are the step-response coefficients of the dynamic matrix, $\Delta u(k+l-i)$, are the past and future inputs, and $y_p(k+l)$ are the predicted values of the output for $l = 1, 2, 3, ...P$. The indices in the equation correspond to a discrete time index, with k equal to the current time. P and M are the prediction and control horizons, respectively. The first term in the equation accounts for the effects of the past control moves, and the second term accounts for the effect of the future control moves on the output prediction.

In the case of a nonlinear parametric model composed of differential and algebraic equations, an integration method must be applied to determine the values of the future outputs. Figure 2.4 shows the equations for a stirred-tank reactor that must be integrated to calculate the predicted output for the model state variables. Integration methods, such as the Runge-Kutta type methods, could be used to perform the integration.

In the prediction step, non-parametric models are superior in terms of speed of computation and stability. Based solely on vector and matrix

addition and multiplication, the prediction of the output can be quickly determined. Also, the prediction step need only be performed one time for each time a control move is determined.

As mentioned before, the drawback of non-parametric model predictions is the inability to deal with multiple operating conditions. Also, the output predictions are determined only once, so changes in possible future control moves are not accounted for during the control move determination step.

Parametric models that involve the integration of differential and algebraic equations more accurately predict the future output of the process and can account for different operating conditions. Each time the control move determination step is performed, a new set of output predictions is determined, accounting for changes in the possible future control moves. In the past, this made both the prediction calculation and the control move calculation computationally slower. The speed of computers has improved greatly in the past decade, however, so a slower prediction step does not affect the performance of MPC algorithms using parametric models.

In this work, a parametric model is used to determine the output predictions of the state variables of a process. More details about the formulation of the specific prediction step are given in Chapter 4.

2.2.2 Correction

The correction step of MPC algorithms represents the area where the most research is being performed to improve MPC algorithms. Eaton *et al.* [20] point out that any practical control algorithm, including traditional PID-type algorithms and MPC algorithms, must include feedback to account for imperfect modeling. The correction step attempts to correct for discrepancies between the current value of the actual process output and the value of the model predicted output at the current time. The discrepancies arise from a number of sources, including mismatch between the process and the model, unmeasured process disturbances, sensor failure, and measurement signal noise. In this work, the discrepancies will be referred to as feedback, which is defined as the difference between the actual process output and the value of the model predicted output at the current time:

$$FB(k) = y(k) - y_p(k). \tag{2.2}$$

The correction step of MPC algorithms introduces feedback into the control algorithm. If the model of the process perfectly represents the actual process and all process disturbances are accounted for, the MPC algorithm requires no feedback information and can operate in an open loop configuration. Unfortunately, accurate models are not easily obtained, unmeasured process disturbances exist, and measurement signals usually contain noise. Ignoring these sources of feedback information causes MPC algorithms to have steady state offset between the actual value of the process

outputs and the desired value (the setpoints) of the process outputs. In PID-type control algorithms, integral action eliminates steady state offset. To introduce integral action in MPC algorithms, feedback information must be incorporated into the algorithm.

Historically, MPC algorithms have dealt with the discrepancies between the model and the process by adjusting the output predictions with a correction factor calculated as the difference between the process measurement and the predicted value of the output at that time:

$$d(k+l) = y(k+l) - y_p(k+l) \tag{2.3}$$

where y is the actual process variable and y_p is the model predicted value for $l = 1, 2, 3, ...P$. Future values of the process output, $y_p(k+1)$, $y_p(k+2)$, ..., $y_p(k+P)$, are not available, so the discrepancies are calculated based on the value of the current process output:

$$d(k+l) = y(k) - y_p(k) \tag{2.4}$$

where $y(k)$ is the current value of the process output measurement and $l = 1, 2, 3, ...P$. The definition of $d(k+l)$ for $l = 1, 2, ..., P$ is the same as that of $FB(k)$ given in Equation 2.2.

DMC-type algorithms adjust the output predictions by simply adding the value of the difference determined above:

$$y_p^*(k+l) = y_p(k+l) + d(k+l) \tag{2.5}$$

where y_p^* is the corrected value of the output prediction and $l = 1, 2, 3, ...P$. The assumption is made that the value of the difference term remains

constant for all future values of the predictions. Because future values of process output are not known, this assumption is the best one that can be made.

The same sort of prediction correction can be applied to other types of MPC algorithms, such as those with nonlinear parametric models. In this work, a method is developed that can determine the source of the discrepancies between the measured output variable and the model predicted value of the output. If the source of discrepancy comes from mismatch between the model and plant, parameters in the model (such as heat transfer coefficients or heat capacities) are corrected. If the source of the discrepancy is due to unmeasured process disturbances, a bias term similar to the discrepancy term given in Equation 2.4 is determined and added to the model prediction as in Equation 2.5.

Lee and Sullivan [42] describe a Generic Matrix Control (GMC) that uses nonlinear process models. The GMC algorithm attempts to determine the source of the discrepancy between the model output and the process output. The algorithm consists of two steps: 1) model update, and 2) control action calculation. Step 1 provides feedback to the controller to account for the current operating condition of the process. Lee and Sullivan [42] point out that correcting for process-model mismatch is important because mismatch can cause excessive overshoot or slow response to control action. Lee and Sullivan apply the GMC algorithm to a distillation column and demonstrate how updating model inaccuracies improves the effectiveness of

the MPC algorithm.

The method used in this work for determining the source of the discrepancies between plant and model predicted outputs, called feedback cultivation, will be presented in Chapters 3 and 4. Eaton *et al.* [20] attempt to determine the sensitivity of the MPC algorithm to changes in model parameter values to determine which model parameters have the greatest effect on the solution of the optimization. They do not explain how this information is incorporated into the MPC algorithm to eliminate process-model mismatch. In this work, model parameter sensitivity analysis will be used to narrow the number of model parameters included in the feedback cultivation analysis.

2.2.3 Control Move Determination

Determining the future control moves with an MPC scheme involves some sort of optimization algorithm. The type of optimization algorithm ranges from simple, least squares techniques to complex, nonlinear programming techniques.

An example of the simplest form is the control law of an unconstrained DMC algorithm written in closed form [19,53,57,55]:

$$\Delta u = (A^T A)^{-1} A^T e \tag{2.6}$$

where $(A^T A)^{-1} A^T$ = the inverse of the dynamic control matrix, e = a vector of errors defined as the difference between the corrected, predicted outputs

and the desired setpoint of the outputs:

$$e(k+l) = y_{set}(k+l) - y_p^*(k+l) \tag{2.7}$$

for $l = 1, 2, 3, ...P$. The vector of control moves, Δu, usually must be solved for using a least squares technique because the size of the resulting vector on the right side of Equation 2.6 is larger than the Δu vector.

Equation 2.6 represents an MPC scheme where the model of the process is explicitly inverted. The model of the process, the dynamic matrix A, appears in the equation in the inverted form A^{-1}.

The next level of complexity above an unconstrained DMC scheme involves the addition of constraints on the control moves and process outputs in the form of upper and lower limits. For example, a control valve can control a flow rate between zero (when the valve is fully closed) and some maximum (when the valve is fully open). The resulting constrained MPC scheme is called Quadratic DMC (QDMC), which solves for the control moves using quadratic programming techniques [59,27]. The optimization for QDMC appears in the following form:

$$\begin{aligned} \text{Min} \quad & \tfrac{1}{2}[A\Delta u - e]^T[A\Delta u - e] \\ \Delta u & \\ \text{s. t.} \quad & u_{low} \leq u \leq u_{high} \\ & y_{p,low} \leq y_p \leq y_{p,high}. \end{aligned} \tag{2.8}$$

The control moves, Δu, are the search variables, and the objective function is a squared difference between terms similar to those in Equation 2.6.

A linear DMC (LDMC) algorithm is similar to QDMC except that the objective function is formulated so that a linear programming technique such as the Simplex method can be used to solve for the future control moves [10,1,52,11,8]:

$$\begin{aligned} \text{Min} \quad & |A\Delta u - e| \\ \Delta u & \\ \text{s. t.} \quad & u_{low} \leq u \leq u_{high} \\ & y_{p,low} \leq y_p \leq y_{p,high}. \end{aligned} \tag{2.9}$$

The most general form of the optimization involves a nonlinear objective function and a nonlinear model [36,17,9,37,38,39,2]:

$$\begin{aligned} \text{Min} \quad & \Phi \\ u & \\ \text{s. t.} \quad & \dot{x} = f(x,u,p,d) \\ & y_p = g(x,u,p,d) \\ & u_{low} \leq u \leq u_{high} \\ & y_{p,low} \leq y_p \leq y_{p,high} \end{aligned} \tag{2.10}$$

where $f(x,u,p,d)$ and $g(x,u,p,d)$ represent the nonlinear differential and algebraic equations of the model that depend on the model state variables, x, the control moves, u, the model parameters, p, and the process disturbances, d. Using nonlinear models in an MPC scheme requires the use of nonlinear programming techniques to solve for the future control moves. Two nonlinear programming algorithms exist in a form suitable for computer programming: successive quadratic programming (SQP) and generalized reduced gradient

(GRG) [3]. Both schemes are available as pre-programmed subroutines in computer software packages. The specifics of the nonlinear optimal control problem are given in Chapter 4.

Biegler [6] used an SQP algorithm to solve for future control moves in a nonlinear MPC scheme for a stirred-tank reactor. Jang [36] used a GRG scheme in a nonlinear MPC algorithm for several chemical processes. In this work, the GRG scheme is used.

Unlike the simple DMC control law shown in Equation 2.6, the model inversion in all other MPC algorithms is implicit. This makes it difficult to analyze the properties of MPC algorithms theoretically and compare the properties to those known about traditional PID-type algorithms. Zafiriou [73] provides a theoretical framework for the study of the properties of control algorithms that are based on the on-line optimization of some objective function subject to constraints, i.e. MPC algorithms. The framework is based on the Operator Theory Approach [21].

2.3 MPC Tuning Methods

To control a specific plant or process MPC algorithms must be tuned in the same fashion that traditional PID algorithms must be tuned. Instead of controller gain, integral time, and derivative time – the tuning parameters of PID controllers – MPC algorithms use the prediction horizon and the control horizon as tuning parameters.

Zafiriou [73] states that a drawback of MPC algorithms is a lack of exact tuning procedures for the MPC tuning parameters. Because applications of MPC algorithms are still few in number, only general rules of thumb exist for specifying the values of the prediction horizon, the control horizon, and other MPC tuning parameters. Fortunately, it seems that ballpark figures for these parameters appear to be all that is necessary to achieve desirable performance from MPC algorithms.

The prediction horizon dictates how far into the future the model predicts the process output. Typically, the prediction horizon is set to a value equal to or greater than the time constant of the process, or, in other words, the length of time the process requires to reach steady state after a disturbance.

The control horizon represents how far into the future the control actions are determined. The control horizon is typically set lower than the prediction horizon. The control horizon needs to be large enough to prevent excessive variations in the control moves yet small enough to keep the optimal control problem solution from becoming too time-consuming.

Other methods of tuning exist for the nonlinear optimal control algorithm. These methods involve weighting factors for the terms in the objective function. The objective function of an MPC algorithm is optimized (maximized or minimized) using the future control moves as the search variables of the optimization. A commonly used form of the objective

function involves output and control move terms:

$$\Phi = \sum_{i=1}^{P} \beta_i (y_i - y_{set})^2 + \sum_{j=1}^{M} \gamma_j (u_j - u_{j-1})^2 \tag{2.11}$$

where β_i, $i = 1, ..., P$ are weighting factors of the output terms and γ_j, $j = 1, ..., M$ are weighting factors for the change in control moves. Equation 2.11 is shown here in discrete form as it is implemented in a computer program. The summation indices, P and M, represent the prediction and control horizons, respectively.

The output weighting factors, β_i, $i = 1, ..., P$, can be used to rate the importance of the outputs in a multivariable control scheme. Also, β can be used to weight early output predictions (i near 0) more or less heavily than later predictions (i near P). Most commonly, a constant value of β over the range of the prediction horizon is used.

The weighting factors for the change in control moves, γ_j, $j = 1, ..., M$, introduce move suppression to the nonlinear optimal control algorithm. Without move suppression, MPC algorithms tend to correct for process disturbances too quickly, causing excessive oscillation of the process outputs. Another method of implementing move suppression in an MPC algorithm involves increasing the control horizon. The values for γ and the control horizon must be balanced. High values for γ allow for smaller control horizons, which means a faster solution to the optimal control problem but a more sluggish controller response to process disturbances. Lower values for γ lead to longer control horizons, which slows down the solution of the optimal

control problem. This balance between the values of γ and the control horizon is process specific, and the determination of the balance is a matter of trial and error.

The output weighting factors, γ_j, $j = 1, ..., M$, can also be used to weight the importance of control moves in a multivariable MPC algorithm.

2.4 Summary

A brief history and definition of model predictive control has been given in this chapter. In Chapter 4, a specific description of the nonlinear optimal control MPC algorithm used in this work will be given. More extensive information on any MPC methods other than the nonlinear MPC algorithm may be found in [29,12,31,28,64,16,32,6,54].

CHAPTER 3

System Cultivation

Accounting for unmeasured process disturbances, correcting process-to-model mismatch, and eliminating measurement noise are areas of considerable research for model predictive control (MPC) algorithms. In this work, application of a relatively new perspective called system cultivation addresses these problems. Farell [22,23] describes system cultivation as a set of tools capable of extracting subtle but important relationships from a historical process database. The historical database consists of a time series of process variable measurements, controller actions, and other data that may contain useful information about the condition of the system, which includes both the process and controller.

System cultivation includes tools such as hypothesis feedback modeling, feedback cultivation, neural networks, and signal filtering. This work uses feedback cultivation with neural networks to detect and diagnose process-to-model mismatch and unmeasured process disturbances. Adaptive filtering is used to remove noise from the process output and disturbance measurements. This chapter first summarizes the work done by Farell in the areas of hypothesis feedback modeling and neural networks. Then feedback cultivation and adaptive filtering are described.

3.1 Hypothesis Feedback Modeling

The development of system cultivation began with hypothesis feedback (HF) modeling. HF-modeling attempts to extract subtle information from patterns of variation in an available historical database [22,23]. In many chemical plants today, large, time-sequenced historical databases are accessible through computer data logging and storage. To understand how well a process operates, HF-modeling considers the overwhelming amount of data and the many interacting variables and relationships in time-sequenced data.

Moore [47,48,46] introduced HF-modeling to analyze patterns of variation in multivariable processes keeping in mind the fundamental ideas of quality control: to detect abnormal patterns and identify the cause, and to strive continually to improve the normal pattern of operation. The basic concepts of HF-modeling are described below. Farell [23] gives a complete description of HF-modeling along with two practical applications.

Suppose a system model can be given by the equation

$$Y = f_n(x_m(1), ..., x_m(n1), x_u(1), ..., x_u(n2)) \tag{3.1}$$

where the function f_n depends on the state of the process, x_m is a set of $n1$ measured system variables, and x_u is a set of $n2$ unmeasured system variables. Geometrically, $(Y, x_m(1), ..., x_m(n1), x_u(1), ..., x_u(n2))$ defines a surface in the $n1 + n2 + 1$ dimensional space. The process itself determines the shape of that surface. Each system inefficiency can be identified by its effect on this surface. Because the unmeasured variables, x_u, are not

accessible, only the "viewing space" [46], $(Y, x_m(1), ..., x_m(n1))$ is available for analysis. HF-modeling seeks to characterize process inefficiencies and unmeasured disturbances by relating them to patterns in the viewing space.

The first step in HF-modeling is to define the hypothesis models for the process in the form

$$Y_{dep} = f(x_m(1), ..., x_m(n1)) \tag{3.2}$$

where Y_{dep} are system dependent variables (outputs). The measured variables, x_m, are important independent system variables (inputs) and important interaction terms. Examples of important system variables include feed flowrates, feed temperatures, heating or cooling medium inlet conditions, and others. An example of an interaction term involves combining a feed flowrate, F, with a feed temperature, T_i, to derive a measure of the energy entering the process in the feed stream:

$$E_i = F * T_i. \tag{3.3}$$

Hypothesis models monitor mass, energy, and momentum balances around a system. For example, the hypothesis model for a shell-and-tube heat exchanger derived by Farell [22] has the form

$$F_c = f(F * T_i, T_{ci}, T_{amb}) \tag{3.4}$$

where F_c is the coolant flowrate that serves as the controller manipulated variable used to keep the exchanger outlet stream at a constant temperature. The HF-model in Equation 3.4 indicates that F_c depends on the energy of the

feedstream, $F * T_i$, the inlet coolant temperature, T_{ci}, and the ambient temperature, T_{amb}. System models rely on both theoretical models and engineering judgement.

The second step in HF-modeling is to define the resolution of each variable in the hypothesis model. Defining this resolution reduces the precision of each variable, breaking down the entire span of each variable into increments such that the process operation can be considered reasonably constant within each increment. Geometrically, the $n1$-dimensional space consisting of x_m is broken down into regions of hypercubes. For example, the heat exchanger HF-model in Equation 3.4 requires the coordinates of this space to be $F * T_i$, T_{ci}, and T_{amb}. The regions defined by each hypercube are referred to as constant operating conditions.

Once the hypothesis model and the resolution of the system variables have been defined, data in the historical database can be sorted into groups of constant operating conditions. The result of the sorting is called the reduced database. The reduced database contains statistics and relationships concerning important system variables represented within each operating condition. Each line of data contains an operating condition address, averages and standard deviations of important dependent system variables and manipulated variables, and approximate partial derivatives monitoring important system relationships. The reduced database contains important information in an easy-to-interpret, enhanced, yet compressed form.

Once the reduced database is available, the information can be

presented in graphical form through multivariable statistical plots and relationship plots. These graphs examine cross sections of the viewing space. Farell [23] demonstrates the use of multivariable statistical plots and relationship plots to diagnose system problems for a shell-and-tube heat exchanger and a stirred-tank reactor.

3.2 Neural Networks

The reduced database resulting from HF-modeling contains the patterns of variation of a system in an enhanced and compressed form. A pattern classification tool such as a neural network can be used to interpret these patterns of variation automatically.

Neural networks are massively parallel computational models designed to emulate what is known about biological neural systems. Because of their massively parallel structure, neural networks store information in a distributed-associative memory. A distributed memory stores each piece of information partially across all memory units. An associative memory's output corresponds to the nearest match stored in memory, even when presented with a partial or incomplete input. These distributed-associative memories can generalize, giving reasonable responses when presented with noisy or incomplete data, and are fault tolerant to memory damage. The result of using distributive-associative memory is that neural networks learn by example much like a child learns from experience.

Rosenblatt introduced the perceptron neural network algorithm in 1957 based on a reward-and-punishment concept of learning [61]. During the 1950s and 1960s, Rosenblatt and many others developed neural networks as pattern classifiers for a host of applications. In 1969, Minsky and Papert [44,45] published results demonstrating that neural network applications were limited to linearly separable problems. Following this publication, work with neural networks dropped off until 1979. In 1979, Rumelhart, McClelland, *et al.* [62,63] presented a solution to the credit assignment problem that had limited neural network applications to linearly separable problems. Since then, neural network applications have grown tremendously. Entire conferences and journals dedicated to neural networks now exist.

Potential and currently achieved applications of neural networks in chemical engineering include signal processing, adaptive filtering, fault detection and diagnosis [34,23], interpretation of spectral data, and nonlinear steady-state and dynamic modeling [5,4,67]. Many forms of neural networks have been proposed. In this work, a multilayer perceptron network with back-propagation learning is used.

3.2.1 Neural Network Structure

Although many different forms of neural networks exist, every network contains nonlinear processing elements sometimes called neurons. Figure 3.1 illustrates a typical nonlinear processing element. As indicated in the figure, the neuron output equals a nonlinear function of the weighted sum of the

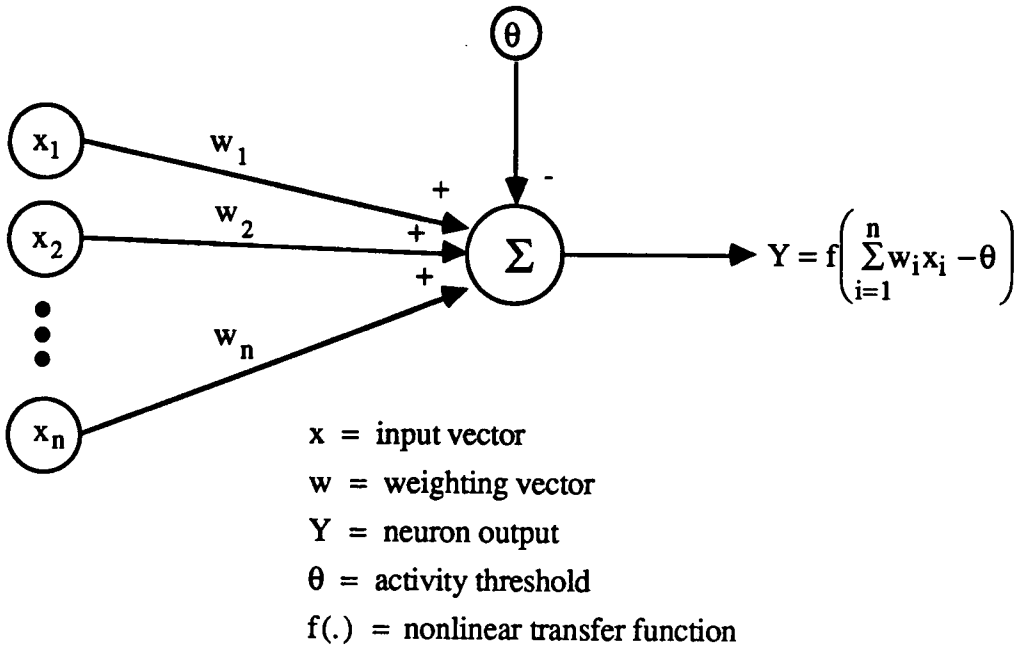


Figure 3.1: General nonlinear processing element or neuron

neuron inputs. Neural networks consist of many neurons interconnected in a massively parallel fashion much like the neurons in the human brain are connected via synapses.

The various forms of neural networks differ from one another in three areas:

1. the learning rule that adjusts the interconnection weights,
2. the structure of the interconnections between the neurons, and
3. the nature of the nonlinear transfer function of the neurons.

These differences give each type of neural network unique characteristics. The following sections discuss each of the three aspects and present an

example of the form for the perceptron back-propagation network. The perceptron neural network is used in this work as a pattern classifier.

3.2.1.1 The Learning Rule

To teach the network a desired pattern, the weights of the neuron interconnections of a neural network must be adjusted. The method of adjustment is called the learning rule. Two types of learning rules are used to train neural networks: supervised and unsupervised. Supervised learning adjusts the interconnection weights based on the value of the difference between the actual network and the desired network output. Unsupervised learning uses only input data and local network information to adjust the weights of interconnections. Unsupervised learning is used mainly to group data into unique and meaningful clusters.

The perceptron back-propagation neural network uses a form of supervised learning called the delta learning rule. The delta learning rule adjusts the weights of the neuron interconnections using a gradient descent optimization method. The delta learning rule used by the perceptron neural network is

$$\Delta w_{ij,new} = \alpha * e_j * x_{ij} + \beta * \Delta w_{ij,old} \tag{3.5}$$

where $\Delta w_{ij,new}$ equals the new value for the change in the weight of the interconnection between the i-th input to neuron j, $\Delta w_{ij,old}$ is the previous change in the weight, x_{ij} is the value of the i-th input, and e_j is the value of the error for neuron j.

The delta learning rule has two adjustable learning parameters: α and β. Parameter α, usually varying between 0 and 1, affects both the stability and the rate of convergence of the optimization. Lower values of α lead to convergence of the weights that is more stable and less oscillatory but slower. The parameter β is a momentum term used to moderate the change in the interconnection weights. The perceptron network used in this work does not include a momentum term.

The method for determining the value for the error e_j is clear for the top layer of the network. The error for the top layer is defined as a function of the difference between the desired neuron output and the actual neuron output:

$$e_j = y_j(1 - y_j)(d_j - y_j) \tag{3.6}$$

where y_j is the actual top layer neuron output and d_j is the desired output of the top layer neuron.

The method for calculating the error for the neurons in the middle and bottom layers of the network is not as clearly defined as for the top layer. The difficulty of calculating the middle and bottom layer errors is the credit assignment problem solved by Rumelhart [63] in 1979. The error of each neuron in the middle and bottom layers of the network is a weighted sum of the errors in the next layer above:

$$e_j = x_{ij}\overline{(1 - x_{ij})}_k \overline{\epsilon}_k \overline{w}_{ik} \tag{3.7}$$

where i refers to the layer above the layer containing neuron j. Subscript k

means the average over all the neurons in layer i. The output of neuron j to layer i is x_{ij}. So, in the perceptron back-propagation network, the error "back-propagates" from the output layer to the input layer.

3.2.1.2 Neuron Interconnections

Three important features of a neural network are the nature of the neuron connections, the number of neurons in the network, and the number of layers of neurons in the network. The nature of the neuron connections determines network characteristics such as how information flows and how the network learns and stores information. The number of neurons and the organization of the neurons in layers dictates the network's ability to solve complex problems.

Each neuron in a network can be connected to any other neuron or to itself. Figure 3.2 shows that a neuron can propagate an output signal forward toward the output layer (feedforward connections), backward toward the input layer (feedback connections), laterally to other neurons in the same layer, and to itself. The propagated neuron signal can be excitory or inhibitory depending on whether it increases the neuron's activity or decreases the neuron's activity, respectively. The perceptron back-propagation neural network uses feedforward connections only. The feedforward signal is excitory or inhibitory depending on the sign of the interconnection weight determined by the delta learning rule.

The number of neurons and the number of layers in a network determine the ability of the network to solve complex problems. Farell [23]

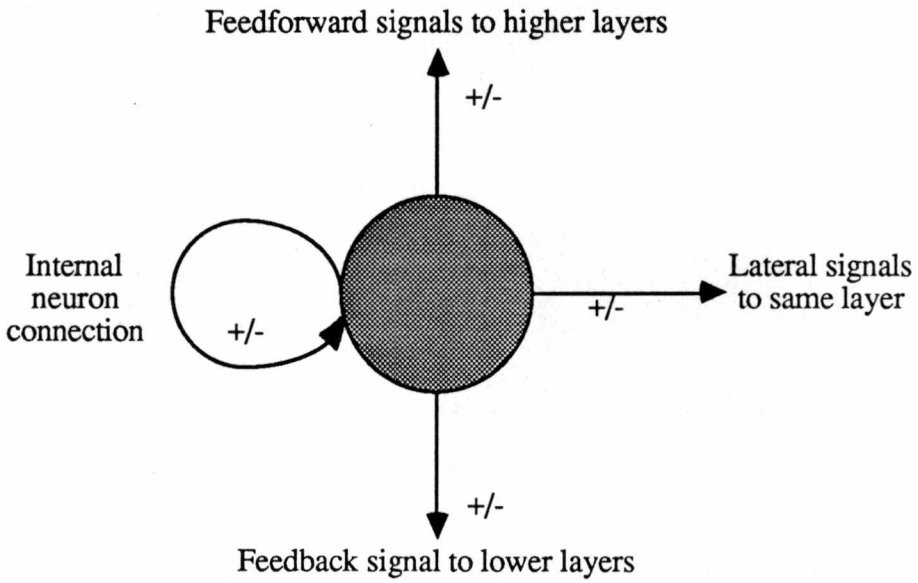


Figure 3.2: Possible neuron connections

explained in detail that a network's ability to draw nonlinear decision boundaries depends on the number of layers in the network and the number of neurons in each layer. Farell determined that having three layers of neurons improves the efficiency of the network in forming nonlinear decision boundaries. The three-layer network required fewer overall neurons than a two-layer network [23]. The perceptron back-propagation network used in this work contains three layers of neurons above the input layer. Figure 3.3 illustrates the construction of the perceptron back-propagation network.

The number of neurons in each layer of the network affects both the network's ability to learn the desired patterns and the speed of learning. The number of neurons in the input and output layers is specified by the network

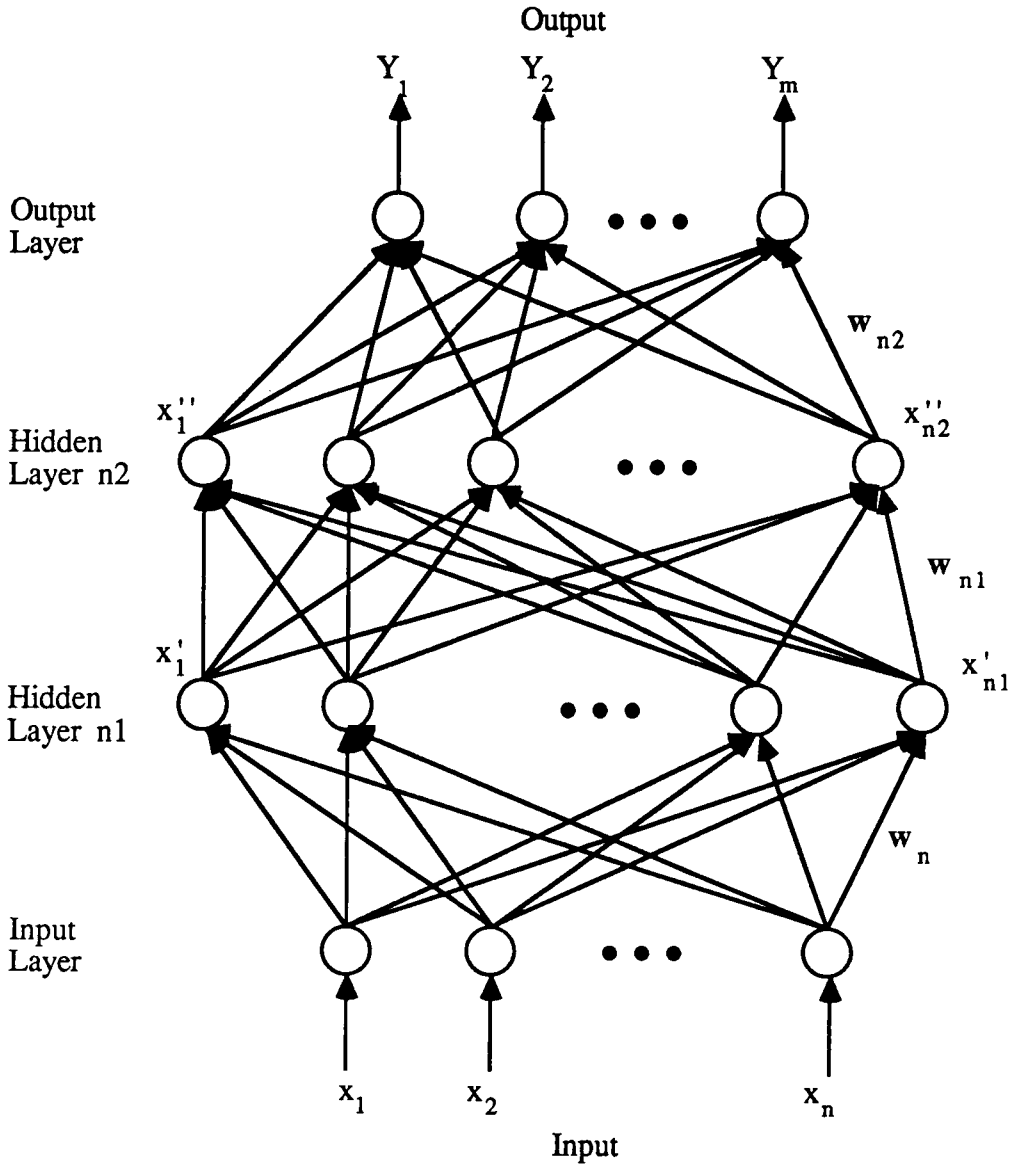


Figure 3.3: The three-layer perceptron back-propagation neural network

application. The perceptron back-propagation network contains two hidden layers of neurons. Specifying the optimal number of neurons for these two hidden layers is important. Having too many neurons results in a lookup table effect by the network and reduces the network's ability to generalize and interpolate. Having too few neurons prevents the network from adequately learning the desired patterns. Kolmorgorov [33] derived a theorem that states that a three-layer perceptron network with $N(2N+1)$ neurons can describe any function of N variables. This property can result in very large networks that are difficult to train. For most problems, using $2I+1$ (where I is the number of input neurons) hidden layer neurons distributed between the two hidden layers, works adequately.

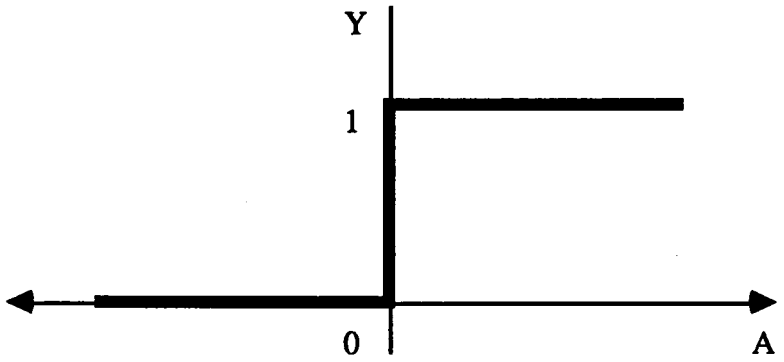
3.2.1.3 The Neuron Transfer Function

The main source of nonlinearities in a neural network comes from the type of transfer function used to calculate each neuron's output from the summation of neuron inputs. As shown in Figure 3.1, the output of a neuron is a function of the summation of neuron inputs:

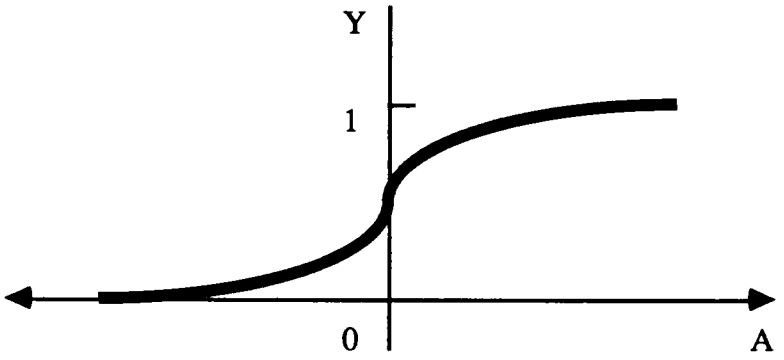
$$y_j = f\left(\sum_{i=1}^{n} w_{ij}x_{ij} + \theta\right) = f(S) \tag{3.8}$$

where n is the number of inputs to the neuron and θ is the neuron threshold value. The transfer function f represents the source of nonlinearities in the network.

The two most commonly used transfer functions are the hard-limiter and the sigmoid function. Figure 3.4 illustrates both of these transfer



Hard-Limiter Transfer Function



Sigmoid Transfer Function

$$Y = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right) = \text{neuron output}$$

$$A = \sum_{i=1}^{n} w_i x_i - \theta = \text{neuron activity}$$

θ = activity threshold

Figure 3.4: Nonlinear neuron output transfer functions

functions. The hard-limiter linearly separates the output into two regions. When the summation in Equation 3.8 is less than or equal to a specified value, the neuron output is set to 0. When the summation is greater than the specified value, the output of the neuron is set to 1.

The sigmoid function provides a continuous transition from 0 to 1 such that very low values of the summation result in a neuron output of 0 and very high values of the summation result in a neuron output of 1. The sigmoid function has the form

$$f(S) = \frac{1}{1 + e^{-S}} \tag{3.9}$$

where S is the summation of the interconnection weights times the neuron inputs. The advantage of the sigmoid function is that it is continuous and continuously differentiable. Differentiability is essential for the learning algorithm of some networks because the method used to determine the optimum interconnection weights requires the first and sometimes the second derivative of the transfer function. The perceptron back-propagation neural network uses the sigmoid form of transfer function because the first derivative of the transfer function is needed to perform the optimization to determine the interconnection weights.

3.2.2 Modes of Operation

Neural networks have two modes of operation: training and classifying. In the training mode, the learning rule adjusts the weights to recognize the patterns in the training datasets. When classifying, the network is subjected

to unknown datasets to determine what pattern the datasets most closely resemble.

3.2.2.1 Neural Network Training

Neural network training requires training datasets representative of each class of problem the network is being trained to distinguish between. For supervised learning, the training sets must have network inputs and the corresponding desired network output. Unsupervised learning requires only input data for each class of problem. The perceptron back-propagation network uses supervised learning and requires both input and desired output in the training datasets.

The training of the perceptron back-propagation neural network proceeds in the following steps:

1. Specification of the dimensions of the network including the number of input neurons, the number of hidden layer neurons in each of the two hidden layers, the number of output neurons (usually the number of classes of data), the learning rate (α in Equation 3.5), and the neuron threshold value (θ in Equation 3.9).

2. Initialization of all the neuron interconnection weights to random numbers between -1 and 1.

3. Presentation of an input vector $(x_1, ..., x_N)$ to the network and computation of the network's output.

4. Adjustment of the weights of the neuron interconnections using the delta learning rule in Equation 3.5 using Equations 3.6 and 3.7 in the determination of the value of the error used in Equation 3.5.

5. Repetition of steps 3 and 4 until the network error for all input vectors in the training datasets for all classes of data falls below a specified convergence threshold.

Steps 3 and 4 may have to be repeated thousands of times before convergence is reached. The training time of the perceptron back-propagation network depends on the amount of training data, the number of neurons in the hidden layers, and the complexity of the decision boundaries between the classes of data.

3.2.2.2 Network Classification

Once trained, a neural network can be used to classify patterns in unknown datasets. Given the unknown input vector, classification using the perceptron back-propagation network involves simple addition and multiplication to determine the output of the network. The output neuron with a value nearest to 1 determines the class that the input vector most closely represents. For unknown datasets with multiple input vectors, a percentage of the number of vectors in each class is determined for the dataset.

A common misconception about neural networks is that classification of unknown datasets is time-consuming. Classification is not to be confused

with training, which is a time-consuming operation for some forms of networks including the perceptron back-propagation network. Once trained, the perceptron network classifies very quickly because only addition and multiplication are required to compute the network output. Addition and multiplication are the fastest of all computer operations.

3.2.3 Summary

The perceptron back-propagation neural network works as an effective pattern classifier when properly set up and trained. Farell [23] demonstrates the network's general characteristics including the form of the decision boundaries it can draw and the effect of changing the network dimensions and learning parameters for a simple 2-input, 2-class problem. His findings demonstrate that the three-layer network is capable of drawing nonlinear decision boundaries between classes of data.

Farell [23] also used the perceptron neural network with HF-modeling to diagnose system problems for a shell-and-tube heat exchanger and a stirred-tank reactor. HF-modeling generated reduced datasets that were used as the network training datasets and unknown datasets. Farell [23] showed that using reduced datasets rather than raw, historical datasets greatly increased the network's speed of training and improved its ability to distinguish between different classes of system problems.

In this work, the perceptron neural network is used in conjunction with feedback cultivation to detect process-to-model mismatch and

unmeasured process disturbances. Feedback cultivation generates reduced datasets from historical datasets in a fashion similar to HF-modeling. The next section fully describes feedback cultivation.

3.3 Feedback Cultivation

Calculating feedback terms determines how well the model relates to the process or detects the presence of unmeasured disturbances. Feedback is defined as the difference between the actual process measurement of a variable and the value the model predicted for that variable:

$$FB(k) = y(k) - y_p(k) \tag{3.10}$$

where y equals the process measurement at the current time indicated descretely by k and y_p equals the model prediction of the measurement at time k. Notice that Equation 3.10 is the same as Equation 2.2. Values of y and y_p can be saved in the historical database along with control move values and output setpoints. The data in the historical database can be used in a new technique called feedback cultivation. Feedback cultivation works like hypothesis feedback modeling in that feedback data from the historical database is summarized and consolidated into a smaller, reduced database. The difference between feedback cultivation and HF-modeling is that feedback cultivation maintains the temporal (time-related) aspect of the feedback data. HF-modeling sorts the data in the historical database so that the time-related aspect of the data vanishes.

Applying feedback cultivation to a historical database results in a reduced feedback database that contains information about possible process-to-model mismatch and unmeasured disturbances. The reduced feedback database can be used in conjunction with a neural network to detect differences between process and model parameters and unmeasured disturbances. As explained in the previous section, a neural network requires training data to learn the difference between various classes of data corresponding to the different model parameters and unmeasured disturbances. The model parameters can be perturbed one at a time while collecting historical datasets representative of changes in each of the particular model parameters. The same procedure does not apply to unmeasured disturbances. If the variable is truly unmeasured, the model does not contain a variable that can be changed to generate a neural network training dataset. Ideally, unmeasured disturbances affect the process in a way unlike any mismatched model parameters so that the neural network will classify data corresponding to unmeasured disturbances in a class called NULL. A NULL classification indicates that the data does not represent anything the network has been previously trained to recognize.

Feedback cultivation determines summaries and statistics for blocks or windows of data in the historical database that relate the dynamics of the process and controller. Examples of the statistics include: average value of the feedback (FB) for each measured variable in the data window, standard deviation of FB in the window, median value of FB in the window, average

value and standard deviations of the control actions in the data window, and other statistics relating to the feedback terms, control moves, and controller error. Controller error is the difference between output setpoint values and output measurements of process variables. Summary variables include: the sum of squared feedback terms, the sum of square errors, the sum of absolute values of the feedback terms, the sum of absolute value of the errors, the fraction of feedback terms greater and less than zero, and other terms. Figure 3.5 shows definitions of some of these statistics and summaries.

The size of the data windows depends on the time constant of the process and the frequency of control actions. The data window should cover at least one time constant of the process and include a significant number of control moves so that the dynamics of the feedback are fully captured. To further capture the dynamics of the feedback, the windows of data should overlap each other by half the length of time covered by each window as indicated in Figure 3.6.

In this work, feedback cultivation is used with the nonlinear optimal control algorithm to detect process-to-model mismatch and unmeasured disturbances. Figure 3.7 shows a block diagram of the feedback cultivation algorithm. The algorithm involves three steps: 1) generating a reduced dataset from the available historical dataset, 2) identifying inaccurate model parameters or unmeasured disturbances using a trained neural network, and 3) updating inaccurate model parameters and bias terms. The first two steps were described in the preceding sections. The third step involves a modified

Statistics and Summaries Covering All Points in the Data Window

$\sum_{i=1}^{P}[FB_j(i)]^2$	$\sum_{i=1}^{P}\|FB_j(i)\|$
$\sum_{i=1}^{P}[error_j(i)]^2$	$\sum_{i=1}^{P}\|error_j(i)\|$
$\sum_{i=2}^{P}\Delta u_i^2$	$\sum_{i=2}^{P}\|\Delta u_i\|$
$\overline{FB_j}$	$s(FB_j)$
FB_j Range	Median FB_j
$\overline{\Delta u}$	$s(\Delta u)$
$\overline{error}$	$s(error)$
Fraction of $FB_j(i) > 0$	Fraction of $FB_j(i) < 0$

Statistics and Summaries Covering First Half and Second Half of Window

$\sum_{i=1}^{P/2}[FB_j(i)]^2$	$\sum_{i=P/2+1}^{P}[FB_j(i)]^2$
$\sum_{i=1}^{P/2}[error_j(i)]^2$	$\sum_{i=P/2+1}^{P}[error_j(i)]^2$
$\overline{FB_j(i)}$ $i \leq P/2$	$\overline{FB_j(i)}$ $i > P/2$

Variable Definitions

$$FB_j(i) = y_j(i) - y_{p_j}(i)$$

$$error_j(i) = x_{set_j}(i) - x_j(i) \qquad i = 1, 2, ...P$$

$$\Delta u_i = u(i) - u(i-1)$$

$s(x)$ = standard deviation of x

Figure 3.5: Examples of feedback cultivation statistic and summary terms

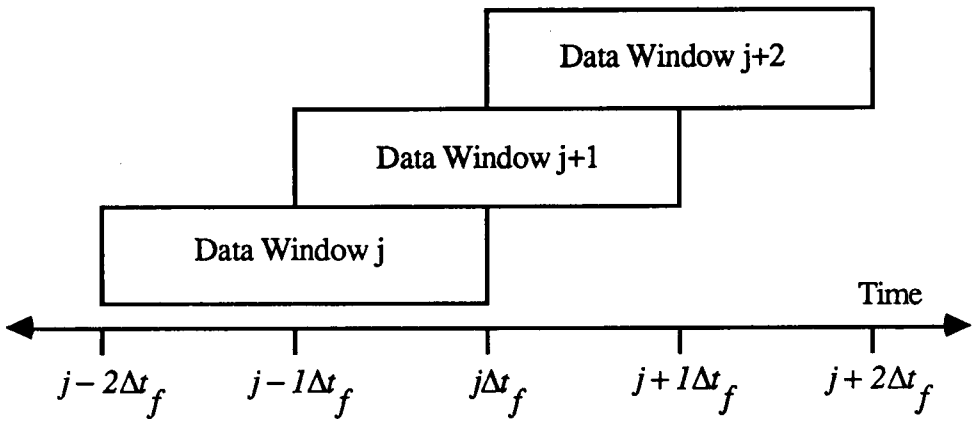


Figure 3.6: Overlapping data windows used in feedback cultivation

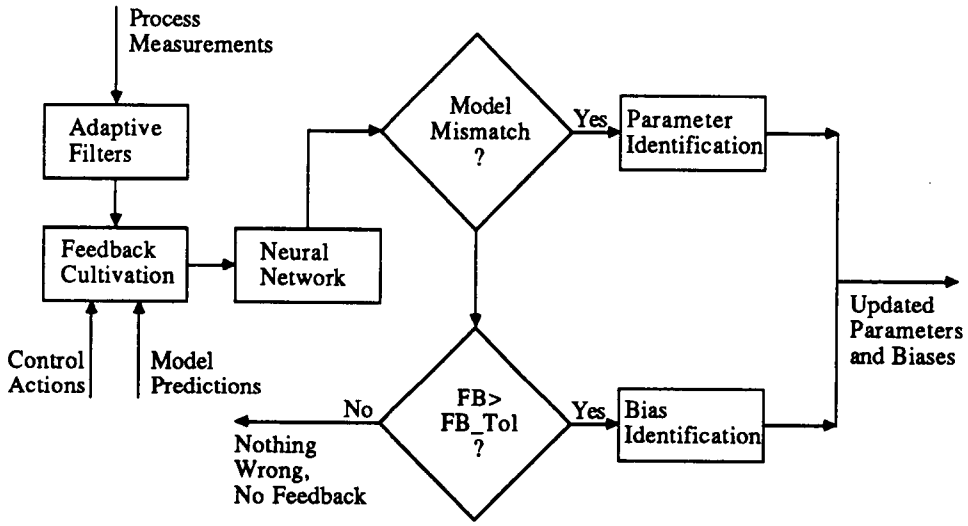
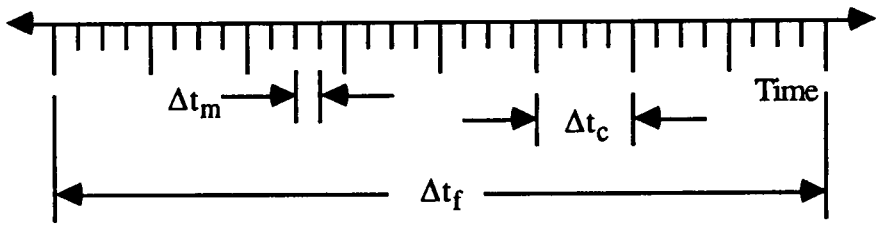


Figure 3.7: Block Diagram of the feedback cultivation algorithm

version of the nonlinear optimal control algorithm and is described in Chapter 4.

Updating the model parameters and bias terms adds feedback to the nonlinear optimal control algorithm. Without feedback, the algorithm contains no integral action. Integral action, in any control strategy, eliminates steady state offset between the output variables and their desired setpoints. Slow or gradual addition of integral action in the control algorithm prevents excessive oscillations of the process response to disturbances. In the nonlinear optimal control algorithm, the feedback cultivation scheme incorporates integral action. Feedback cultivation is performed after a number of control moves have been determined, simulating gradual addition of the integral action. To visualize this concept, Figure 3.8 shows a time line with three different time intervals indicated. The smallest time interval, Δt_m, represents the frequency of process variable measurements. The next largest time interval, Δt_c, represents the length of time between implementation of the nonlinear optimal control algorithm to calculate a new control move. The largest interval, Δt_f, represents the length of time between applications of feedback cultivation. New model parameters and bias terms may be updated after the presence of a problem has been confirmed. Confirmation of a problem requires several consecutive, positive identifications of the problem by the feedback cultivation algorithm.

Chapter 4 explains how feedback cultivation fits into the overall nonlinear optimal control scheme. Chapters 5 and 6 demonstrate how



Δt_m = process variable measurement interval
Δt_c = control move determination interval
Δt_f = feedback cultivation time interval

Figure 3.8: Time intervals of measurements, control action determination, and feedback cultivation

feedback cultivation works on a first-order process and model example and a stirred-tank reactor example, respectively.

3.4 Filtering

One problem associated with using process measurements to update the state and disturbance variables in the model is the possibility of noise corrupting the measurement signals. Common practice in the application of MPC algorithms involves filtering the measurements to reduce the effect of noise on the MPC algorithm. Many applications use a first-order exponential filter in

the discretized form. Widrow and Winter [70,72] suggest the use of adaptive filters to eliminate noise from the measurements. A description of first-order filters will be given in the following section followed by a more detailed description of adaptive filtering. Chapter 6 contains results comparing the performance of the nonlinear optimal control algorithm using first-order filtering and adaptive filtering.

3.4.1 First-order Filtering

A first-order exponential filter in discretized form involves a recursive term and a nonrecursive term:

$$\hat{y}_i = \alpha \hat{y}_{i-1} + (1 - \alpha) y_i \tag{3.11}$$

where $\hat{y}_i$ = the latest, filtered value of the measurement, y_i, $\hat{y}_{i-1}$ = the previous value of the filtered measurement, and α = the filter constant specified such that $0 \leq \alpha \leq 1$. The term $\alpha \hat{y}_{i-1}$ is the recursive term and the term $(1 - \alpha) y_i$ is the nonrecursive term. The value of the filter constant, α, depends on the time constant of the measurement being filtered or the overall time constant of the process. The filter constant is defined in terms of the filter time constant and the time interval of the measurement:

$$\alpha = e^{-\Delta t / \tau_f} \tag{3.12}$$

where Δt is the time interval of the measurement and τ_f is the time constant of the filter. A first-order exponential filter works best when the filter

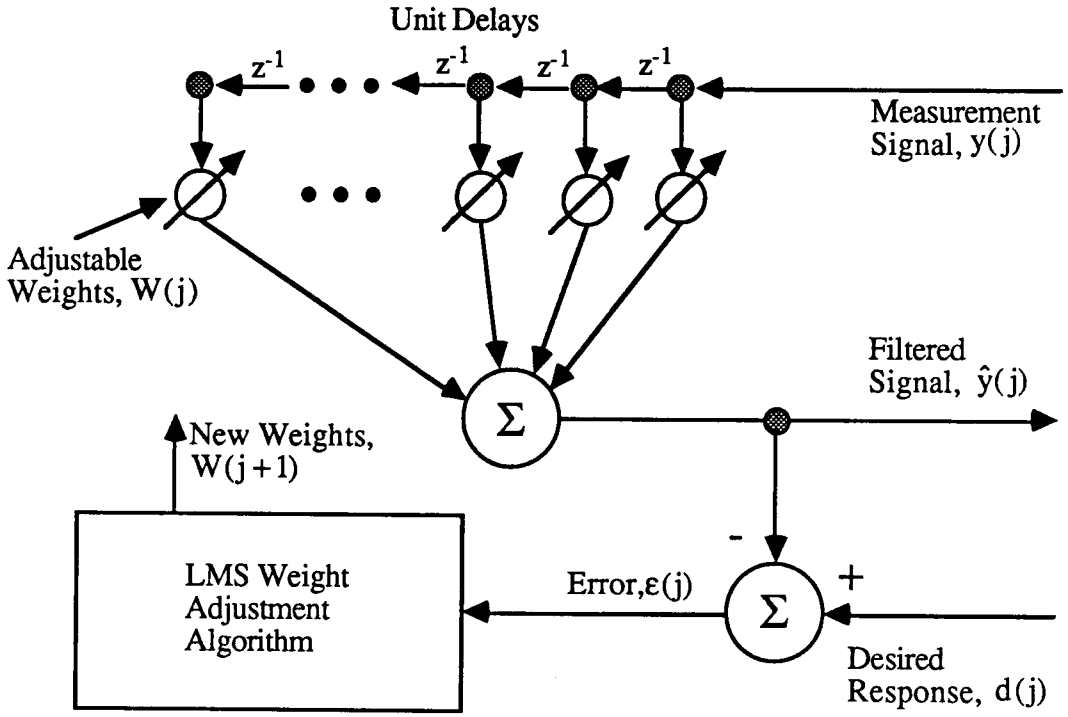
constant is specified such that the filter time constant is less than the process time constant.

Other than setting the filter time constant less than the process time constant, no other criteria exist for specifying the filter constant for first-order exponential filters. Typically, the filter constants are arbitrarily set using engineering judgement. Sometimes, this may not be adequate to effectively remove noise from a measurement signal. In this case, a filtering method that is less sensitive to the specification of one parameter would be desirable.

3.4.2 Adaptive Filtering

Adaptive filters have the form of an n-th order, nonrecursive z-transform with adjustable, instead of constant, coefficients. The filter considered consists of a tapped delay line, variable weights, a summer to add the weighted input signals, and an optimization algorithm to adjust the weights. Figure 3.9 schematically illustrates the adaptive filter used to remove noise from measurement signals.

The adaptive filter works like a neural network in that it has two modes of operation: training and operating. During training, a steepest descent form of optimization adjusts the weights to minimize a mean square error. The error is defined as the difference between the desired filter output and the actual filter output. In the operating mode, the filter sums the weighted inputs, using the weights from the training process, to form a filtered output signal.



LMS Algorithm: $W(j+1) = W(j) - 2\mu\varepsilon(j)Y(j)$

Filtered Output: $\hat{y}(j) = W(j)Y(j)$

Figure 3.9: Schematic of a discrete adaptive filter

The difference between the neural network described previously and the adaptive filter algorithm is that the adaptive filter algorithm cycles between the training and operating modes continuously. After determining a new filtered output, the training process adjusts the weights to be used the next time a filtered output is called for.

3.4.2.1 Adaptive Filter Training Mode

The weights of the adaptive filter are adjusted using a steepest descent optimization algorithm which adjusts the weights proportional to the gradient of the mean square error:

$$W(j+1) = W(j) + \mu\hat{\nabla}\left[\bar{\epsilon}^2(j)\right] \tag{3.13}$$

where W_{j+1} = the weight vector of length n after adaptation, W_j = the weight vector before adaptation, and μ is a scalar constant controlling the rate of convergence and stability.

An estimate of the gradient of the mean square error, $\hat{\nabla}\left[\bar{\epsilon}^2(j)\right]$, needs to be determined to use Equation 3.13 to adjust the weights. Widrow [68] derives the estimated gradient from the gradient of a single time sample of the squared error resulting in the following definition for the gradient estimate:

$$\hat{\nabla}\left[\bar{\epsilon}^2(j)\right] = -2\epsilon(j)Y(j) \tag{3.14}$$

where the error is defined as

$$\epsilon(j) = d(j) - W^T(j)Y(j) \tag{3.15}$$

and $Y(j)$ is the vector of inputs to the adaptive filter. Substituting Equation 3.14 into 3.13, the weight iteration function becomes

$$W_(j+1) = W_j - 2\mu\epsilon(j)Y(j) \tag{3.16}$$

which Widrow refers to as the Least Mean Square (LMS) algorithm [68].

To use the LMS algorithm given in Equation 3.16, the desired output of the adaptive filter, $d(j)$, is required to compute the error as shown in Equation 3.15. Of course, the ideal desired output would be the value of the measurement signal without noise. If the value of the measurement signal without the noise was available, filtering would not be necessary. Widrow [68] describes a method to cope with this situation that uses a delayed value of the error:

$$\epsilon = d(j-\delta) - W^T(j-\delta)Y(j-\delta) \tag{3.17}$$

and then setting the desired output equal to the latest measurement of the signal: $d(j-\delta) = y_1(j)$ where $y_1(j)$ is the latest, undelayed measurement.

In this work, a variation of the method suggested by Widrow to determine the error of the adaptive filter is used. Since a number of process signal measurements are made in between each control move determination (see Figure 3.8), the desired filter output is set equal to the average of the measurements between control moves:

$$d(j) = \sum_{i=1}^{K} y_i(j)/K \tag{3.18}$$

where K is the number of process measurements between control moves.

Using the desired filter output given by Equation 3.18, the LMS algorithm given in 3.16 can be used to adjust the weights of the adaptive filter.

3.4.2.2 Adaptive Filter Operating Mode

The output of the adaptive filter is the weighted sum of the previous n values of the measured variable:

$$\begin{aligned} \hat{y}(j) &= w_1(j)y(j) + w_2(j)y(j-1) + \cdots + w_n(j)y(j-n+1) \\ &= W^T(j)Y(j) \end{aligned} \tag{3.19}$$

where $\hat{y}(j)$ is the filtered value of the measurement signal. The weights in vector $W(j)$ are the latest values determined by the LMS algorithm. Once the filtered value of the output is determined, the adaptive filter goes into training mode to determine the optimum weights for the next time a filtered output is calculated.

3.4.2.3 Adaptive Filter Parameters

The adaptive filter algorithm has two parameters that must be specified: the number of weights, n, and the LMS convergence constant, μ. These parameters do not directly affect the output of the filter the way the first-order exponential filter constant affects the filter output in Equation 3.11. Therefore, specification of these parameters is not as critical to the operation of the adaptive filter.

Widrow [68] explains that the convergence constant, μ, depends on the maximum eigenvalue of the correlation matrix formed by the input vector,

$Y(j)$. Since the input vector changes every time the adaptive filter algorithm computes a new filtered output, it is not clear how to determine the maximum eigenvalue of the correlation matrix. For stability of the steepest descent optimization algorithm, a value of μ between 0 and 1 will work. The higher the value of μ, the faster the LMS algorithm converges to new values of the weights, but the convergence will become more oscillatory. Too high a value of μ will cause instability in the LMS algorithm.

Specification of the number of weights depends on the nature of the measurement signal. Widrow [70,69,68,71] gives several examples where he uses between 5 and 10 weights. Selecting an appropriate value for n is a matter of trial and error. The lower the value of n, the faster the LMS algorithm converges to new weights. However, dynamics of the measurement signal may not be adequately captured with too low a value of n.

3.4.3 Summary

The previous sections explain two types of measurement signal filtering: first-order filtering and adaptive filtering. In this work, adaptive filtering is used primarily. Both process disturbance measurements and output measurements are filtered. Before the nonlinear model of the process is updated with new values of the disturbance measurements, each signal passes through an adaptive filter. Each output variable measurement passes through an adaptive filter before storage in the historical database for use in feedback cultivation.

CHAPTER 4

The Control Algorithm

The complete control algorithm used in this work includes adaptive filtering, feedback cultivation, neural networks, parameter re-identification, and nonlinear optimal control. This chapter first describes the nonlinear optimal control technique used to determine the new control actions and to re-identify model parameters. Then, the methodology of the complete control system is described.

4.1 Nonlinear Optimal Control

This dissertation uses a nonlinear optimal control algorithm based on the principles of optimal control or the Pontryagin Maximum Principle [56]. Bryson and Ho [9] explain the theory of optimal control in detail. Balchen and Mummé [2] state that of all the methods that provide a systematic derivation of a control system structure, optimal control is the most generally applicable. A control method based on optimal control theory leads to a multivariable feedback system that usually minimizes a quadratic objective function.

In this work optimal control theory applies to two situations. First, the nonlinear optimal control algorithm determines the future control moves

for the process. Second, a modified version of the nonlinear optimal control algorithm solves for appropriate model parameters when re-identification is called for by feedback cultivation. Solving the nonlinear optimal control problem for the future control moves will be explained first.

4.1.1 Control Move Determination

A nonlinear process can be modeled by differential equations determined by performing mass, energy, and momentum balances on the process, resulting in a general model of the form

$$\dot{x} = f(x, u, p, d) \tag{4.1}$$

$$y_p = g(x, u, p, d) + bias \tag{4.2}$$

where $\dot{x}$ = a vector of state variable derivatives, x = a vector of process state variables, u = a vector of control moves, p = the model parameters, d = process disturbances, and $f(x, u, p, d)$ = the nonlinear functions representing mass, energy, and momentum balances. The nonlinear functions $g(x, u, p, d)$ translate the state variables, x, to model predicted output variables, y_p. The term *bias* in Equation 4.2 accounts for unmeasured process disturbances. Chapter 3 described how feedback cultivation can be used to detect unmeasured disturbances, and Section 4.1.2 describes a technique for determining the values for *bias*. The nonlinear optimal control problem that incorporates the model given in Equations 4.1 and 4.2 stated as a

minimization is

$$\begin{aligned} &\underset{u}{\text{Min}} \quad \Phi \\ &\text{s.t.} \quad \dot{x} = f(x,u,p,d) \\ &\qquad y_p = g(x,u,p,d) + bias \end{aligned} \tag{4.3}$$

where Φ = the objective function. The control moves, $u = (u_1, u_2, ..., u_M)$, represent the search variables of the optimization where M is the control horizon. The equations of the model shown in Equations 4.1 and 4.2 are the constraints of the optimization. Because the process is continuous, the continuous form of the model appears in the optimal control problem even though the process measurements and control actions are discrete. A discrete form of the optimal control problem is used in computer programmed applications.

The nature of the objective function, Φ, can be based on things such as economic factors, product specifications, desired setpoints of state variables, or a combination of all these. In optimal control the objective function is usually stated as two terms:

$$\Phi = S(y_f) + \int_{t_0}^{t_f} L(y_p, u)dt \tag{4.4}$$

The first term, $S(y_f)$, represents criteria based on the final value of the state variables, for example, specification of the desired final concentration of the product leaving a reactor. The second, integrated term accounts for intermediate criteria specified between the initial and final times considered in the optimization. For example, $L(y_p, u)$ may equal the squared difference

between a process output variable predicted by the model in Equation 4.2 and the desired setpoint of that output variable:

$$L(y_p, u) = (y_p - y_{set})^2. \tag{4.5}$$

To solve Equation 4.3 for the future control moves, u, the constraints are adjoined with constant multiplier functions:

$$\Phi = S(y_f) + \int_{t_0}^{t_f} [L(y_p, u) + \lambda_1^T \{f(x, u, p, d) - \dot{x}\} + \lambda_2^T \{g(x, u, p, d) - y_p\}] dt. \tag{4.6}$$

Then, a scalar function called the Hamiltonian is defined as

$$H(x, \lambda_1, \lambda_2, u) = L(y_p, u) + \lambda_1^T f(x, u, p, d) + \lambda_2^T g(x, u, p, d). \tag{4.7}$$

Next, 4.7 is substituted into 4.6 and integrated by parts to yield

$$\Phi = S(y_f) - \lambda_1^T(t_f)x(t_f) + \lambda_1^T(t_0)x(t_0) + \int_{t_0}^{t_f} [H(x, \lambda_1, \lambda_2, u) + \dot{\lambda}_1^T x - \lambda_2^T y_p] dt. \tag{4.8}$$

Then, considering the variation of Φ due to the variations in the control vector u for fixed times t_0 and t_f gives

$$\begin{aligned} \delta\Phi = \; & \left[\tfrac{\partial S}{\partial y_p} \delta y_p\right]_{t=t_f} - [\lambda_1^T \delta x]_{t=t_f} + [\lambda_1^T \delta x]_{t=t_0} + \\ & \int_{t_0}^{t_f} \left(\left[\tfrac{\partial H}{\partial x} + \dot{\lambda}_1^T\right] \delta x + \tfrac{\partial H}{\partial u} \delta u + \left[\tfrac{\partial H}{\partial y_p} - \lambda_2^T\right] \delta y_p \right) dt. \end{aligned} \tag{4.9}$$

Choosing the coefficients of δx and δy_p to vanish gives

$$\dot{\lambda_1} = -\frac{\partial H(x, \lambda_1, \lambda_2, u)}{\partial x} \tag{4.10}$$

$$\lambda_2^T = \frac{\partial H(x, \lambda_1, \lambda_2, u)}{\partial y_p} \tag{4.11}$$

with boundary conditions

$$\lambda_1(t_f) = \frac{\partial S(x_f)}{\partial x} \mid_{t=t_f}. \tag{4.12}$$

Equation 4.9 then becomes

$$\delta\Phi = [\lambda_1^T \delta x]_{t_0} + \int_{t_0}^{t_f} \frac{\partial H}{\partial u} \delta u dt. \tag{4.13}$$

The first term in 4.13 is constant or zero because x_0 is given. For an extremum, the derivative of Φ with respect to u must be zero, and this can only happen if

$$\frac{\partial H(x, \lambda_1, \lambda_2, u)}{\partial u} = 0 \tag{4.14}$$

or, from Equation 4.7

$$\left(\frac{\partial L}{\partial u}\right)^T + \left(\frac{\partial f}{\partial u}\right)^T \lambda_1 + \left(\frac{\partial g}{\partial u}\right)^T \lambda_2 = 0. \tag{4.15}$$

To find the solution of 4.15, the following differential and algebraic equations must be solved:

$$\dot{x} = \frac{\partial H}{\partial \lambda_1} = f(x, u, p, d) \tag{4.16}$$

$$\dot{\lambda}_1^T = -\frac{\partial H}{\partial x} = -\lambda_1^T \frac{\partial f}{\partial x} - \lambda_2^T \frac{\partial g}{\partial x} \tag{4.17}$$

$$\lambda_2 = \frac{\partial H}{\partial y_p} = \frac{\partial L}{\partial y_p}. \tag{4.18}$$

Two boundary conditions are required to solve Equations 4.16 and 4.17. Usually the boundary conditions are given as

$$x(t_0) = x_0 = given \tag{4.19}$$

$$\lambda_1(t_f) = \frac{\partial S(x_f)}{\partial x} \mid_{t=t_f} \tag{4.20}$$

resulting in a two point boundary value problem for the optimization problem stated in Equation 4.3.

Computer solution of the optimal control problem in Equation 4.3 requires discretization of the equation. Solving 4.3 discretely involves the following steps:

1. Integration of the model given in Equations 4.1 and 4.2 over the prediction horizon which is t_0 through t_f, or in discrete terms, $k = 1, 2, ..., P$ where P is the prediction horizon. The initial conditions for the integration are given in Equation 4.19. The values of the state and output variables determined at each step of the integration must be saved for the next step.

2. Solution of Equation 4.18 while integrating Equation 4.17 backwards from t_f through t_0 or, discretely, for $k = P, P-1, ..., 1$. The initial condition for the integration of Equation 4.10 is given in Equation 4.20. The values of λ_1 and λ_2 must be saved at each step of the integration for use in the next step.

3. Calculation of the gradient of the objective function using Equation 4.15 over the range of $k = 1, 2, ..., M$ where M is the control horizon.

Two algorithms most frequently used to solve the optimal control problem are successive quadratic programming (SQP) and generalized reduced gradient (GRG). Both methods can handle the nonlinear objective functions and the nonlinear constraints present in the optimal control

problem [3]. In this work, the GRG method is used to solve the optimal control problem because the SQP method can sometimes lead to infeasible intermediate solutions [38].

4.1.2 Parameter Identification

Nonlinear optimal control can be used to determine new values for the model parameters if feedback cultivation detects errors due to process-model mismatch or unmeasured disturbances. If feedback cultivation detects an inaccurate model parameter, the particular incorrect parameter should be updated. If feedback cultivation detects unmeasured process disturbances, bias terms for the model-predicted outputs should be calculated or corrected. Chapter 3 explains feedback cultivation and how the bias terms fit into the model equations.

The nonlinear optimal control problem for parameter re-identification follows the same principles as the nonlinear optimal control problem for control move determination. The differences appear in the nature of the objective function and the optimization search variables. The nonlinear optimal control problem for parameter re-identification is stated as follows:

$$\begin{aligned} &\text{Min} && \Phi_I \\ &p_j \text{ or } bias && \\ &\text{s.t.} && \dot{x} = f(x,u,p,d) \\ &&& y_p = g(x,u,p,d) + bias \end{aligned} \tag{4.21}$$

where p_j = the inaccurate model parameter detected by feedback cultivation

and $bias$ = the bias terms used to modify the output variables of the model to account for unmeasured disturbances. The constraints of the optimal control problem are the same state and output equations used in Equation 4.3.

The objective function for the parameter re-identification optimal control problem appears as follows:

$$\Phi_I = \int_{t_0}^{t_i} (y - y_p)^2 dt \tag{4.22}$$

where y = the measured value of the process output and y_p = the model predicted value of the process output. The integration in 4.22 proceeds from the current time of the process operation, t_0, backwards to time t_i in the past. To implement the parameter re-identification, measured values of the process output must be saved from time t_i through t_0.

The use of nonlinear optimal control to solve the parameter re-identification problem was suggested by Jang [36]. The difference between the method presented here and the method used by Jang involves the search parameters used to minimize the objective function given in Equation 4.22. Jang's method lumps all the model parameters and the initial conditions of the model state variables together as the search variables of one optimal control problem. This approach combines state estimation and parameter re-identification into one step. The drawback of this approach is that no distinction is made between problems caused by unmeasured disturbances and problems caused by process-model mismatch. These two problems affect the controller very differently and should be treated as separate problems. In

this work, feedback cultivation determines what problems, if any, appear to be affecting the process or controller. Once the problem has been isolated to one specific model parameter or the output bias terms, the optimal control problem in Equation 4.21 can be solved for the specific parameter or bias terms. State estimation is completely separated from parameter re-identification and is discussed in the next section.

Solving Equation 4.21 proceeds in the same fashion as the solution of the optimal control problem in Equation 4.3 for the future control moves. The problem must first be discretized for computer implementation. Then the GRG method can be used to solve Equation 4.21. Jang [36] describes the method for determining the derivatives of Φ_I with respect to the search variables (p_j or *bias*) necessary to use the GRG method.

4.1.3 State Estimation

The nonlinear model predictive control algorithm formulated as the optimal control problem in Equation 4.3 requires initial values for the state variables, x_0, as shown in Equation 4.19. Ideally, the initial values of the state variables, x_0, can be obtained from process measurements. For example, if the temperature of the liquid in a stirred-tank reactor is one of the state variables in the model of the reactor, the latest measurement of the temperature in the reactor would be used as the initial value of the reactor temperature state variable in the nonlinear optimal control algorithm. Sometimes a state variable cannot be measured because either no sensor is available to measure

the variable or the variable is impossible to measure physically. For example, the concentration of a component in the stirred-tank reactor may be a state variable in the model, but no measurement of concentration is available – a common problem in the chemical process industry. In this case, an estimate of the initial value of the state variable must be determined.

Several schemes exist for estimating the unknown x_0. A frequently used scheme is the technique of Kalman Filtering [41]. Another scheme, suggested by Jang [36] and described in the previous section, uses a variation of the nonlinear optimal control algorithm to find the optimum values of the unknown x_0. Jang [36] compared the nonlinear optimal control variation to the Extended Kalman Filter method, which attempts to account for model nonlinearities, and found the nonlinear optimal control variation to be superior.

An even simpler approach to estimating the unknown x_0 is to assume the process is in a psuedo-steady state at the current time. This means $\dot{x} = 0$, and the equations in 4.1 and 4.2 can be solved for the unknown x_0. This method is valid only if enough state variables are measured to give enough information about the process to solve the equations for the remaining unknown state variables.

In this work, at the start of a simulation, the initial conditions for the measured model state variables equal the available measurements. The initial conditions for unmeasured state variables are determined by solving Equations 4.1 and 4.2 with $\dot{x} = 0$ for the unknown initial conditions.

During the simulation, if feedback cultivation detects no problems with the process or controller, the initial conditions are updated each time a new control action is called for. Before a new control action is determined, the model is integrated from the previous initial conditions to the present time using the control move determined at the last control time step. The final values of the state variables after the integration equal the initial conditions for the model state variables at the current time.

If feedback cultivation detects process or controller problems, the initial condition values are updated differently. In this work three possible methods for updating the initial conditions are examined:

1. Set the initial condition values equal to the latest values of the variable measurements. If measurements are not available, assume a steady state and solve the state equations for the unknown state initial conditions.

2. Apply a technique similar to that used by Jang [36] to search for the initial condition values along with the model parameter or bias terms determined to be inaccurate by feedback cultivation.

3. Use a feedback cultivation identification horizon large enough to reduce the effect of inaccurate initial condition values on the model parameter search optimization.

Of the three possible methods for determining the initial conditions, the first one is the most desirable if good measurements are available. Any noise is removed from the signals by filtering. The second method adds search

variables to the parameter re-identification optimization. More search variables leads to slower optimization convergence. The third method may require too much data to be useful. Once feedback cultivation detects a process or controller problem, enough data must be accumulated during which the problem exists. This may lead to long periods of operation where upsets or setpoint deviations are present.

4.2 Control System Methodology

The complete control system includes the nonlinear optimal control algorithm in Equation 4.3, adaptive filtering, feedback cultivation, and the parameter re-identification algorithm given in Equation 4.21. Figure 4.1 shows a block diagram of the complete control system implemented in this work. A flowchart of the control system methodology given in Figure 4.2 shows the decision path for the methodology. Each step indicated in the flowchart is described in the following list:

1. Data acquisition – Obtaining the latest disturbance and output measurements from the process. If more than one measurement occurs in between control move calculations, an average of the measurements can be used.

2. Filtering – Adaptively filtering the disturbance and output measurements involves the following steps:

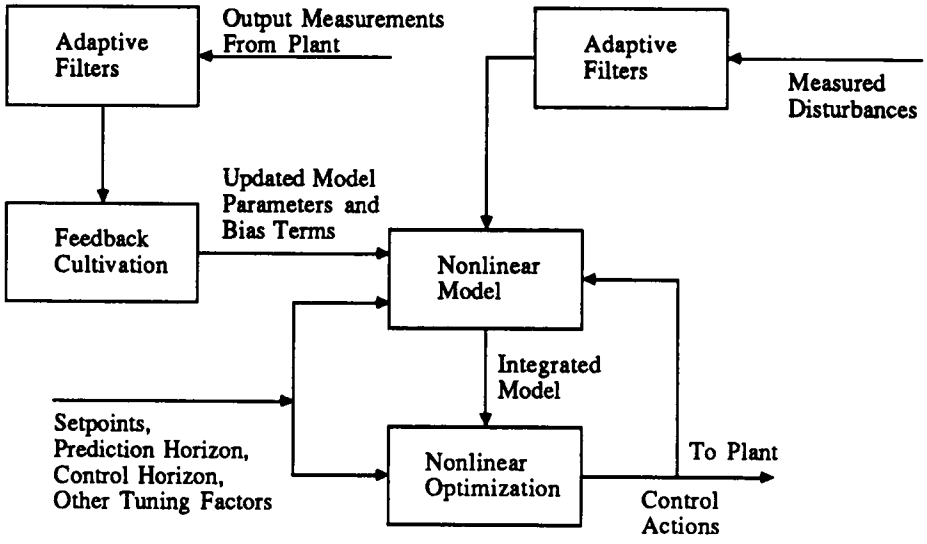


Figure 4.1: Complete nonlinear optimal control system

- Saving each of the latest measurements into an n-dimensional vector shifting the oldest measurement out of the vector.
- Calculation of the average of the n values in each vector to use as the desired filter output in the weight adjustment step.
- Calculation of the filtered measurements using Equation 3.19.
- Optimization of the weights using Equation 3.16 by minimizing the error given in Equation 3.15.
- Passing the filtered values of the disturbance measurements to the nonlinear optimal control algorithm and the filtered values of the output measurements to the feedback cultivation algorithm.

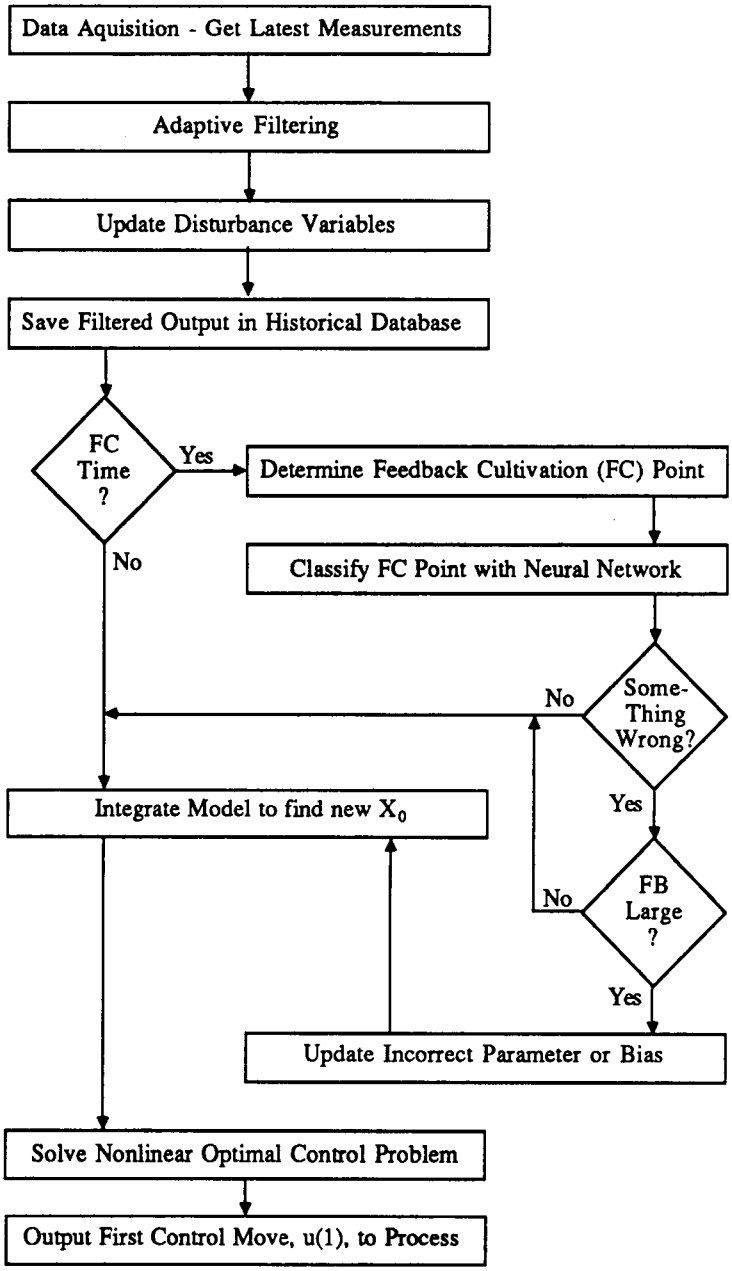


Figure 4.2: Control system methodology flowchart

3. Checking for feedback cultivation time – If the specified number of control moves have been determined since the last execution of the feedback cultivation algorithm, the following steps are taken:

   - Cultivation of the values in the historical database accumulated since the last two feedback cultivation executions (i.e. half previously cultivated data and half new data because the cultivation windows overlap as shown in Figure 3.6). One feedback cultivation point results.
   - Inputing the feedback cultivation point to the neural network for classification, possibly along with several previously determined cultivation points.
   - Interpretation of the neural network results. If the points fall into a parameter error class or the null class, check the feedback values (Equation 3.10) for significance. If the feedback is significant, set the appropriate flag indicating which model parameter or bias terms need correction.

4. Checking flags from feedback cultivation – If something is wrong and it has been detected for a specified number of successive feedback cultivation applications, update the appropriate parameter or bias terms using Equation 4.21 and one of the three state estimation methods to determine the state variable initial conditions.

5. Control move determination – Determination of the next control move

involves the following steps:

- Updating the appropriate model variables with the latest filtered disturbance measurements.
- Updating the model parameters and bias terms if any new values determined are in the parameter re-identification phase
- Integration of the model from the last control time to the current time to determine the initial values of the model state variables, x_0. If the bias terms or any model parameters have been re-identified, the model integration starts from the time of the last accurate initial conditions at the beginning of the feedback cultivation horizon.
- Saving the necessary variables for feedback cultivation and parameter re-identification in the historical database.
- Shifting the previously $M-1$ determined control moves so that $u_{2,old} = u_{1,new}$, etc. and use these values as the initial guesses for the next control moves.
- Solving the optimal control problem given in Equation 4.3 for the next control moves $u = (u_1, u_2, ...u_M)$.

6. Outputing the first control moves for each manipulated variable, i.e. u_1, to the process.

The next two chapters demonstrate the use of this control system methodology. Chapter 5 applies the methodology to a first-order process with

a first-order model. The application is simple enough to give a feel for how the algorithm works. Chapter 6 applies the methodology to a stirred-tank reactor that involves a nonlinear reaction, process measurement noise, and unmeasured process disturbances.

CHAPTER 5

First-Order Process and Model Example

The first-order process with the nonlinear optimal control algorithm using a first-order model described in this chapter provides a simple demonstration of the nonlinear optimal control algorithm operating with feedback cultivation. Section 5.1 gives a description of the system and the operating conditions of the process and controller. Section 5.2 presents the results of simulating the first-order process with the nonlinear optimal control algorithm using feedback cultivation.

5.1 System Description

5.1.1 The Process

The process used in this demonstration involves a first-order equation in the following form:

$$\frac{dx}{dt} = -\frac{1}{\tau}x + \frac{K}{\tau}u_1 \tag{5.1}$$

where x = the state variable of the process, u_1 = the control action, K = the process gain, and τ = the process time constant. The measured output of the process includes a term that simulates unmeasured disturbances of the

process:

$$y = x + d_u \tag{5.2}$$

where y = the measured process output and d_u = the unmeasured disturbance.

The following parameter values define normal operation of the first-order process:

$$\begin{aligned} \tau &= 1146.6 \\ K &= 342.6 \\ d_u &= 0.0. \end{aligned} \tag{5.3}$$

The state variable, x, the output variable, y, and the control variable, u_1, start each simulation with the following values:

$$\begin{aligned} x &= 88.0 \\ y &= 88.0 \\ u_1 &= 0.2569. \end{aligned} \tag{5.4}$$

A fourth-order Runge-Kutta method integrates the process given by Equations 5.1 and 5.2 using an integration time step of 20.0 seconds.

5.1.2 The Controller

The nonlinear optimal control algorithm determines the values for the control actions given by the vector $u = (u_1, u_2, ..., u_M)$ where M is the control horizon. A first-order equation similar to Equation 5.1 models the process in the control algorithm:

$$\frac{dx_m}{dt} = -\frac{1}{\tau_m}x_m + \frac{K_m}{\tau_m}u \tag{5.5}$$

where x_m = the model state variable, K_m = the model gain, and τ_m = the model time constant. The model output includes a bias term determined with feedback cultivation to account for unmeasured disturbances:

$$y_m = x_m + bias. \tag{5.6}$$

The optimal control problem solved by the nonlinear optimal control algorithm uses Equations 5.5 and 5.6 as part of the constraint set in the optimization:

$$\begin{array}{lllll} \text{Min} & \Phi & & & \\ u & & & & \\ \text{s. t.} & \dot{x_m} & = & -\frac{1}{\tau_m}x_m + \frac{K_m}{\tau_m}u & \\ & y_m & = & x_m + bias & \\ & 0 & \leq & u \leq 1 & \\ & u & = & (u_1, u_2, ..., u_M) & \end{array} \tag{5.7}$$

where Φ = the objective function and u = a vector of control moves. The control moves are the search variables of the optimization. Recall that the process implements only the first control move in the vector, u_1, as indicated in Equation 5.1.

The objective function for the nonlinear optimal control algorithm contains two terms:

$$\Phi = \sum_{i=1}^{P} \beta(y_{set} - y_{m_i})^2 + \gamma \left| u_1 - u_{1_{old}} \right| \tag{5.8}$$

where β = the weighting factor for the output term and γ = the weighting factor for the control move change in the process. The first term sums the

differences between the output setpoint, y_{set}, and the model predicted output, y_m. The summation covers the prediction horizon, P. The second term is a difference term between (1) the control move to be implemented in the process next, u_1, and (2) the most recently implemented control move, $u_{1_{old}}$. The absolute value of this term is used instead of the squared value. The absolute value of the difference gives a better indication of the magnitude of the control move change because the difference is always less than one.

Initially, the model parameters, the model state variable, and the model output variable are set to the same values as the process parameters and variables:

$$\begin{aligned} \tau_m &= 1146.6 \\ K_m &= 342.6 \\ x_m &= 88.0 \\ bias &= 0.0 \\ y_m &= 88.0. \end{aligned} \tag{5.9}$$

The controller tuning parameters for the nonlinear optimal control algorithm have the following values:

$$\begin{aligned} P &= 50 \\ M &= 5 \\ \beta &= 1.0 \\ \gamma &= 50.0 \\ \Delta t_c &= 20.0 \text{ seconds} \end{aligned} \tag{5.10}$$

where P = the prediction horizon, M = the control horizon, β = the output

weighting factor, γ = the move suppression factor, and Δt_c = the control action time step. In this example, the control time step is the same as the measurement time step.

The prediction horizon, P, is 50 control time steps which indicates that the process output is predicted 1000.0 seconds (50 times 20.0) into the future. This is about the length of time required for the process to reach steady state after a disturbance enters the process.

Setting the control horizon, M, is a matter of trial and error in this example. Values of M ranging from 2 to 20 were tried before selecting 5 as the value for M. Considering speed of controller operation and oscillation of the process response, a control horizon of 5 gives the best results.

Section 5.2.1 presents results for varying values of the control horizon and the prediction horizon. These results demonstrate that the values $M = 5$ and $P = 50$ result in the best controller tuning of all the values of M and P tried.

The output weighting factor, β, and the move suppression factor, γ, are selected so that the two terms in the objective function in Equation 5.8 have similar magnitudes.

5.1.3 Feedback Cultivation

The model used in the nonlinear optimal control algorithm contains two parameters that relate to the same parameters in the process: K_m, the model gain, and τ_m, the model time constant. Feedback cultivation can be used to

detect differences between the model and process parameter values. Once a difference is detected, values of the appropriate parameters in the model can be corrected to account for the difference.

Implementation of feedback cultivation involves three steps. The first step generates the reduced feedback cultivation dataset or data point. The second step classifies the reduced feedback cultivation data using a trained neural network. The third step interprets the results from the neural network classification and corrects any inaccurate parameters.

Before the first step of feedback cultivation can be implemented, the summary and statistic variables that make up the reduced feedback cultivation dataset or data point must be selected. For the first-order example, statistic and summary variables must be selected that distinguish mismatch between the process and model gain terms and the process and model time constant terms. Also, the statistic and summary variables need to distinguish between parameter mismatch and unmeasured disturbances. Section 5.2.2 presents the results for selecting statistic and summary variables for the first-order example. The selection of the feedback cultivation variables is somewhat arbitrary.

Before the neural network can classify unknown feedback cultivation data points, it must be trained with datasets representative of differences between the process and model gain, and differences between the process and model time constant. The neural network is trained to recognize five classes of process and controller operation: one representing normal operation, one

representing an increase in the process gain, one representing a decrease in the process gain, one representing an increase in the process time constant, and one representing a decrease in the process time constant. Five datasets representing each of the five classes are generated by simulating the first-order process with the nonlinear optimal controller but without feedback cultivation. For each simulation, the modes of the process are excited by varying the setpoint of the process output variable. Figure 5.1 shows the value of the setpoint for about a 4-hour simulation. Figure 5.2 shows the values of the process and model parameters for each of the five simulations needed to generate the historical databases for each of the five training datasets.

The model parameters and the bias term for the first-order example are updated using the modified nonlinear optimal control algorithm described in Chapter 4. If a mismatch between process and model gain terms is detected by the neural network, the modified nonlinear optimal control algorithm searches for a new value of the model gain. Likewise, if a mismatch between process and model time constants is detected, the modified nonlinear optimal control algorithm searches for a new value of the model time constant.

The neural network classifies unmeasured disturbances in the null class indicating that the data does not represent anything the network has been trained to recognize. The bias term in Equation 5.6 accounts for unmeasured process disturbances. When the neural network detects unmeasured

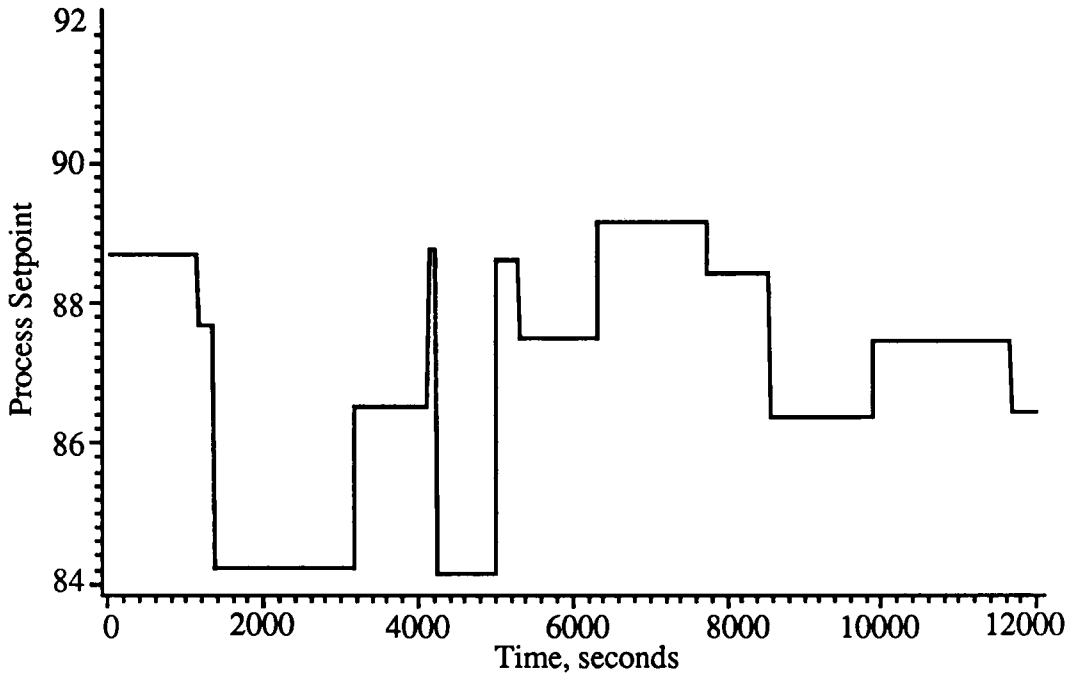


Figure 5.1: Setpoint disturbance of first-order process for neural network training sets.

Set #	Set Name	Model Time Constant	Process Time Constant	Model Gain	Process Gain
1	τ–	**1032.5**	1146.5	342.6	342.6
2	τ+	**1260.5**	1146.5	342.6	342.6
3	K-	1146.5	1146.5	**325.6**	342.6
4	K+	1146.5	1146.5	**359.6**	342.6
5	Normal	1146.5	1146.5	342.6	342.6

Note: Changed parameters appear as boldface text.

Figure 5.2: Model and process parameters for the five training datasets.

disturbances, the modified nonlinear optimal control algorithm searches for new values of the bias term.

Section 5.2.3 presents the results of using the modified nonlinear optimal control algorithm to update the model parameters and the bias term. Also presented are results for each of the three methods of updating the initial value of the state variable in the model. The first-order example helps give a clear explanation of the effect of several state estimation techniques on the operation of the nonlinear optimal control algorithm.

As discussed in Chapter 4, the choice for the value of the initial value of the state variable, x_0, determines how accurately the parameter identification algorithm identifies a new model parameter. Three methods are explored in this example: using the measured value of the state variable, searching for a new value for x_0 along with the inaccurate model parameter, and using a large identification horizon to minimize the effect of an inaccurate initial condition value. Section 5.2.3 presents results for each of these three initial condition determination methods.

5.2 First-Order Example Results

The results for the first-order example are presented in the next three sections. Section 5.2.1 presents results demonstrating the effect of varying the control horizon and the prediction horizon in the nonlinear optimal control algorithm. Section 5.2.2 gives results for the development of the feedback

cultivation algorithm for the first-order example. Finally, Section 5.2.3 presents simulations of the first-order process using the nonlinear optimal control algorithm with feedback cultivation.

5.2.1 Controller Tuning Results

The results of simulating the first-order process using the nonlinear optimal control algorithm with varying control horizons and prediction horizons are given in this section. The model in the nonlinear optimal control algorithm uses the same gain and time constant as the process. Also, the simulations contain no unmeasured disturbances. Because ideal model parameters are used and no unmeasured disturbances are present, the feedback cultivation algorithm is not implemented in these simulations.

The duration of the simulations is about 3 hours. The setpoint of the process output varies between 84.0 and 90.0 during the simulations as indicated in Figure 5.3. The results report the variation of the process output and the model predicted output, the variation of the control variable, and the variation of the deviation of the process output from the desired setpoint for each simulation.

Figures 5.4 and 5.5 show the results of simulating the first-order process with the setpoint disturbance shown in Figure 5.3 with a control horizon of $M = 5$ and a prediction horizon of $P = 50$. These parameters represent a satisfactory compromise between the desire to limit excessive oscillation of the process output and the quickness of response of the process

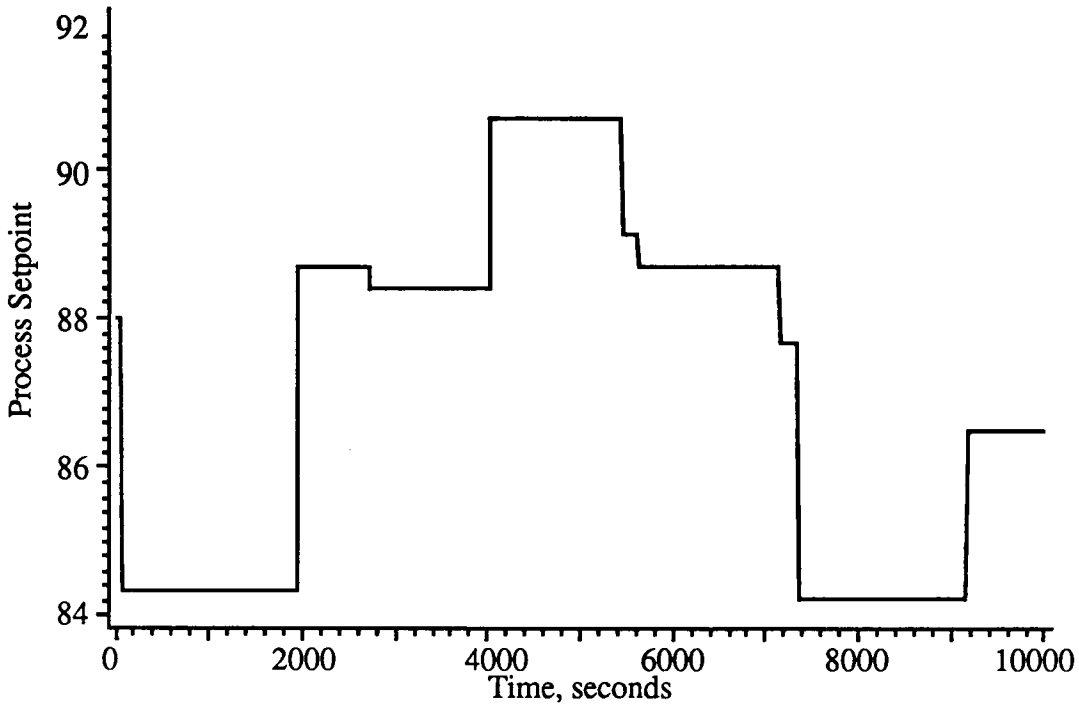


Figure 5.3: Setpoint disturbance of first-order process for controller tuning simulations.

output to changes in the setpoint. All simulations of the first-order process presented in Sections 5.2.2 and 5.2.3 use a control horizon of $M = 5$ and a prediction horizon of $P = 50$.

Figures 5.6 and 5.7 show the results of simulating the first-order system with a control horizon of $M = 2$. Figures 5.8 and 5.9 show the results of simulating the first-order system with a control horizon of $M = 10$. In both cases, the prediction horizon has a value of $P = 50$. The figures indicate that changes in the control horizon affects the speed of the controller's response to changes in the process output setpoint.

Figure 5.6 shows that too low a value for the control horizon results in

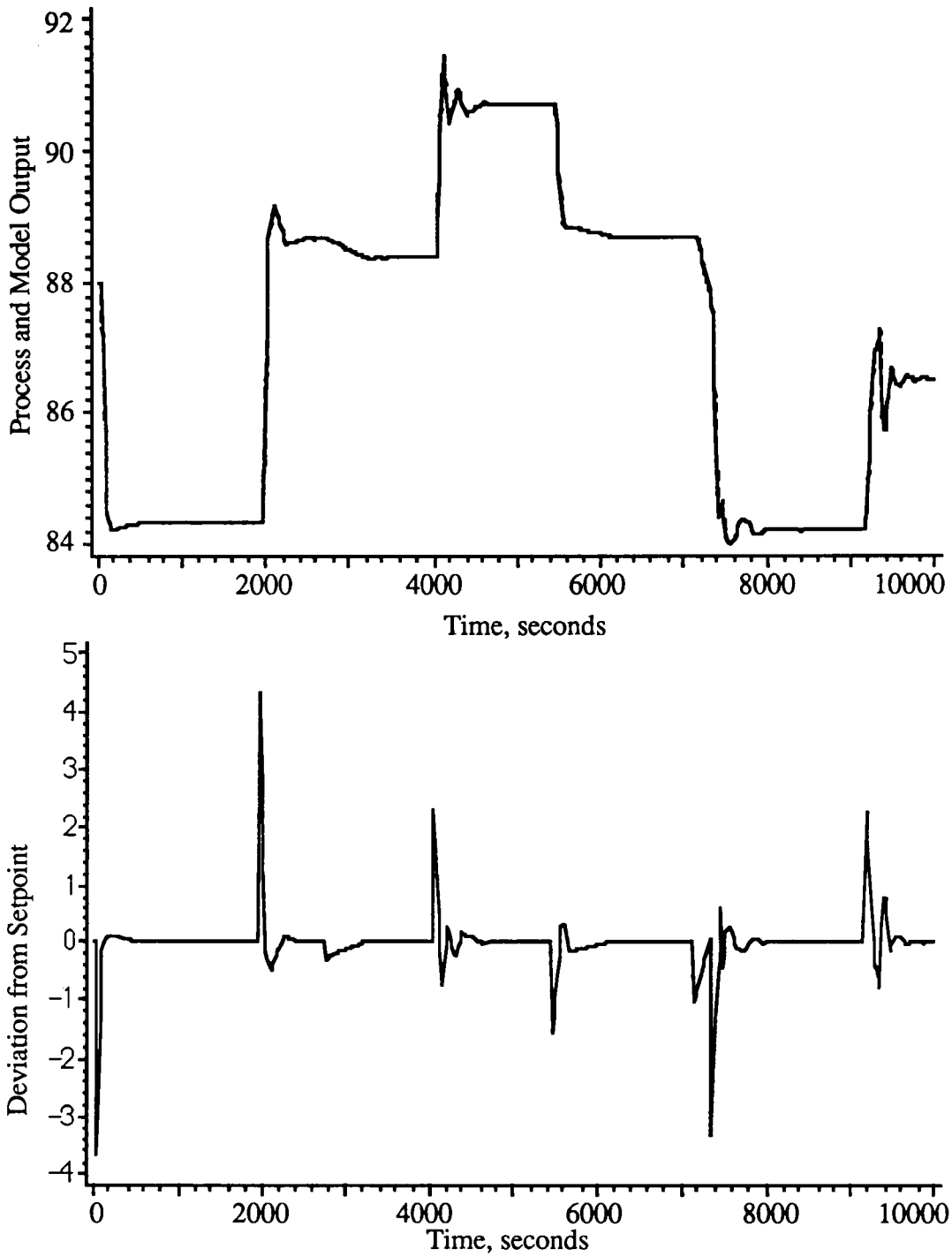


Figure 5.4: Outputs and deviation from setpoint for simulation with $M = 5$ and $P = 50$.

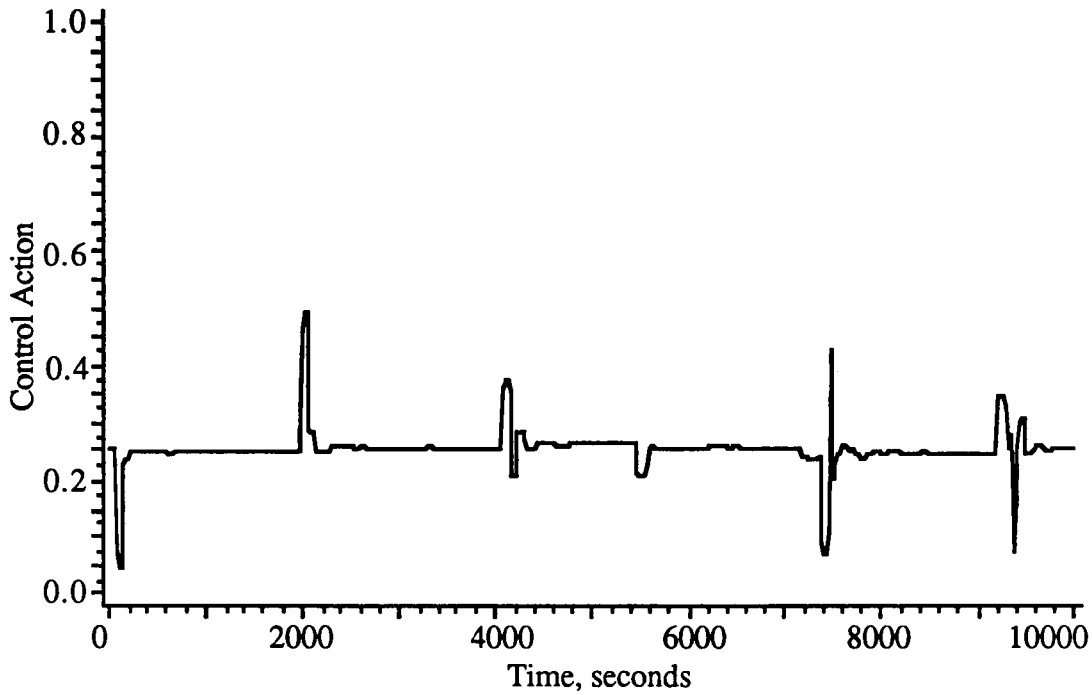


Figure 5.5: Control Action for simulation with $M = 5$ and $P = 50$.

a very sluggish response of the controller to the setpoint changes. Figure 5.8 shows that too high a value for the control horizon results in a faster but more oscillatory response of the controller to the setpoint changes.

Figures 5.10 and 5.11 show the results of simulating the first-order system with a prediction horizon of $P = 10$. Figures 5.12 and 5.13 show the results of simulating the first-order system with a prediction horizon of $P = 80$. In both cases, the control horizon has a value of $M = 5$. The figures indicate that changes in the prediction horizon affects the degree of oscillation of the controller's response to changes the setpoint.

Figure 5.10 clearly shows that too low a value for the prediction horizon increases the amount of oscillation in the controller response. The

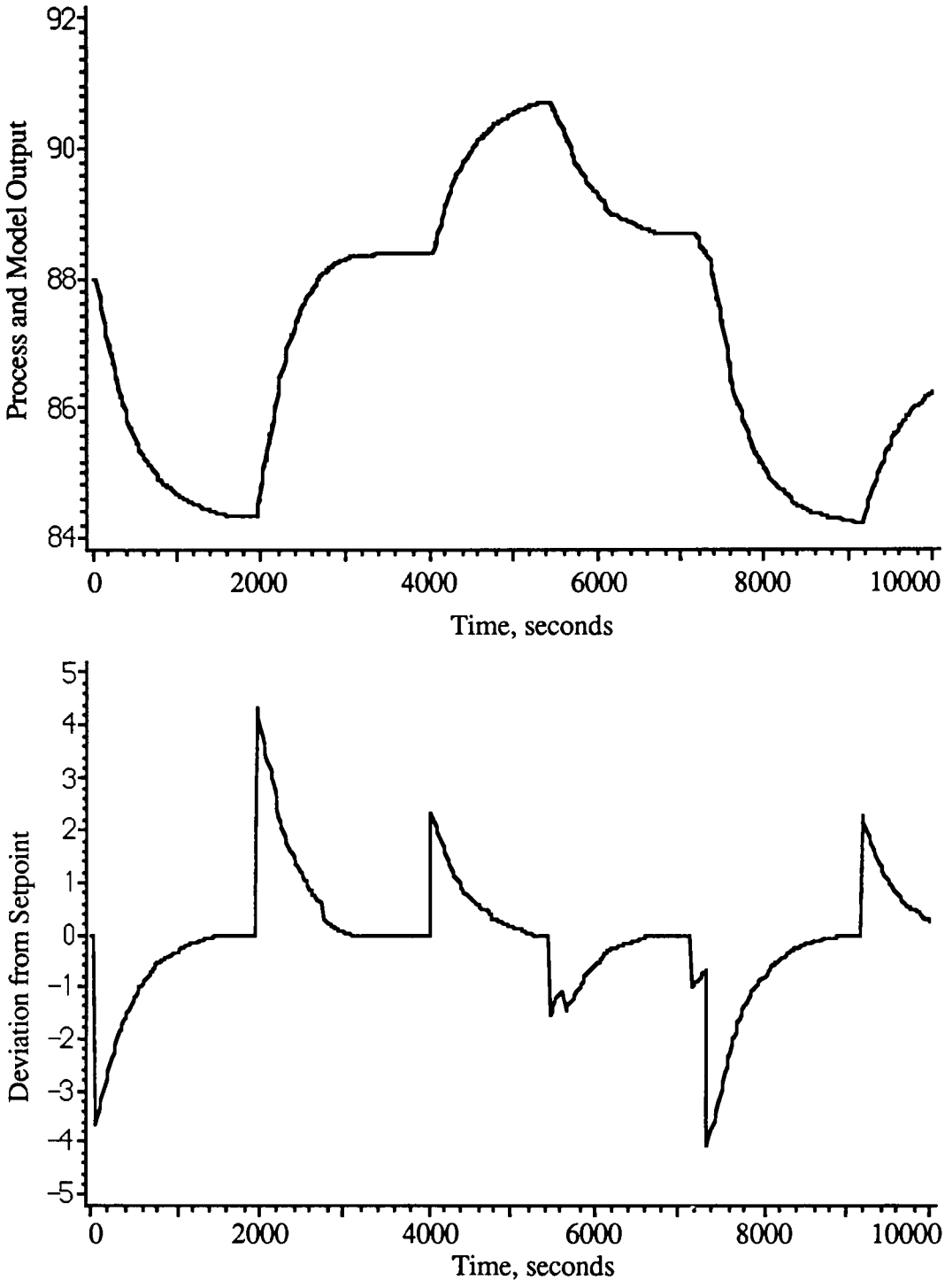


Figure 5.6: Outputs and deviation from setpoint for simulation with $M = 2$ and $P = 50$.

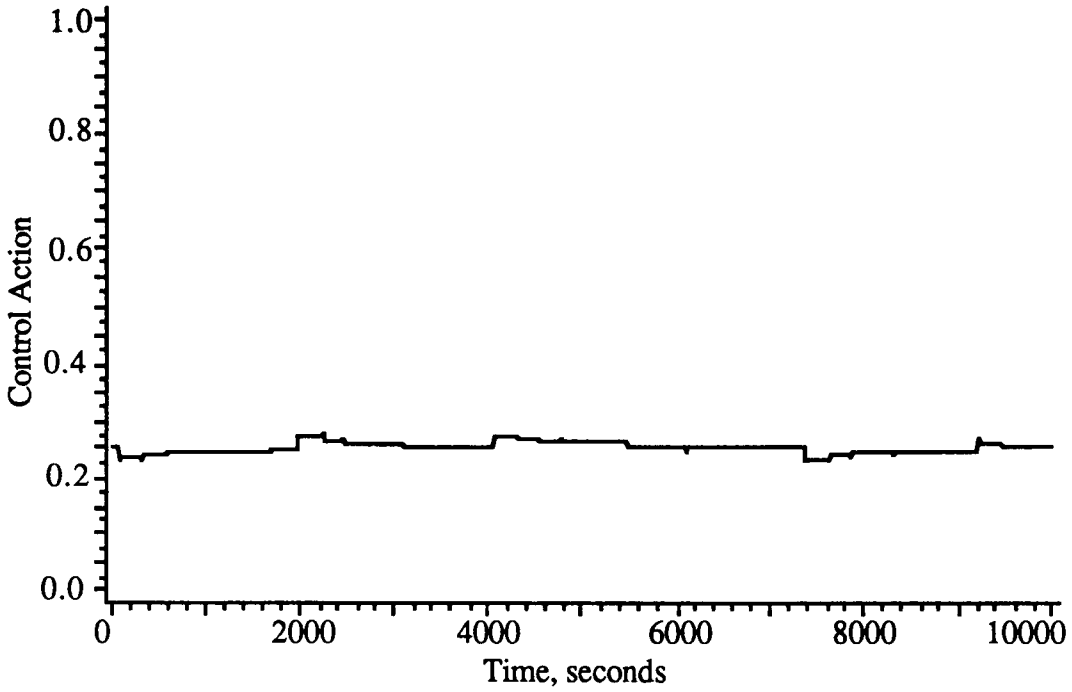


Figure 5.7: Control action for simulation with $M = 2$ and $P = 50$.

first-order process does not have time to settle to a new setpoint between changes in the setpoint value due to the excessive oscillations of the control variable.

Figure 5.12 shows that the higher the prediction horizon, the lower the amount of oscillation of the control variable due to changes in the setpoint. In fact, for some of the setpoint changes, the controller responds better than the controller with a prediction horizon of $P = 50$ as indicated by the plot of the deviation between the process output and the setpoint in Figure 5.12. The disadvantage of using a value of $P = 80$ instead of $P = 50$ relates to the length of computation time required for the nonlinear optimal control algorithm to determine a new set of control moves. The higher the value of

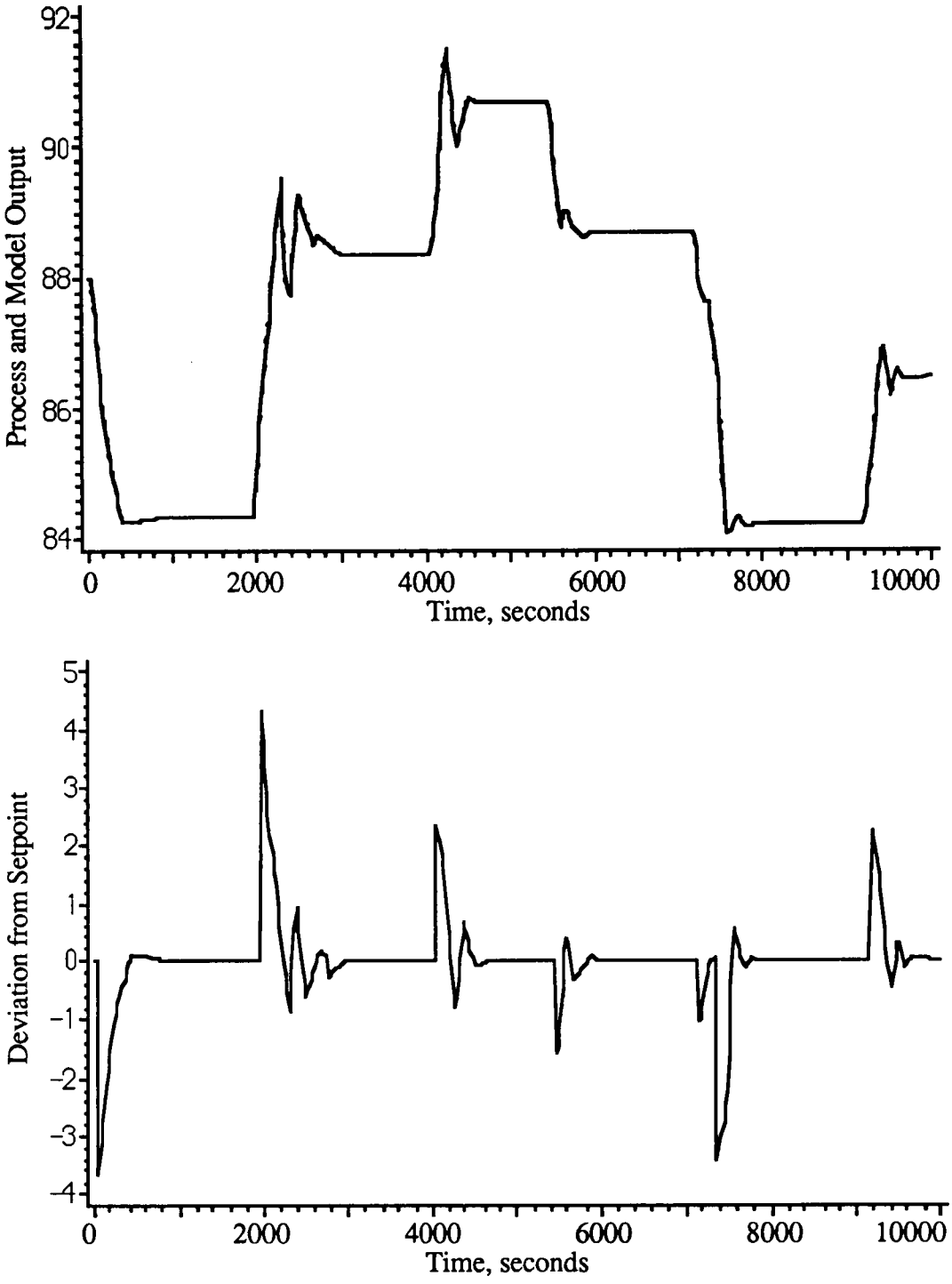


Figure 5.8: Outputs and setpoint deviation for simulation with $M = 10$, $P = 50$.

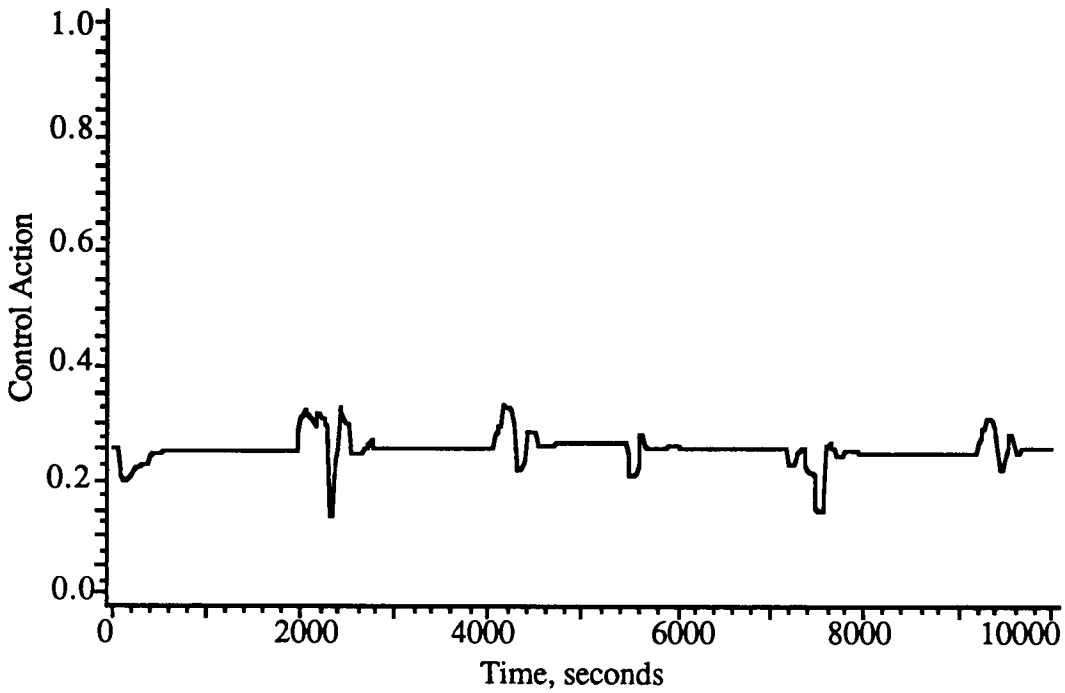


Figure 5.9: Control action for simulation with $M = 10$, $P = 50$.

the prediction horizon, the slower the nonlinear optimal control algorithm functions. A value of $P = 50$ represents a compromise between limiting oscillations and speed of operation of the nonlinear optimal control algorithm.

The nonlinear optimal control algorithm contains two other parameters that affect the response of the controller to process changes: the output weighting factor, β, and the move suppression factor, γ. These factors fit into the optimal control algorithm through the objective function as indicated in Equation 5.8. In the simulations of the first-order process, the output weighting factor is set to a constant value of $\beta = 1.0$. The move suppression factor is set to a value that scales the move suppression term to the same magnitude as the output term in the objective function in

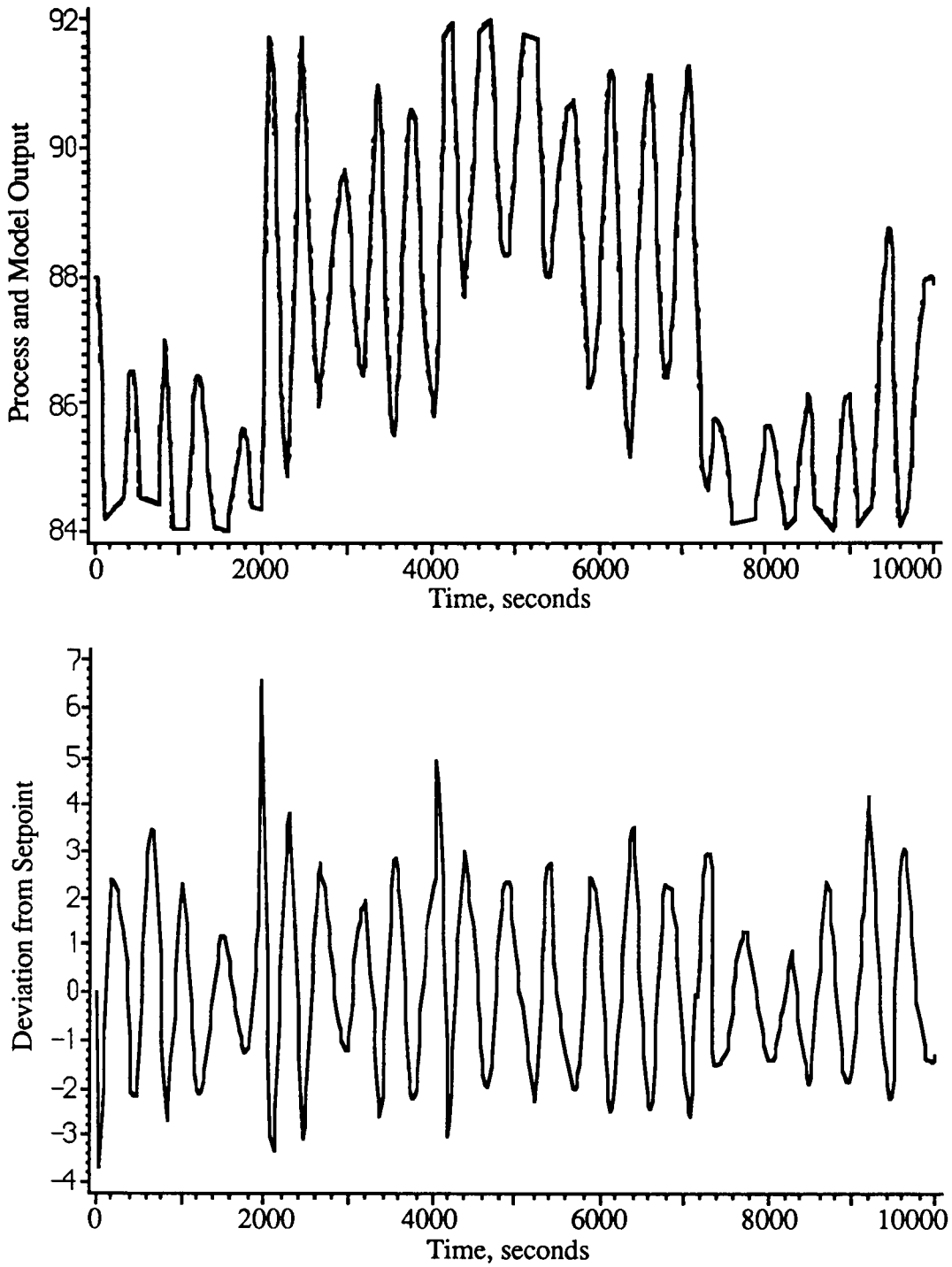


Figure 5.10: Outputs and setpoint deviation for simulation with $M = 5$, $P = 10$.

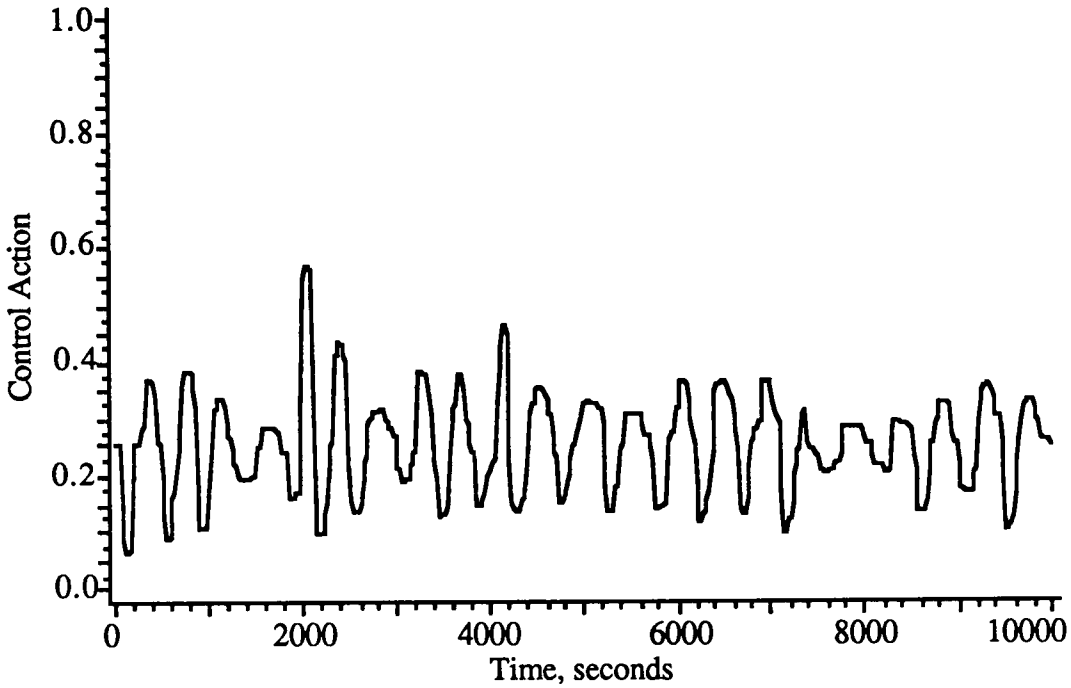


Figure 5.11: Control action for simulation with $M = 5$, $P = 10$.

Equation 5.8. In all the simulations of the first-order process, the move suppression factor has a constant value of $\gamma = 50.0$.

5.2.2 Feedback Cultivation Results

Before the feedback cultivation algorithm works, the neural network that classifies feedback cultivation data must be trained. As previously mentioned, the neural network learns to distinguish between model time constant error, model gain error, and normal operation using five sets of feedback cultivation data as training data. Once the neural network training finishes, the network operates as a pattern classifier.

The feedback cultivation variables used in this example consisted of 7

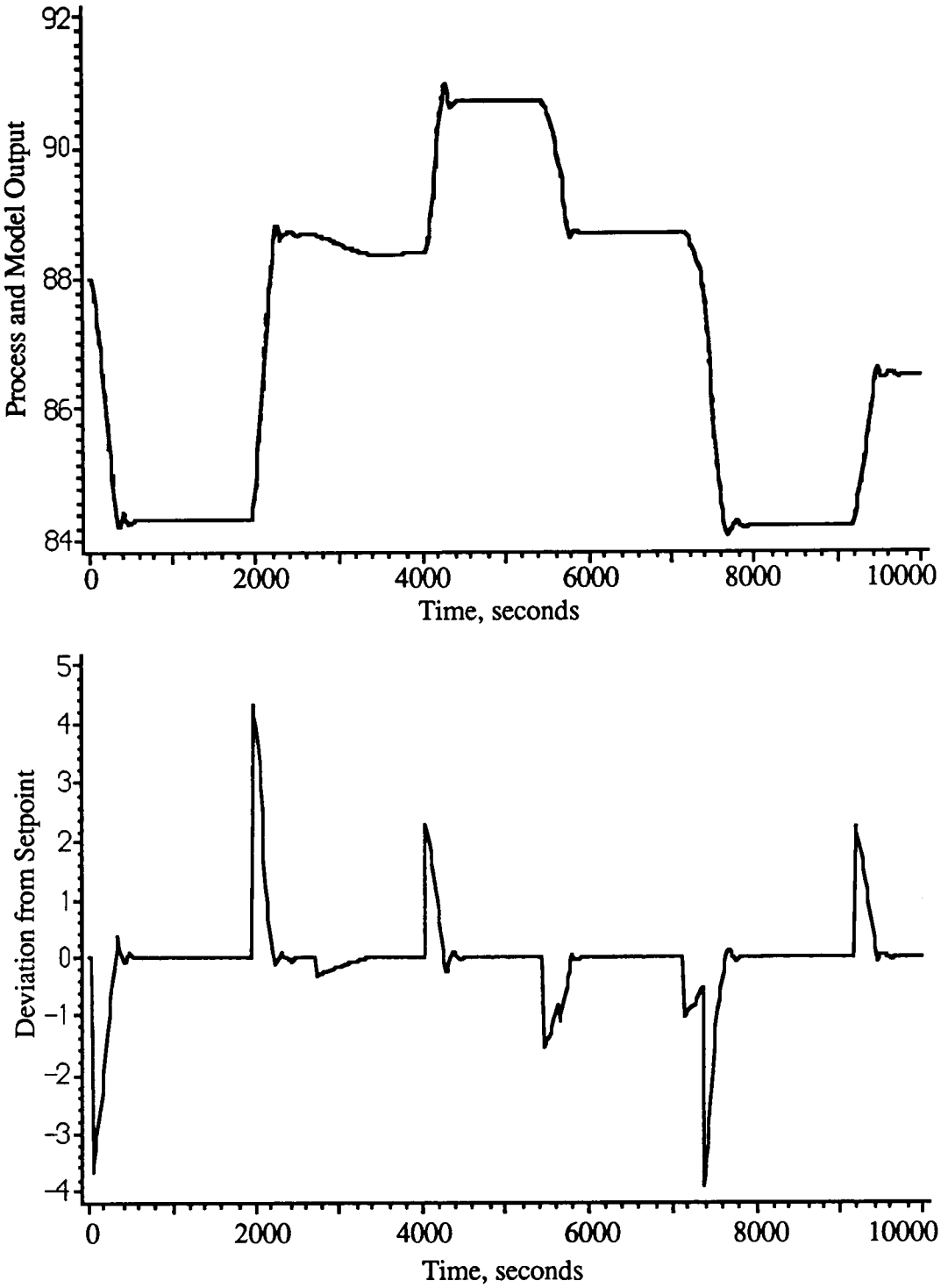


Figure 5.12: Outputs and setpoint deviation for simulation with $M = 5$, $P = 80$.

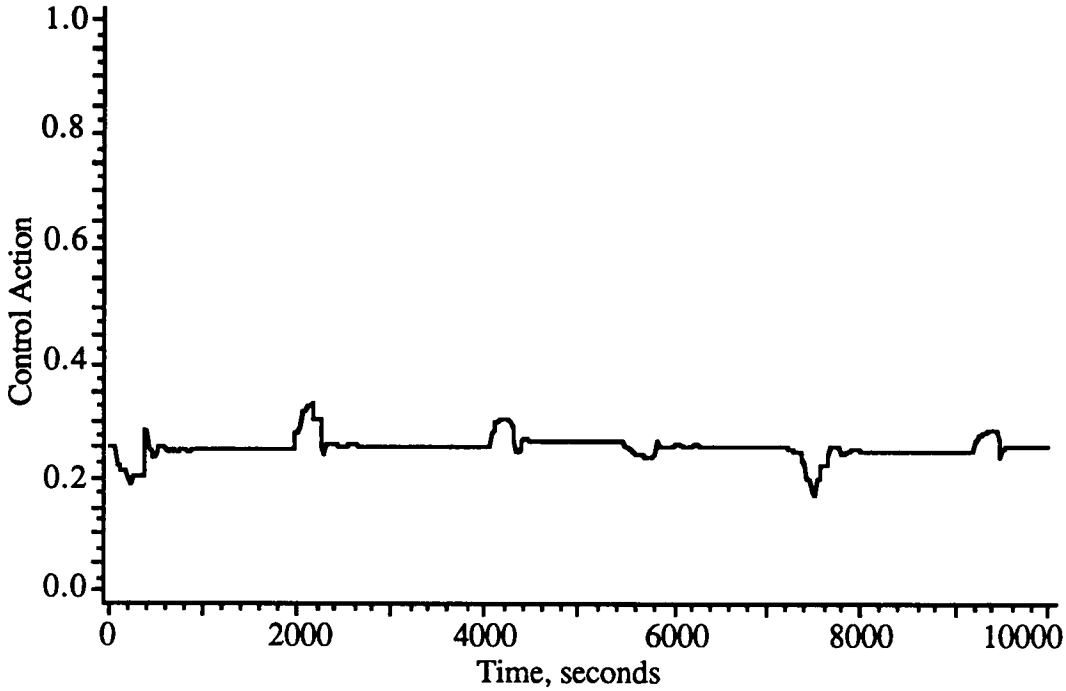


Figure 5.13: Control action for simulation with $M = 5$, $P = 80$.

statistic and summary variables. Figure 5.14 defines the feedback cultivation variables. Ratios of the summary and statistic variables are used to invoke natural scaling of the data. The fraction of positive and negative feedback points in the datasets always lies between 0 and 1. Most of the ratios vary between -1 and 1 naturally, some scaling keeps all values between -1 and 1. All of the data in the training and test datasets is scaled between -1 and 1 before input to the neural network.

The neural network used in this example consists of 7 input nodes, one for each feedback cultivation variable, and 5 output nodes, one for each class of feedback cultivation data the network is trained to recognize. The two hidden layers of the network contain 15 and 11 nodes respectively. The

$frac\_neg$	Fraction of points with negative values of $FB = y - y_m$
$frac\_pos$	Fraction of points with positive values of $FB = y - y_m$
$\frac{ssfb}{sse}$	Ratio of sum of square FB values and the sum of square errors, $error = y_{set} - y_m$
$\frac{fb\_avg}{fb\_range}$	Ratio of the average value of FB and the range of FB
$\frac{sae}{sse}$	Ratio of the sum of absolute errors and the sum of squared errors
$\frac{skew}{kur}$	Ratio of the degree of skewness of FB and the kurtosis of FB*
$\frac{ssfb}{safb}$	Ratio of the sum of square FB values and the sum of absolute FB values

* Skewness measures the deviation of the median of the data from the mean value of the data. Kurtosis measures the degree of peakedness of the data.

Figure 5.14: Feedback cultivation variables for the first-order example.

learning rate constant, variable α in Equation 3.5, has a value of 0.45 during training. Each of the five training data sets contains 11 feedback cultivation data points.

The results of classifying the training data sets after completion of the training are shown in Figure 5.15. The results show that the backpropagation neural network correctly classifies 100% of the training data.

Before incorporating the feedback cultivation algorithm into the nonlinear optimal control algorithm, classification of a number of test datasets by the neural network determines how well the network performs under different conditions from the training conditions. Figure 5.16 shows the results for nine neural network tests. Results for both smaller and larger magnitude changes in each of the plant parameters than the magnitude change of the training data are represented in the figure. Also shown are results of classifying an unmeasured process disturbance. For an unmeasured disturbance of $d_u = -1.5$, the network correctly classifies the data as the null class. But, for an unmeasured disturbance of $d_u = 3.0$ the network incorrectly classifies the data as a positive difference between the process and model gain terms. More judicious selection of the feedback cultivation statistic and summary variables may fix this problem with the neural network classifying some unmeasured disturbances as model gain errors. This research does not look at any other possible feedback cultivation variables for this example.

To justify the need for data reduction in the form of feedback cultivation, training the neural network to distinguish between model errors

	% of Data Classified in Each Class of Training Data					
Dataset	τ–	τ+	K–	K+	Normal	Null
$\Delta\tau = -57.0$	100	0	0	0	0	0
$\Delta\tau = +57.0$	0	100	0	0	0	0
$\Delta K = -17.0$	0	0	100	0	0	0
$\Delta K = +17.0$	0	0	0	100	0	0
Normal	0	0	0	0	100	0

Figure 5.15: Results of neural network classification of training datasets.

	% of Data Classified in Each Class of Test Data					
Dataset	τ–	τ+	K–	K+	Normal	Null
$\Delta\tau = -114.0$	**82**	18	0	0	0	0
$\Delta\tau = +114.0$	45	**55**	0	0	0	0
$\Delta\tau = -28.0$	**64**	27	0	0	0	9
$\Delta\tau = +28.0$	27	**46**	0	0	0	27
$\Delta K = -8.0$	18	0	**55**	0	0	27
$\Delta K = +8.0$	18	0	0	**73**	0	9
$\Delta K = -34.0$	0	0	**82**	0	0	18
du = -1.5	0	0	0	0	0	**100**
du = 3.0	0	0	0	100	0	**0**

Note: Correct classifications are indicated with bold type.

Figure 5.16: Results of neural network classification of test datasets.

and normal operation with raw data was attempted. After using 27 hours of DEC VAX 6000-440 CPU time, the network correctly classified only 30% of the training data. This compares to 30 - 45 minutes DEC VAX 6000-440 CPU time required to train the neural network with the reduced feedback cultivation datasets. Using feedback cultivation helps the neural network learn much faster. Farell [23] presents similar results comparing the training of the perceptron neural network with raw data and reduced data resulting from hypothesis feedback modeling.

5.2.3 Model Re-identification Results

This section presents results of six different cases of simulating the first-order process with the complete nonlinear optimal control algorithm using feedback cultivation. All six simulations use the setpoint changes shown in Figure 5.17. For each simulation, a process gain change is made near the beginning of the simulation at a time of 1000.0 seconds, a process time constant change is made at a time of 10000.0 seconds, and an unmeasured disturbance occurs at a time of 25000.0 seconds. The process gain is changed from a value of $K = 342.6$ to a value of $K = 325.6$. The process time constant is changed from a value of $\tau = 1146.5$ to a value of $\tau = 1260.5$. The magnitude of the unmeasured disturbance is $d_u = -1.5$. To ensure that the system reaches steady state after each change, the controller uses a model re-identification horizon of 5 or 7 times the feedback cultivation horizon depending on the initial condition determination method. For initial condition re-identification

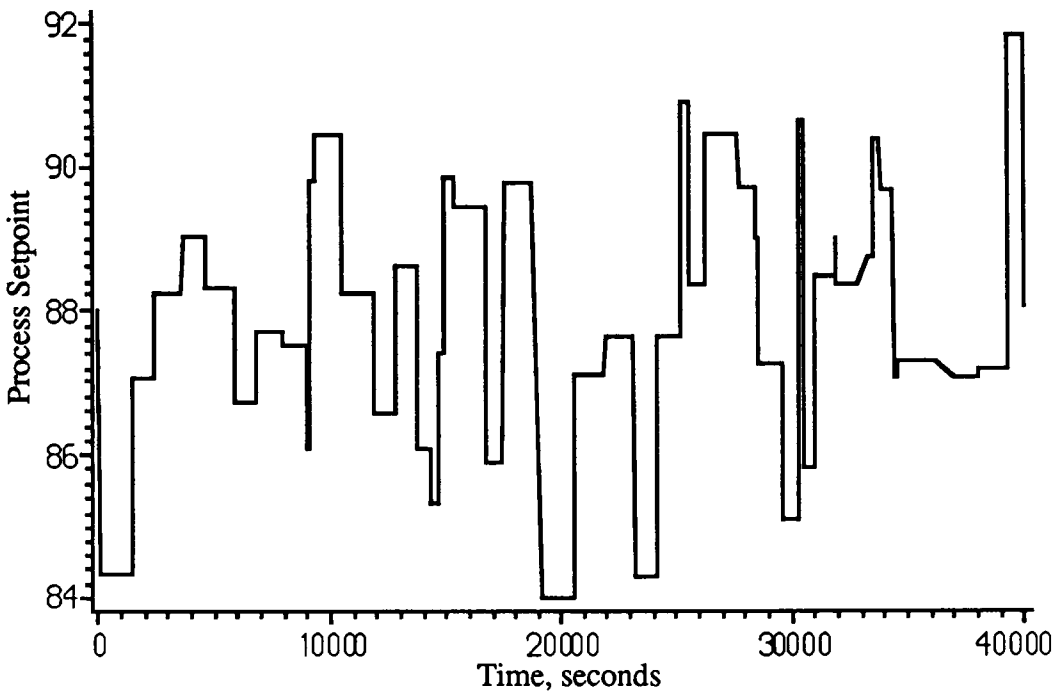


Figure 5.17: Setpoint disturbance used during model re-identification tests.

methods 1 and 2, a model re-identification horizon of 5 times the feedback cultivation horizon is used. For initial condition re-identification method 3, a model re-identification horizon of 7 times the feedback cultivation horizon is used. The simulation time corresponding to the model re-identification horizons is 3000.0 to 3500.0 seconds. In other words, feedback is introduced into the nonlinear optimal control algorithm at a time interval of 3000.0 to 3500.0 seconds.

The results compare the three different methods of re-identifying the model initial condition, x_0, for the first-order system:

1. setting x_0 equal to the latest measurement minus the model bias term:

$$x_0 = y - bias, \tag{5.11}$$

2. searching for x_0 along with the inaccurate model parameters or bias term, and

3. using a large identification horizon so that the effects of an inaccurate x_0 are minimized. In this case, x_0 is the value available to the model based on the value from the beginning of the simulation.

The first method can only be used if measurements of the state variables are available. In this first-order example, measurements of the state variable are available. Many times, the state variables for model predictive controllers do not have equivalent measurements in the process and therefore, method one is not possible.

For each of the three initial condition determination methods, two methods of searching for new model parameters are used. One method involves using feedback cultivation to determine the inaccurate model parameter and re-identifying only that parameter. The second method by-passes the feedback cultivation algorithm and re-identifies all the model parameters at once. The frequency of re-identification for the second method is the same as that used for the feedback cultivation method. The second re-dentification method along with the second method of determining the initial condition is used by Jang [36].

5.2.3.1 Case 1: IC Method 1, Search Method 1

Figure 5.18 shows the results of using feedback cultivation to identify the inaccurate model parameters and the latest available measurement for the initial condition. This method works the best for the first-order example. In the figure, the process output is overlayed with the model predicted output. The model predicted output tracks the setpoint changes shown in Figure 5.17. The process output deviates from the model predicted output and the setpoint when errors are present between the process and model.

The error between the process and the model occurs for one of two reasons. First, the error may exist because the feedback cultivation algorithm does not yet have sufficient data available to perform the model re-identification. The shaded regions between the process output curve and the model predicted output curve in Figure 5.18 indicates this error type. Second, the error may exist after the model is re-identified but not accurately enough. In this case, little or no error is present after re-identification of the model.

Figure 5.19 shows the model re-identification results for Case 1. As indicated, re-identification occurs three times, corresponding to the three process disturbances. Each identification correctly re-identifies the inaccurate parameter to the process value of the parameter. In the case of the time constant change, the feedback cultivation algorithm takes twice as long to determine that the time constant is inaccurate. This is because changes in the time constant tend to be canceled in the first-order model equation.

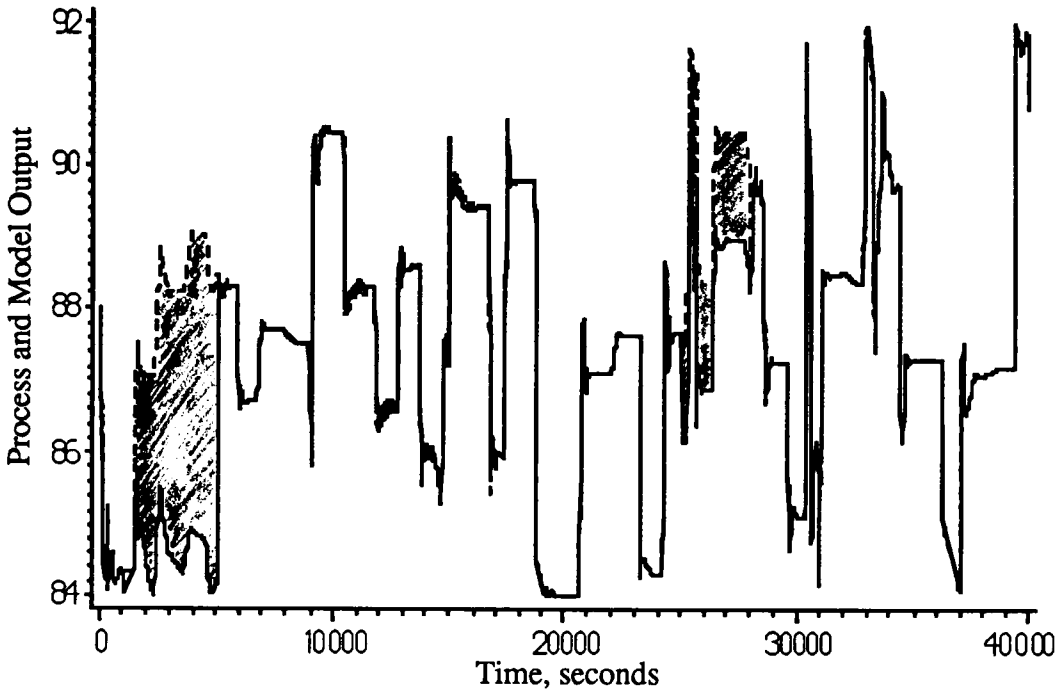


Figure 5.18: Plot of process and model predicted output for Case 1 model re-identification.

Examination of Equation 5.5 shows that the time constant appears in both terms on the right side of the equation but with opposite signs. Figure 5.18 shows that changing the process time constant affects the process far less significantly than a change in the process gain or the occurance of an unmeasured disturbance.

5.2.3.2 Case 2: IC Method 1, Search Method 2

Figure 5.20 shows the results of searching for all parameters during the model re-identification step and using the latest measurement for the initial condition. This method is similar to the method used by Jang [36]. The

ID Phase Execution Time, sec.	Search Parameter	Model Parameters After Search τ	K	*bias*	Initial Condition $x(0)$	Actual Output y
0	initial	1146.5	342.6	0	88	88
5040	K	1146.5	325.6	0	84.2	84.2
17040	τ	1260.5	325.6	0	85.9	85.9
28040	*bias*	1260.5	325.6	-1.5	88.2	88.2

Process Parameter Changes:

	Process Parameter Values		
Time, sec.	τ	K	*du*
0	1146.5	342.6	0
1000	1146.5	**325.6**	0
10000	**1260.5**	325.6	0
25000	1260.5	325.6	**-1.5**

Figure 5.19: Model re-identification results for Case 1.

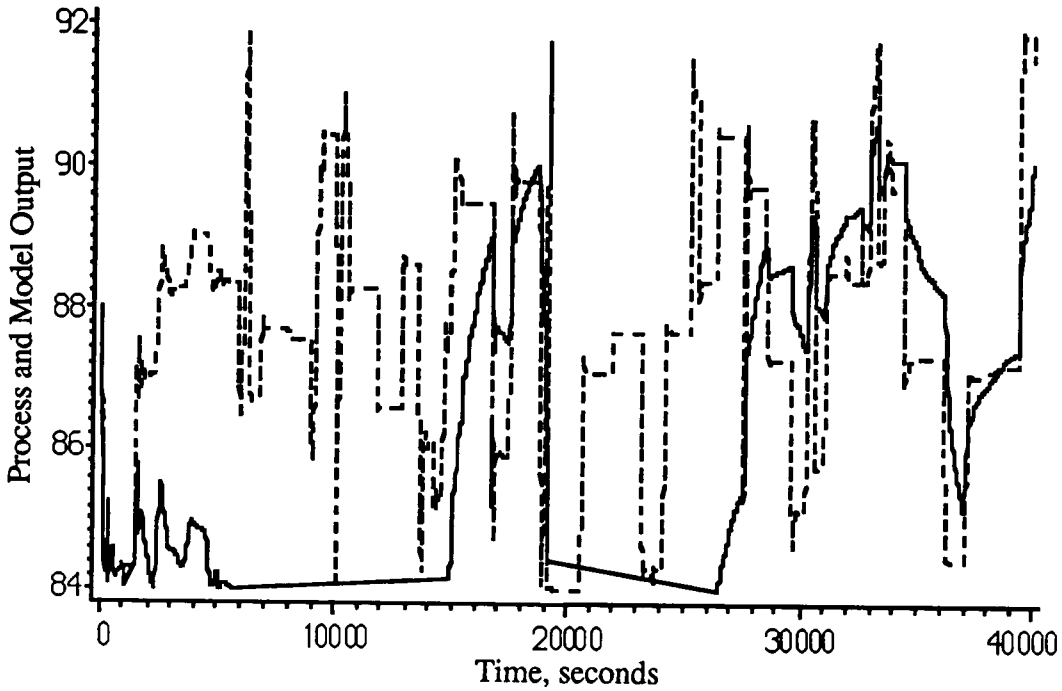


Figure 5.20: Plot of process and model predicted output for Case 2.

dashed line in the figure represents the model predicted value of the process output and the solid line represents the actual process output. The model predicted output tracks the setpoint changes shown in figure 5.17. As indicated in Figure 5.20, the process output fails to follow the setpoint changes during most of the simulation duration.

Figure 5.21 shows the model re-identification results for Case 2. The data shown in Figure 5.21 indicate that the model re-identification fails completely to accurately re-identify the model parameters. The reason for this method failing so completely to re-identify the model is not known. Failure of the re-identification to function properly results in the nonlinear optimal controller to function poorly resulting in large deviations of the

ID Phase Execution Time, sec.	Search Parameter	Model Parameters After Search			Initial Condition	Actual Output
		τ	K	*bias*	*x(0)*	*y*
0	initial	1146.5	342.6	0	88	88
6040	All	573.2	513.4	2.9	102.5	82.2
10040	All	286.6	410.3	-2.8	68.1	55.7
13540	All	429.9	332.5	-2.9	73.9	71.3
19040	All	286.1	432.1	-1.6	107	87.6
23540	All	323.6	342.2	-2.3	68.2	66.6
27540	All	325.5	342.2	-2.4	87.6	85.5
32540	All	325.5	327.1	-2.4	91.7	89.5

Process Parameter Changes:

	Process Parameter Values		
Time, sec.	τ	K	*du*
0	1146.5	342.6	0
1000	1146.5	**325.6**	0
10000	**1260.5**	325.6	0
25000	1260.5	325.6	**-1.5**

Figure 5.21: Model re-identification results for Case 2.

process output from the target setpoint values.

In this case, the errors are too large to distinguish between errors due to the delay accumulating feedback cultivation data and errors due to an improperly re-identified model. No distinction is made in Figure 5.20 between the two types of errors.

5.2.3.3 Case 3: IC Method 2, Search Method 1

Figure 5.22 shows the results of using feedback cultivation to select which model parameter to re-identify. In this case, the initial condition is also a search variable of the re-identification. As in Case 1, the errors due to the delay in accumulating feedback cultivation data appear as shaded regions in Figure 5.22. In general, the nonlinear optimal control algorithm keeps the process output close to the setpoint after model parameter re-identification.

Figure 5.23 shows the feedback cultivation results for Case 3. The results indicate that the model re-identification occurred as frequently as allowed by the model re-identification horizon, approximately every 3500.0 seconds, during the simulation. The first time the model re-identification occurs, feedback cultivation correctly determines that the model gain term is inaccurate. The new value of the model gain, $K_m = 328.1$, determined by the model re-identification is close to the value of the process gain of $K = 325.6$ with a deviation of 0.7%. This deviation is enough to cause problems during the rest of the simulation.

Feedback cultivation indicates that re-identification of the model time

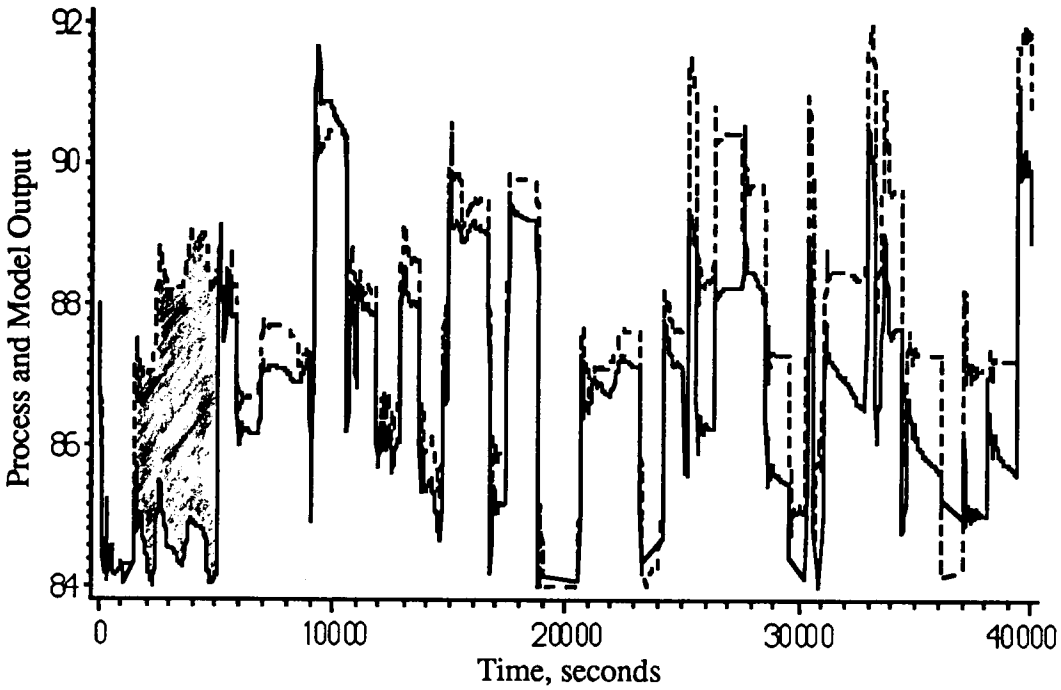


Figure 5.22: Plot of process and model predicted output for Case 3.

constant is necessary at a time of 8540.0 seconds to eliminate the deviation between the model predicted output and the actual process output even though the process time constant remains unchanged. The model re-identification incorrectly changes the model time constant to a value of $\tau_m = 1719.8$.

After changing the process time constant at a time of 10000.0 seconds, the model re-identification algorithm settles on a value for the model time constant of $\tau_m = 1220.3$ at a time of 24040.0 seconds. This is 3.2% off the process time constant of $\tau = 1260.5$. In the time between changing the process time constant and this last re-identification, the feedback cultivation algorithm modifies the time constant 3 times before settling on the final

ID Phase Execution Time, sec.	Search Parameter	Model Parameters After Search			Initial Condition	Actual Output
		τ	K	*bias*	$x(0)$	y
0	initial	1146.5	342.6	0	88	88
5040	K	1146.5	328.1	0	84.3	84.2
8540	τ	1719.8	328.1	0	87.1	86.9
11040	τ	1230.7	328.1	0	86.9	86.8
15540	τ	1361.5	328.1	0	89.1	88.9
21040	τ	1365.5	328.1	0	86.6	86.5
24040	τ	1220.3	328.1	0	83.8	83.6
27540	*bias*	1220.3	328.1	-0.03	89.1	88.4
31040	*bias*	1220.3	328.1	-0.06	85.9	85.1
34540	*bias*	1220.3	328.1	-0.09	85.7	84.9
38040	*bias*	1220.3	328.1	-0.12	85.9	85

Process Parameter Changes:

	Process Parameter Values		
Time, sec.	τ	K	*du*
0	1146.5	342.6	0
1000	1146.5	**325.6**	0
10000	**1260.5**	325.6	0
25000	1260.5	325.6	**-1.5**

Figure 5.23: Model re-identification results for Case 3.

value. The changes in the model time constant are shown in Figure 5.23.

Finally, after imposing an unmeasured disturbance at a time of 25000.0 seconds, the model re-identification fails to accurately determine a new value for the model output bias term after four attempts. The errors in the model time constant and the model gain probably caused the model re-identification to fail in accurately accounting for the unmeasured disturbance.

This case demonstrates the sensitivity of the first-order process and nonlinear optimal controller to small discrepancies between parameters. This is expected since both the process and the model contain so few parameters and the simulation does not incorporate any noise. Results presented in Chapter 6 for a case with noise shows that parameter descrepancy sensitivity is reduced in the presence of noise.

5.2.3.4 Case 4: IC Method 2, Search Method 2

Figure 5.24 shows the results of searching for all model parameters and the initial condition during the model re-identification. This model re-identification method is the same as the method employed by Jang [36]. The results shown in the figure indicate that the process output tracks the model predicted output closely after model parameter re-identification. The shaded areas between the two curves represent the error due to the time needed to collect data to use in the model re-identification. In this case, the same amount of time (3500.0 seconds) before re-identification is required as in the case for using feedback cultivation to select the model parameter to be

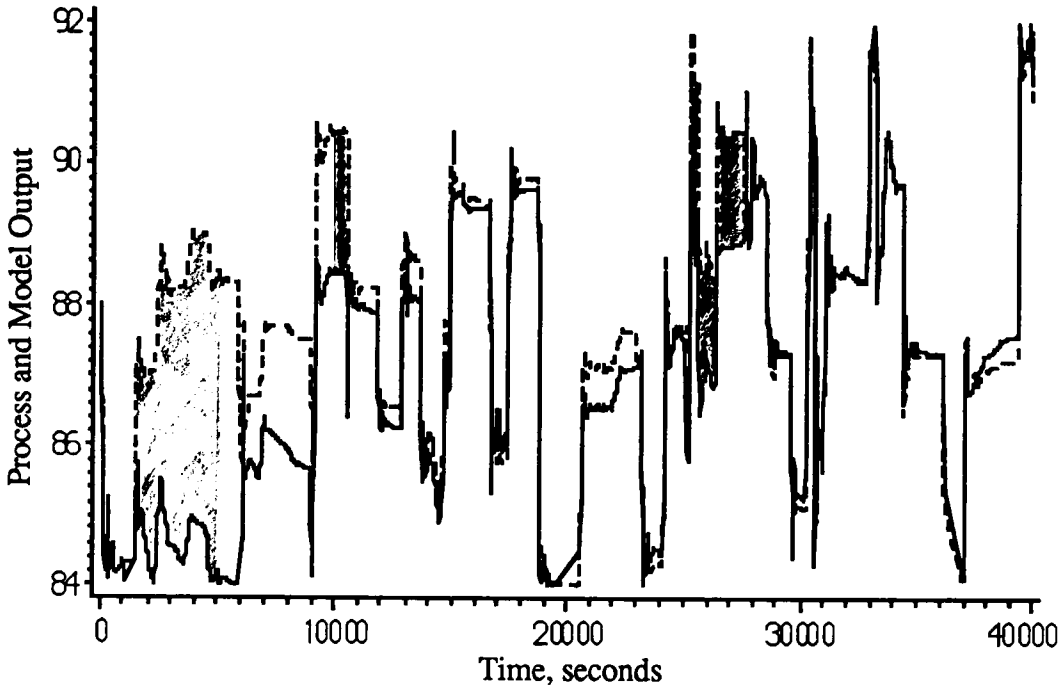


Figure 5.24: Plot of process and model predicted output for Case 4.

identified.

Figure 5.25 show the model re-identification results for Case 4. In this case, the model re-identification performs much better than in Case 2. First, the model re-identification determines a better approximation of the model gain after the first attempt at a time of 6040.0 seconds. Also the re-identification determines a value of the initial condition close to the actual value. Inspite of the closeness of the model predicted output to the process output, the model re-identification algorithm fails to determine accurate values for the model time constant and the model output bias term when the process time constant and the unmeasured disturbance are changed at times of 10000.0 seconds and 25000.0 seconds respectively.

ID Phase Execution Time, sec.	Search Parameter	Model Parameters After Search τ	K	*bias*	Initial Condition *x(0)*	Actual Output *y*
0	initial	1146.5	342.6	0	88	88
6040	All	1146.6	333.4	-0.06	82.7	82.2
10540	All	1146.6	327.8	-0.09	87	86.9
15040	All	1146.7	326.1	-0.09	90	89.6
19040	All	1147.1	327.8	-0.1	84.4	84.4
23040	All	1147.1	325.9	-0.1	87.2	87.1
27540	All	1147.1	320.6	-0.12	89	88.9
37540	All	1147.4	318.7	-0.12	86.9	86.9

Process Parameter Changes:

	Process Parameter Values		
Time, sec.	τ	K	*du*
0	1146.5	342.6	0
1000	1146.5	**325.6**	0
10000	**1260.6**	325.6	0
25000	1260.5	325.6	**-1.5**

Figure 5.25: Model re-identification results for Case 4.

This case shows that the nonlinear optimal controller is most sensitive to changes on the model gain. Examination of the first-order model equation indicates this to be true. As mentioned previously, the effects of changing the model time constant tend to cancel in Equation 5.5. This is not true for the model gain.

5.2.3.5 Case 5: IC Method 3, Search Method 1

Figure 5.26 shows plots of the model predicted output and the actual process output for the case of using feedback cultivation and a much larger window of data in the model re-identification. This case uses the model determined value of the initial condition during the entire simulation. By using a larger window of data in the model re-identification, the effect of inaccuracies in the initial condition are minimized. For the previous four cases, the model re-identification window size is the same as the feedback cultivation window size, 50 data points representing 50 control moves. In this case, the previous 350 data points are used in the model re-identification.

As in the previous cases, shaded areas between the model predicted output and the process output indicate deviations due to the delay in accumulating enough data to perform the model re-identification. Figure 5.26 indicates that the process output tracks the model predicted output and, therefore, the process setpoint with some offset each time after model re-identification.

In this case, the simulation starts with the initial condition equal to

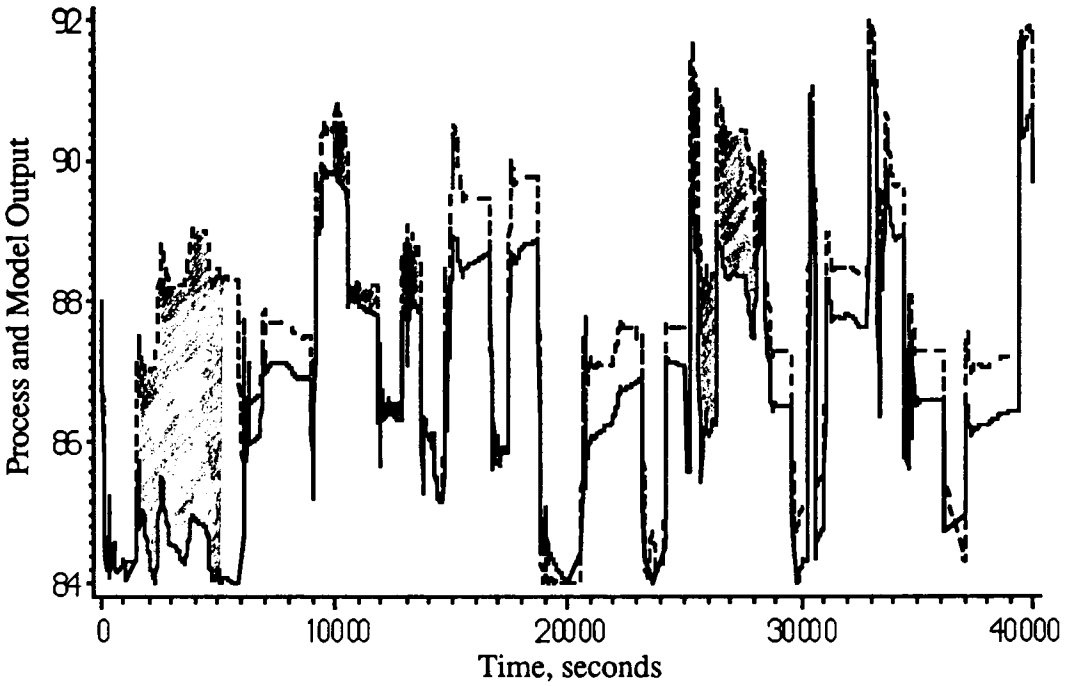


Figure 5.26: Plot of process and model predicted output for Case 5.

the process output. After model re-identification, an updated value of the initial condition is determined by integrating the model over the time length of the model re-identification horizon using the re-identified model parameters.

Figure 5.27 show the model re-identification results for Case 5. As shown, the model re-identification accurately re-identifies the model gain at a time of 6040.0 seconds. The new model gain of $K_m = 327.9$ is within 0.7% of the actual process gain of $K = 325.6$. Integrating the model with the new model gain over the re-identification window results in a value of 82.7 for the initial condition which is within 0.5 of the current process output of 82.2.

After the first model re-identification, the algorithm struggles to deal

ID Phase Execution Time, sec.	Search Parameter	Model Parameters After Search			Initial Condition	Actual Output
		τ	K	*bias*	*x(0)*	*y*
0	initial	1146.5	342.6	0	88	88
6040	K	1146.5	327.9	0	82.7	82.2
10040	τ	874.5	327.9	0	90.9	89.8
15540	τ	874.5	327.9	0	89.5	88.4
23040	τ	1311.7	327.9	0	87.4	86.9
28040	*bias*	1311.7	327.9	-1.4	89.9	87.8
33040	τ	1228.9	327.9	-1.4	93.3	91.1
36540	τ	1102.4	327.9	-1.4	83.4	82

Process Parameter Changes:

	Process Parameter Values		
Time, sec.	τ	K	*du*
0	1146.5	342.6	0
1000	1146.5	**325.6**	0
10000	**1260.5**	325.6	0
25000	1260.5	325.6	**-1.5**

Figure 5.27: Model re-identification results for Case 5.

with the small discrepancy of the model gain and the model initial condition value. At a time of 10040.0, just after the process time constant is changed, the model re-identifies the model time constant to a value of $\tau_m = 874.5$. This value does not compare to the old value of the process time constant of $\tau = 1146.5$ or the new value of the process time constant of $\tau = 1260.5$. Also, the initial condition now deviates from the process output by a value of 1.1.

The model time constant is re-identified to a value of $\tau_m = 1311.7$ at a time of 23040.0 seconds. This value more closely approximates to within 4.0%, the value of the process time constant. Also, the deviation between the initial condition and the process output reduces to 0.7.

At a time of 28040.0 seconds, re-identification of the model output bias term occurs. The actual value of the unmeasured disturbance is -1.5. The resulting value of the model output bias term, $bias = -1.4$, is nearly correct. Also, the deviation between the initial condition and the process output is 0.8, not much different than before.

Finally, at a time of 33040.0 seconds, re-identification of the model time constant occurs bringing the value to within 2.5% of the process time constant. Also, the value of the deviation between the initial condition and the process output reduces to 0.6.

This case demonstrates that using an extended re-identification window works well with the feedback cultivation algorithm. The drawback is the large amount of data needed to perform the model re-identification. While this large amount of data is accumulating, offset occurs between the

process output and the setpoint. This presents a problem when multiple parameter errors are present such as between a time of 10000.0 seconds and 30000.0 seconds when the feedback cultivation algorithm inaccurately corrected the model time constant before the unmeasured disturbance occurred. Inspite of these problems, the model re-identification performed well by eventually determining nearly accurate model parameters.

5.2.3.6 Case 6: IC Method 3, Search Method 2

Figure 5.28 shows the results for using an extended model re-identification window while searching for all the model variables simultaneously. The size of the extended window used for this case is 350 data points, the same as that used in Case 5. In Figure 5.28, the dashed line indicates the value of the model predicted output which tracks the setpoint as in the previous cases. The solid line indicates the value of the process output during the simulation. Also indicated in the figure are the shaded regions between the two lines indicating deviations due to the delay in accumulating data for the model re-identification algorithm.

Figure 5.29 shows the model re-identification results for Case 6. These results indicate that the model re-identification does a poor job of re-identifying the first time at a time of 6040.0. After this initial poor re-identification, the model re-identification struggles to determine accurate values for the three model parameters and hence the initial condition for the model. By the end of the simulation, the model re-identification determines

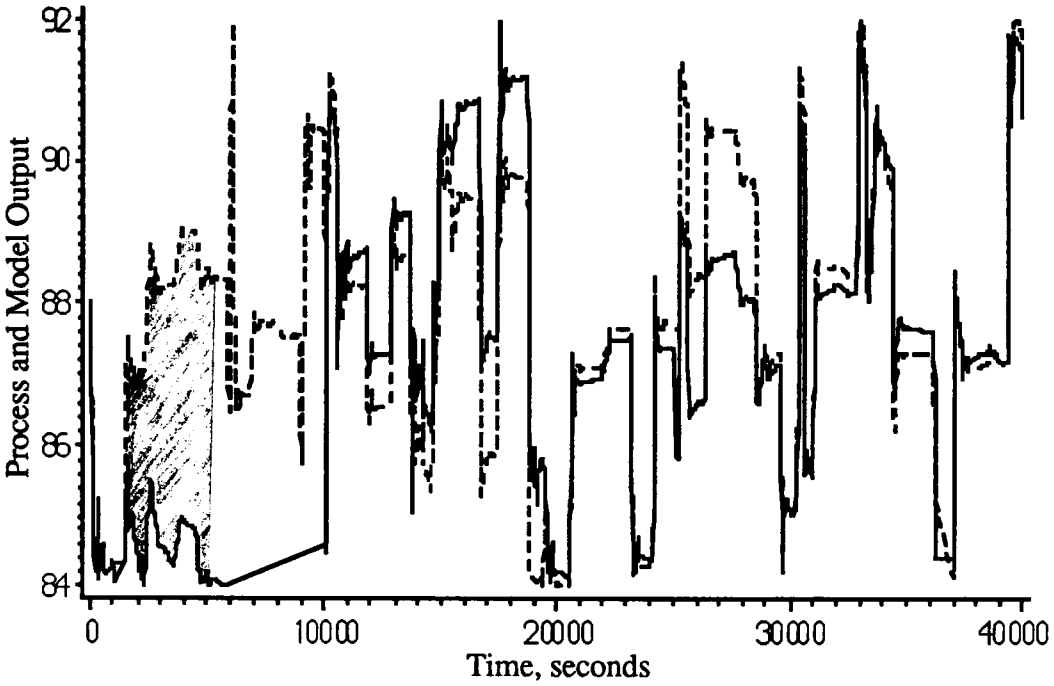


Figure 5.28: Plot of process and model predicted output for Case 6.

model parameters that result in very little deviation between the model predicted output and the process output as indicated in Figure 5.28.

Comparing Figures 5.26 and 5.28 indicate that both search methods with the extended model re-identification window work reasonably well to limit the deviation between the model predicted output and the process output which eliminates offset between the process output and the setpoint. Comparing the results presented in Figures 5.27 and 5.29 indicates that using feedback cultivation to selectively re-identify the model parameters keeps the model parameters closer to the process parameter values. The main difference between the two methods lies in the resulting value of the initial condition. In Case 5, the initial condition never deviates more than 1.1 from

ID Phase Execution Time, sec.	Search Parameter	Model Parameters After Search			Initial Condition	Actual Output
		τ	K	*bias*	*x(0)*	*y*
0	initial	1146.5	342.6	0	88	88
6040	All	1120.4	364.2	0.9	91.4	82.2
10040	All	1120.3	320.4	0.7	78.8	80.1
15540	All	1120.4	318.2	0.5	88.1	90
19540	All	1122.1	323.2	0.7	84.6	85.6
24540	All	1122.1	323.5	0.8	87	87.4
28540	All	1122.2	318.1	0.8	87.2	87.8
33540	All	1122.2	316.5	0.7	88.6	89.5
38540	All	1122.2	317.4	0.7	86.7	87.3

Process Parameter Changes:

	Process Parameter Values		
Time, sec.	τ	K	*du*
0	1146.5	342.6	0
1000	1146.5	**325.6**	0
10000	**1260.5**	325.6	0
25000	1260.5	325.6	**-1.5**

Figure 5.29: Model re-identification results for Case 6.

the actual process output value. In Case 6, the deviation is almost 10.0 after the first application of the model re-identification algorithm.

5.3 Discussion

The results presented in Sections 5.2.1, 5.2.2, and 5.2.3 demonstrate various aspects of the complete nonlinear optimal control algorithm. Section 5.2.1 shows how changing the control horizon and the prediction horizon affects the nonlinear optimal controller. Section 5.2.2 presents the results classifying training and test data with a trained neural network. Section 5.2.3 compares six methods of implementing the feedback cultivation algorithm with the nonlinear optimal controller to update the model to eliminate offset between the process output and the setpoint.

In Section 5.2.1, the effects of changing the control horizon and the prediction horizon demonstrate that changing these parameters affect the process in much the same way as changing the controller tuning parameters for a traditional PID-type controller. Setting the prediction horizon to high caused the process to return to setpoint faster but with more oscillations. This is similar to having too high a reset rate or having too low an integral time in a PID-type controller. Setting the control horizon too high caused the process to respond to setpoint changes in the same amount of time but with more oscillations. Setting the control horizon too low again caused the process to respond to setpoint changes in the same amount of time but this

time with an underdamped response. The same effect occurs when changing the controller gain term in a PID-type controller.

Section 5.2.2 presents the results of training the neural network with five sets of training data. The results show that the network accurately learns the training data and classifies 100% of the data correctly. The results also show that the network does not function quite as well for the test data sets where the magnitude of the model parameter problems are different. Selection of different feedback cultivation variables may improve the performance of the feedback cultivation algorithm.

The results presented in Section 5.2.3 indicate that using the feedback cultivation and model re-identification methods of Case 1 worked the best to accurately determine new model parameters. Using available state variable measurements greatly improves the performance of the model re-identification algorithm for this example. These results also indicate that poor model re-identification occurs when searching for all model parameters at once. In Case 2, the model re-identification fails completely.

The results in Section 5.2.3 also demonstrate the sensitivity of the nonlinear optimal control algorithm to the accuracy of the model parameters. Significant deviations from setpoint result from seemingly small discrepancies between model and process parameter values. This is especially true for the deviation between the model and the process gain term and differences between the model initial condition value and the actual process output value.

This example provides a demonstration of the nonlinear optimal

control algorithm and the feedback cultivation algorithm for the first-order process. The simple nature of the process and the model make detailed analysis easier. Also, a lack of noise in the simulations made the analysis easier but caused problems with the feedback cultivation algorithm. The next chapter presents a demonstration of the nonlinear optimal control algorithm in conjunction with the feedback cultivation algorithm for a more complex process.

CHAPTER 6

The Reactor Example

The nonlinear continuous stirred-tank reactor (CSTR) described in this chapter provides a demonstration of the nonlinear optimal control algorithm with feedback cultivation including adaptive filtering. Section 6.1 describes the CSTR operating conditions, the structure of the nonlinear optimal controller, the objectives of feedback cultivation, and the adaptive filter parameters for this example. Section 6.2 presents the results of the application of the complete nonlinear optimal control algorithm to the CSTR including a comparison of the adaptive filter algorithm to a first-order filter algorithm.

6.1 System Description

6.1.1 The Continuous Stirred-tank Reactor

The nonlinear continuous stirred-tank reactor (CSTR) given by Smith and Corripio in Section 9.5 [65] forms the basis for the CSTR used in this example. A schematic of the jacketed reactor is given in Figure 6.1. Catalyzed second-order irreversible characteristics govern the primary

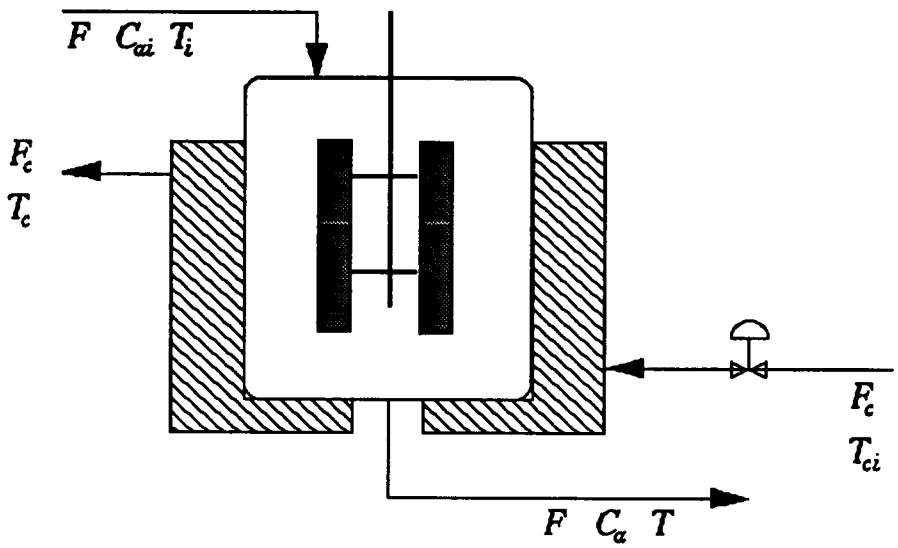


Figure 6.1: Schematic diagram of continuous stirred tank reactor.

reaction for the reactor:

$$A + catalyst \xrightarrow{k_1} P \tag{6.1}$$

where k_1 is the primary reaction rate coefficient, A represents the reactant, and P represents the desired product.

A secondary reaction of feedstream impurities affects the material and energy balances in the CSTR. The secondary reaction follows catalyzed first-order dynamics:

$$B + catalyst \xrightarrow{k_2} Q \tag{6.2}$$

where k_2 is the secondary reaction rate coefficient, B represents the feed stream impurities and Q represents undesirable by-products.

Figure 6.2 shows the mass and energy balance equations used to simulate the CSTR. The CSTR simulation consists of six differential equations: four material balance equations, one energy balance of the reactor contents, and one energy balance of the cooling jacket contents. Also, the CSTR equation set contains two equations defining the reaction rate coefficients and one equation describing the equal percentage control valve on the cooling jacket inlet stream. The definitions and units for all the variables in the CSTR simulation equations are given in Figure 6.3.

The equations for the CSTR given in Figure 6.2 assume that the contents of both the reactor and the jacket are perfectly mixed. The physical properties and volumes are assumed to be independent of the temperature and pressure of the system. The energy balance equation for the cooling jacket contains a term that accounts for heat lost to the surroundings due to changes in the ambient temperature.

The temperature of the contents in the reactor is controlled by manipulating the flowrate of the coolant entering the jacket. The equal percentage control valve accepts the controller output signal, m, determined by the nonlinear optimal control algorithm described in the next section.

Simulations of the CSTR involve disturbances of the inlet feed flowrate, F, the inlet temperature, T_i, the inlet coolant temperature, T_{ci}, and the inlet reactant concentration, C_{ai}. Typical disturbance profiles for these four variables are shown in Figures 6.4 through 6.7.

The measurements of the four disturbance variables and the three

Mass Balance on Reactant A:

$$\frac{\partial C_a}{\partial t} = \frac{F}{V}(C_{ai} - C_a) - k_1 \alpha C_a^2$$

Energy Balance on Reactor:

$$\frac{\partial T}{\partial t} = \frac{F}{V}(T_i - T) - \frac{\Delta H_r \alpha}{\rho c_p} k_1 C_a^2 - \frac{UA}{V \rho c_p}(T - T_c) - \frac{\Delta H_{r2} \alpha}{\rho c_p} k_2 C_b$$

Jacket Energy Balance:

$$\frac{\partial T_c}{\partial t} = \frac{UA}{V_c \rho_c c_{pc}}(T - T_c) - \frac{F_c}{V_c}(T_c - T_{ci}) - \frac{K_1}{V_c \rho_c c_{pc}} \Delta T_{lm}$$

$$\Delta T_{lm} = \frac{(T_{amb} - T_{ci}) - (T_{amb} - T_c)}{\ln \frac{T_{amb} - T_{ci}}{T_{amb} - T_c}}$$

Mass Balance of Contaminant:

$$\frac{\partial C_b}{\partial t} = \frac{F}{V}(C_{bi} - C_b) - k_2 \alpha C_b$$

Mass Balance of Product:

$$\frac{\partial C_p}{\partial t} = k \alpha C_a^2 - \frac{F}{V} C_p$$

Mass Balance of Side Reaction Product:

$$\frac{\partial C_c}{\partial t} = k_2 \alpha C_b - \frac{F}{V} C_c$$

Reaction Rate Coefficients:

$$k_1 = k_0 e^{-\frac{E}{R(T+273.16)}} \qquad k_2 = k_{02} e^{-\frac{E_2}{R(T+273.16)}}$$

Equal % Control Valve:

$$F_c = F_{c,\max} \gamma^{-m} \quad 0 \le m \le 1$$

m from Nonlinear Optimal Control Algorithm

Figure 6.2: Mass and energy balance equations for CSTR.

C_a = concentration of the reactant in reactor, kgmole / m^3

C_{ai} = inlet concentration of reactant, kgmole / m^3

C_b = concentration of contaminant in reactor, kgmole / m^3

C_{bi} = inlet concentration of contaminant, kgmole / m^3

C_c = concentration of impurity in reactor, kgmole / m^3

C_p = concentration of desired product, kgmole / m^3

T = reactor temperature, ^{0}C

T_c = jacket temperature, ^{0}C

T_i = inlet temperature, ^{0}C

T_{ci} = inlet coolant temperature, ^{0}C

T_{amb} = ambient temperature, ^{0}C

F = feed flow rate, m^3 / s

V = reactor volume, m^3

k_1 = primary reaction rate coefficient, m^3 / kgmole s

k_2 = side reaction rate coefficient, m^3 / kgmole s

ΔH_r = primary heat of reaction, J / kgmole

ΔH_{r2} = heat of side reaction, J / kgmole

ρ = density of reactor contents, kgmole / m^3

c_p = reactant heat capacity, J / kgmole ^{0}C

U = overall heat transfer coefficient, J / s m^2 ^{0}C

A = heat transfer area, m^2

V_c = jacket volume, m^3

ρ_c = density of coolant, kg / m^3

c_{pc} = specific heat of coolant, J / kg ^{0}C

F_c = coolant flow rate, m^3 / s

m = controller output signal (0 to 1)

$F_{c,max}$ = maximum flow through control valve, m^3 / s

γ = valve rangeability parameter

α = catalyst activity (0 to 1)

k_0 = Arrhenius frequency parameter, m^3 / s kgmole

k_{02} = side reaction frequency parameter, m^3 / s kgmole

E = activation energy, J / kgmole

E = side reaction activation energy, J / kgmole

R = ideal gas law constant, 8314.39 J / kgmole K

K_1 = outside heat transfer constant term, J / s^2 ^{0}C

K_2 = heat of reaction and reaction rate approximation

Figure 6.3: Variable definitions for CSTR equations.

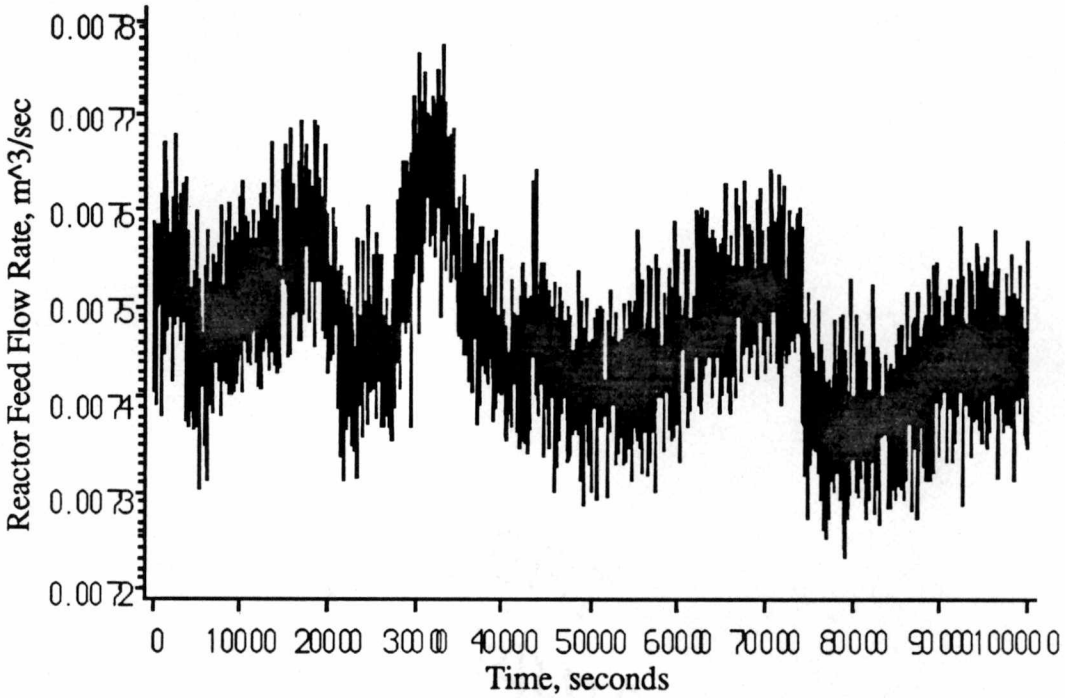


Figure 6.4: Disturbance profile for inlet feed flow rate, F.

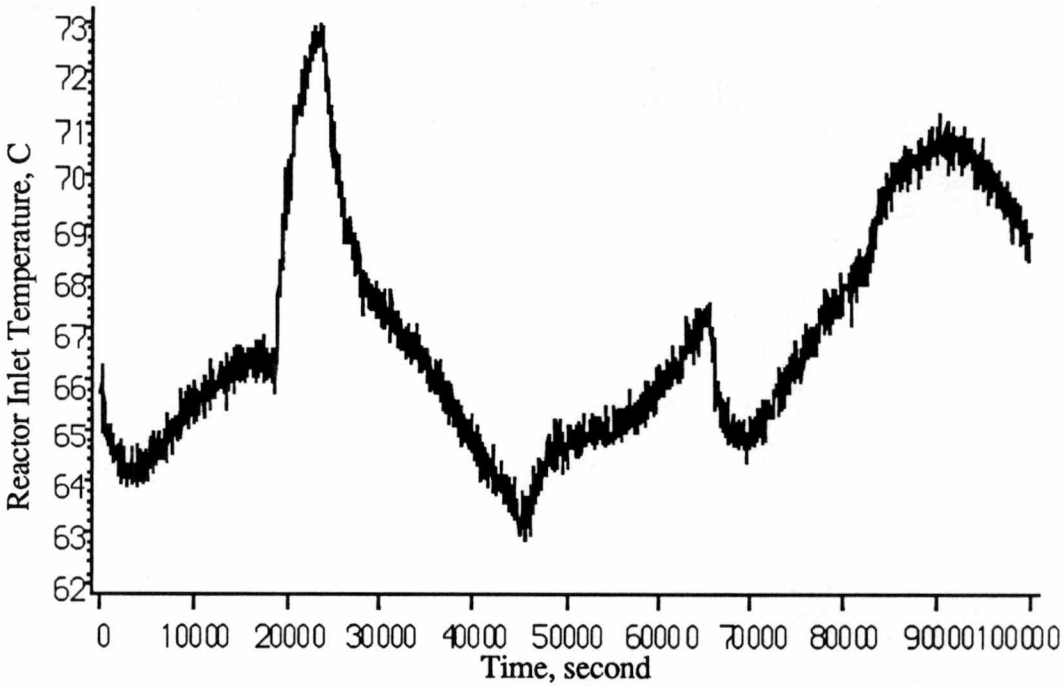


Figure 6.5: Disturbance profile for inlet temperature, T_i.

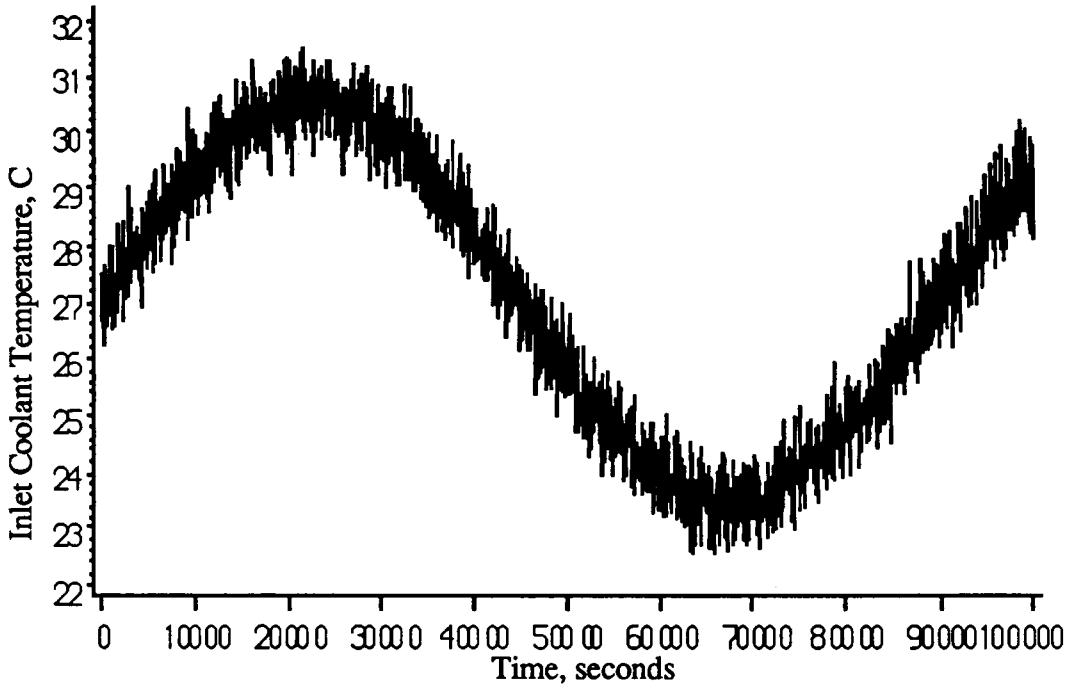


Figure 6.6: Disturbance profile for inlet coolant temperature, T_{ci}.

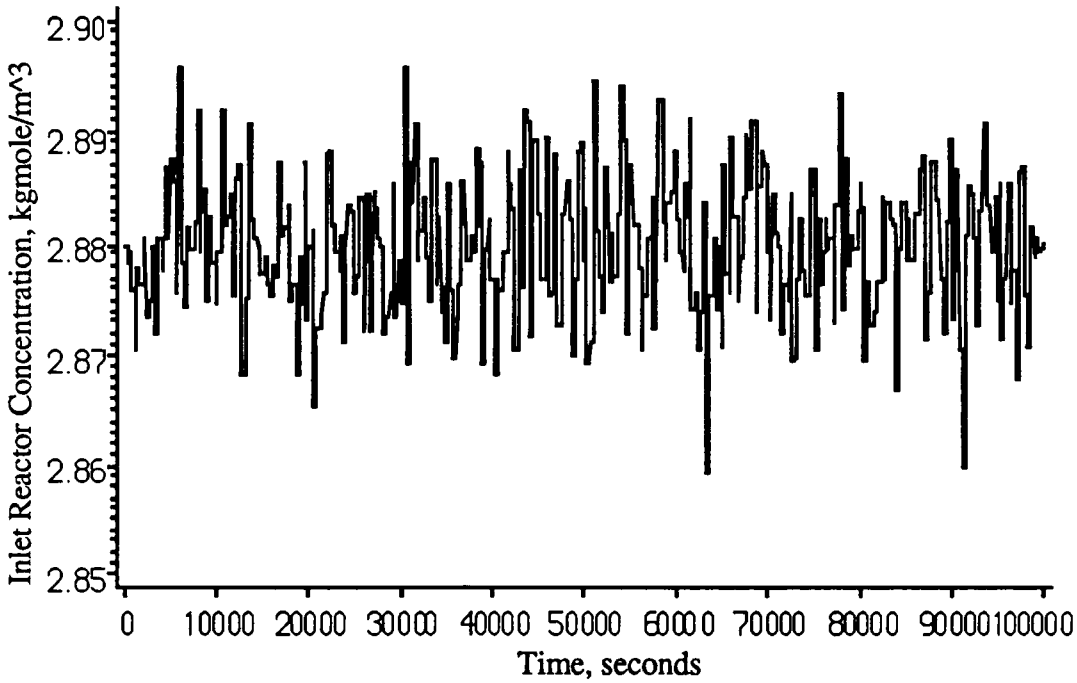


Figure 6.7: Disturbance profile for inlet feed concentration, C_{ai}.

output variables (T, T_c, and C_a) contain first-order measurement lags. Also, each of the measurement signals contains between 5% and 30% white gaussian noise. Besides first-order lags and noise, the inlet and outlet concentration measurements contain additional eight to ten minute measurement lags simulating the delay of on-line composition sensors.

Figures 6.8 through 6.11 show the resulting measured output variable values and coolant flowrate for a simulation of the CSTR with the disturbances shown in Figures 6.4 through 6.7. The duration of the simulation is about 28 hours. In this simulation, the nonlinear optimal control algorithm provides the control action for the equal percentage control valve. As indicated by Figure 6.8, the nonlinear optimal control algorithm maintains the temperature of the reactor contents at the desired setpoint of 88.0 ^{0}C with a variation of approximatly one degree. This simulation contains no unmeasured disturbances and the energy term accounting for heat losses to the surroundings in the reactor jacket energy balance is ignored. Also, no significant process-to-model mismatch is present.

The nonlinear CSTR equations are integrated using a fourth-order Runge-Kutta algorithm. The process time step used is $\Delta t_m = 2.0$ seconds. The temperature and flow measurements are assumed available to the nonlinear optimal controller at every time step. Concentration measurements are available after 8 to 10 minute delays.

For the sake of comparison, a simulation of the reactor is performed with the proportional-integral (PI) controller used in Smith and Corripio [65]

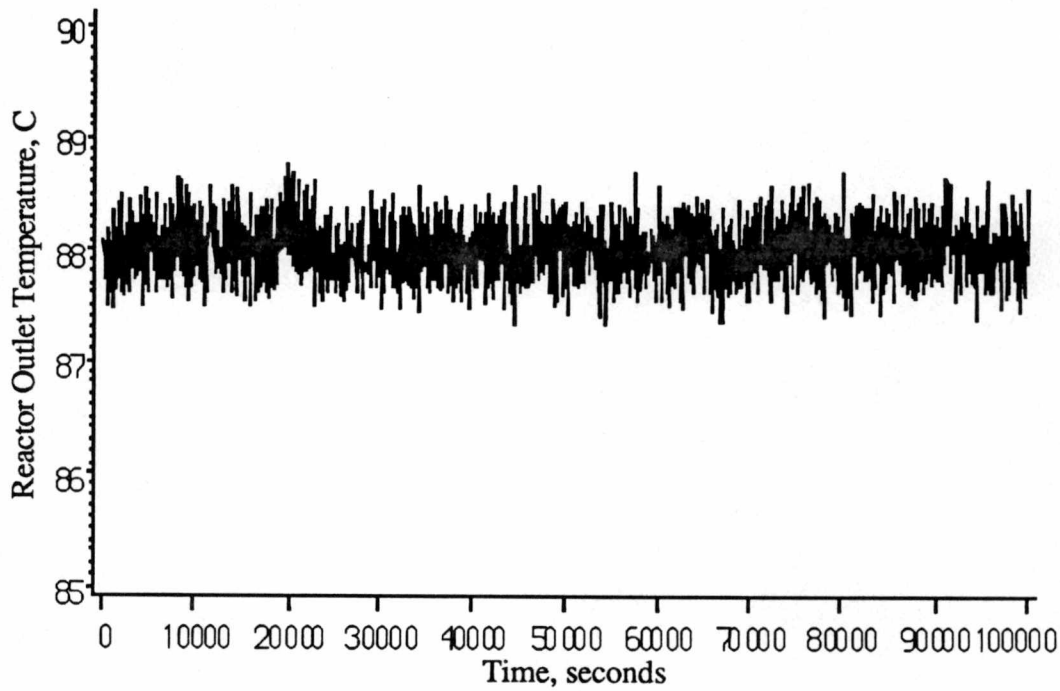


Figure 6.8: Simulation reactor temperature, T.

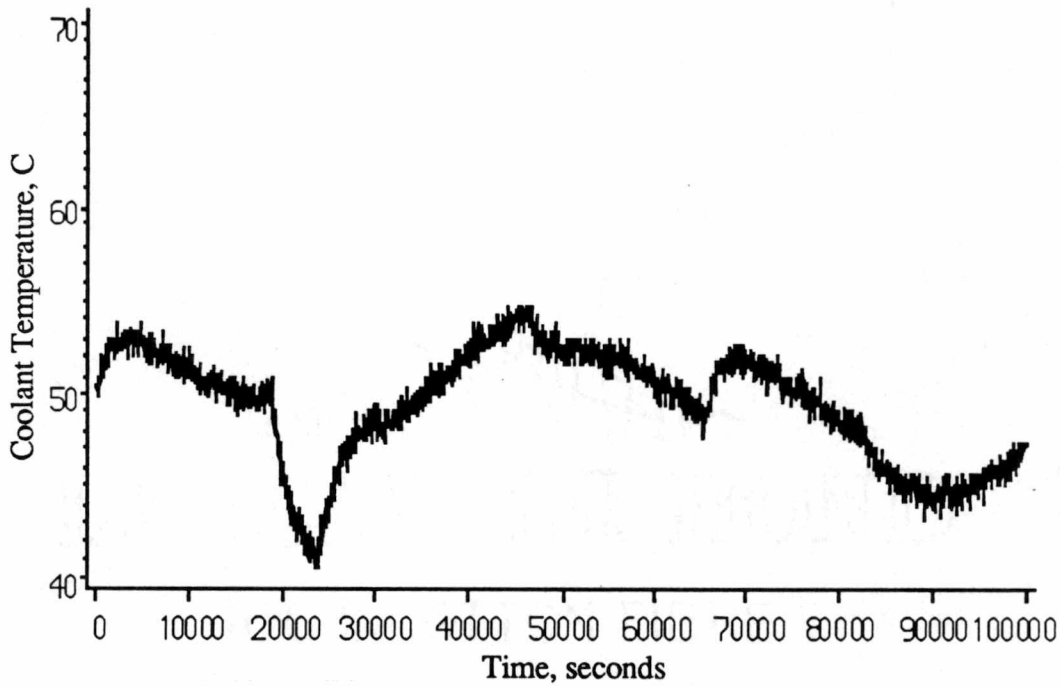


Figure 6.9: Simulation coolant temperature, T_c.

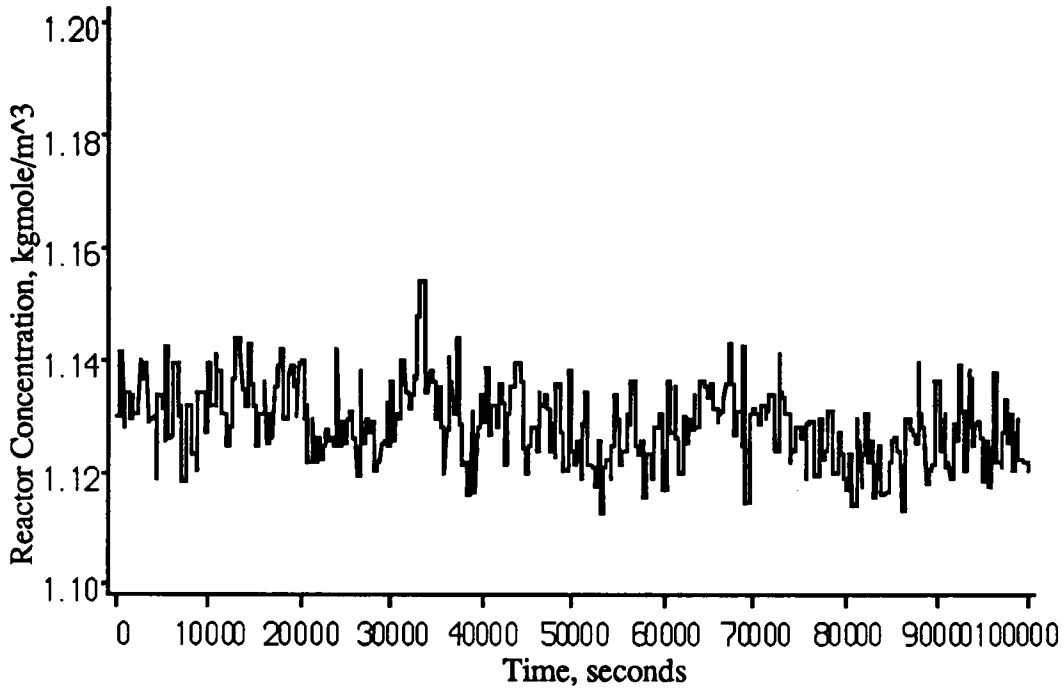


Figure 6.10: Simulation reactor concentration, C_a.

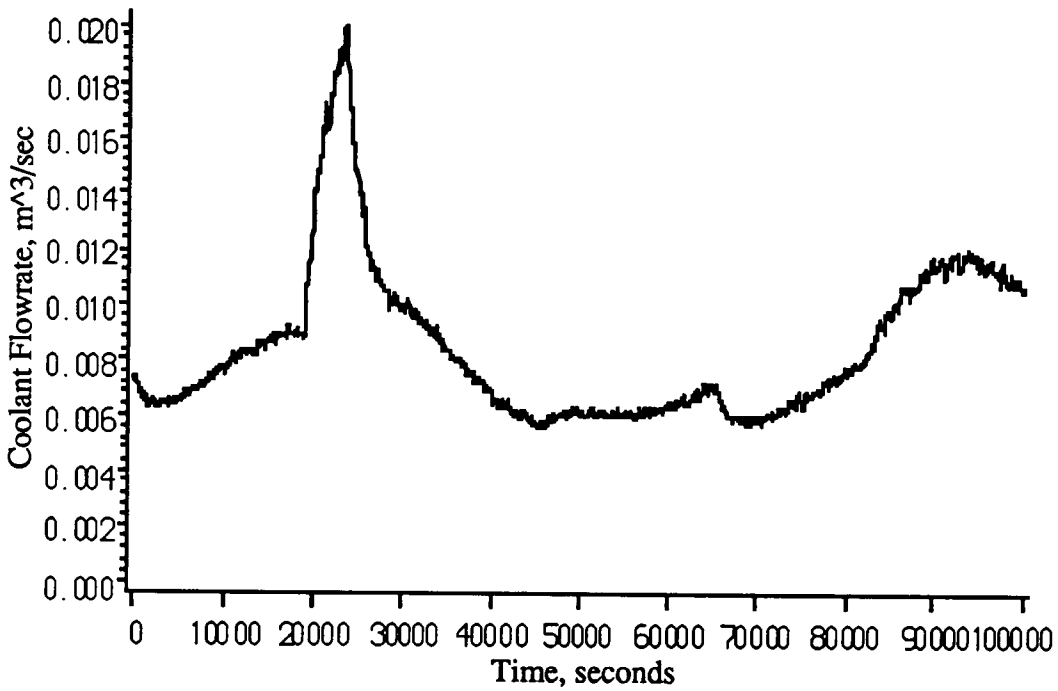


Figure 6.11: Simulation control variable value, coolant flowrate, F_c.

with the same modifications and parameters described above. Instead of an adaptive filter of the temperature measurement, a first-order filter with a filter constant of $\alpha = 0.95$ is used in this simulation to reduce the amount of noise in the temperature signal. Figure 6.12 shows the reactor outlet temperature and the coolant flow rate for this simulation. The results are similar to those shown in Figures 6.8 and 6.11 except that the noise in the temperature signal affects the response of the PI-controller more significantly.

6.1.2 The Reactor Controller

6.1.2.1 The Model

The nonlinear optimal control algorithm for the CSTR determines the future values for the control actions given by the vector $m = (m_1, m_2, ..., m_M)$ where M is the control horizon. The model of the CSTR used by the nonlinear optimal controller consists of a simplified version of the equation set given in Figure 6.2. Figure 6.13 shows the three differential equations, the primary reaction rate coefficient equation, and the equal percentage control valve equation that make up the CSTR model for the nonlinear optimal control algorithm. The variable definitions given in Figure 6.3 also apply to the variables shown in Figure 6.13. In future references, model parameters identical to process parameters in the model equations have a subscript 'm'.

The equations for the model of the CSTR differ from the simulation equations of the CSTR in several ways. First, the model only approximates the effect of the secondary reaction of impurities on the energy balance of the

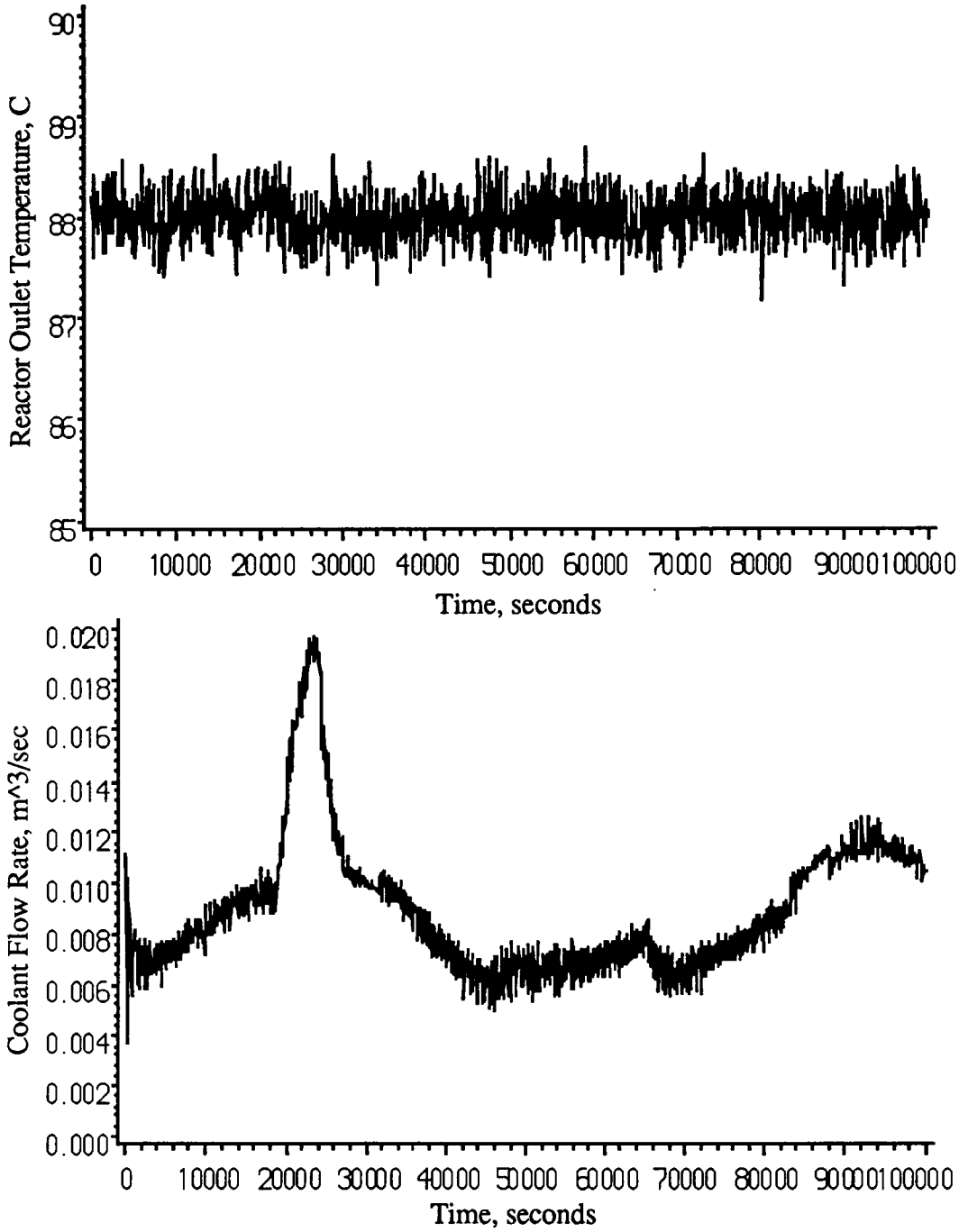


Figure 6.12: Results of a CSTR simulation with a PI-controller.

Mass Balance on Reactant A:

$$\frac{\partial C_a}{\partial t} = \frac{F}{V}(C_{ai} - C_a) - k_1 \alpha C_a^2$$

Energy Balance on Reactor:

$$\frac{\partial T}{\partial t} = \frac{F}{V}(T_i - T) - \frac{\Delta H_r}{\rho c_p} k_1 \alpha C_a^2 - \frac{UA}{V \rho c_p}(T - T_c) - \frac{K_2}{\rho c_p} \alpha C_{bi}$$

Jacket Energy Balance:

$$\frac{\partial T_c}{\partial t} = \frac{UA}{V_c \rho_c c_{pc}}(T - T_c) - \frac{F_c}{V_c}(T_c - T_{ci})$$

Reaction Rate Coefficient:

$$k_1 = k_0 e^{-\frac{E}{R(T+273.16)}}$$

Equal % Control Valve:

$$F_c = F_{c,\max} \gamma^{-m} \quad 0 \le m \le 1$$

m from Nonlinear Optimal Control Algorithm

Figure 6.13: Nonlinear optimal control algorithm model equations for CSTR.

reactor contents. The model ignores the effect of the secondary reaction of impurities on the mass balance of the contents in the reactor. Second, the model ignores heat losses to the surroundings due to changes in the ambient temperature that affects the energy balance of the cooling jacket contents.

The model also contains three equations defining the output variables from the reactor including bias terms to account for unmeasured process disturbances:

$$\hat{C}_{am} = C_{am} + bias_1 \tag{6.3}$$

$$\hat{T}_m = T_m + bias_2 \tag{6.4}$$

$$\hat{T}_{cm} = T_{cm} + bias_3 \tag{6.5}$$

where $bias_1$, $bias_2$, and $bias_3$ are the bias terms determined by the feedback cultivation algorithm to account for unmeasured process disturbances. Initially, all the bias terms equal zero.

The three differential equations shown in Figure 6.13 form the state equation set for the nonlinear optimal control algorithm. The three equations shown in 6.3 to 6.5 form the output equation set for the nonlinear optimal control algorithm.

6.1.2.2 The Optimal Control Algorithm

The nonlinear optimal control algorithm, using the state-and-output model shown in Figure 6.13 and Equations 6.3 through 6.5, determines the future control moves for the CSTR. The objective function of the optimization

contains one term related to the controlled variable for the CSTR and one term related to the manipulated variable:

$$\Phi = \sum_{i=1}^{P} \beta(T_{set} - \hat{T}_m)^2 + \gamma \left| m_1 - m_{1_{old}} \right| \tag{6.6}$$

where Φ is the objective function value, P is the prediction horizon value, β is the output weighting factor, T_{set} is the setpoint of the reactor temperature, $\hat{T}_m$ is the model predicted value of the reactor temperature, γ is the move suppression factor, m_1 is the current value of the control action, and $m_{1_{old}}$ is the most recently implemented control move value.

The controller tuning parameters used in the nonlinear optimal control algorithm have the following values:

$$\begin{aligned} P &= 50 \\ M &= 5 \\ \beta &= 1.0 \\ \gamma &= 50.0 \\ \Delta t_c &= 30.0 \text{ seconds} \end{aligned} \tag{6.7}$$

where Δt_c = the control action time step. The values of the prediction horizon, P, the control horizon, M, the output weighting factor, β, and the move suppression factor, γ are determined by trial-and-error simulation of the CSTR with the nonlinear optimal control algorithm. The trials attempt to find the smallest values of the horizons that provide control of the CSTR with little oscillatory, yet fast response to process disturbances. The smaller the values of the horizons, the faster the nonlinear optimal control algorithm

functions. The output weighting factor and the move suppression are determined such that the two terms in Equation 6.6 have similar magnitudes. In this example, the control action time step is 15 times the process integration time step of $\Delta t_m = 2.0$ seconds.

The generalized reduced gradient (GRG) method of optimization solves the optimal control problem for the future control moves for the CSTR. The equations of the model are integrated using a fourth-order Runge-Kutta integration method.

6.1.2.3 Feedback Cultivation

The nonlinear optimal control algorithm for the CSTR uses feedback cultivation to detect and correct for model errors and unmeasured process disturbances. Implementation of feedback cultivation involves three steps. The first step generates the reduced feedback cultivation dataset or data point. The second step classifies the reduced feedback cultivation data using a trained neural network. The third step interprets the results from the neural network classification and corrects any inaccurate parameters. Before these three steps operate in conjunction with the nonlinear optimal control algorithm, a number of initialization procedures must be executed.

First, the feedback cultivation variables that make up a feedback cultivation data point must be selected. Selection of the feedback cultivation variables is a matter of trial and error. The number of statistic and summary variables selected needs to be small enough so that the number of inputs to

the neural network is minimized. Fewer inputs results in a smaller network leading to faster and easier network training. The variables selected need to have different values or relationships for each class of data representing different process-to-model parameter mismatch errors. Also, the size of the feedback cultivation window and the frequency of implementation of feedback cultivation must be selected.

Figure 6.14 defines the feedback cultivation variables and shows the feedback cultivation parameters used in the CSTR example. In this example, selection of the feedback cultivation variables involved engineering judgement and examination of all possible summary and statistic variables for several classes of data. Other summary and statistic variables could have been selected giving equal or better results than those shown in Figure 6.14.

Some of the feedback cultivation variables used in this CSTR example involve ratios of statistic and summary variables. Using ratios applies natural scaling to some of the feedback cultivation variables. All the variables must be scaled to a range of -1.0 to 1.0 for input to the neural network.

The length of time corresponding to the size of the feedback cultivation window approximates the time constant of the CSTR. Each feedback cultivation data window contains 50 data points corresponding to the latest 50 control moves determined by the nonlinear optimal control algorithm. The simulation duration represented by the feedback cultivation window is 1500.0 seconds.

The second step of feedback cultivation requires a trained neural

$frac\_neg\_T$	Fraction of points with negative values for FB_T
$frac\_neg\_T_c$	Fraction of points with negative values for FB_T_c
err_avg	Average error in data window
fb_avg_T	Average value of FB_T in data window
$\frac{sae}{sse}$	Ratio of the sum of absolute errors and the sum of squared errors in the data window ($error = T_{set} - T$)
$\frac{ssfb\_T}{ssfb\_T_c}$	Ratio of the sum of square FB_T terms and the sum of square FB_T_c terms
$\frac{safb\_T}{safb\_C_a}$	Ratio of the sum of absolute values of FB_T terms and the sum of absolute values of FB_C_a terms
$\frac{safb\_T_c}{safb\_C_a}$	Ratio of the sum of absolute values of FB_T_c terms and the sum of absolute values of FB_C_a terms
$\frac{C_a\_avg}{C_{am}\_avg}$	Ratio of the average value of C_a and the average value of C_{am} in the data window

where

FB_T	$= T - T_m$, a feedback term
FB_T_c	$= T_c - T_{cm}$, a feedback term
FB_C_a	$= C_a - C_{am}$, a feedback term

- 50 historical data points per window
- The neural network structure: 9 input nodes, 50 nodes per hidden layer, and 14 output nodes (one per class plus one NULL class)

Figure 6.14: Feedback cultivation variables and parameters for CSTR.

network to perform the reduced feedback cultivation data classification. Before the network can be trained, training feedback cultivation datasets must be generated representing each of the problems desired to have the network learn.

Because the CSTR model contains many parameters, the controller's sensitivity to discrepancies in each of these parameters is tested. The neural network is trained to recognize discrepancies in the parameters that the nonlinear optimal controller is most sensitive.

The sensitivity study included only those parameters deemed most likely to change during the operation of the CSTR. Changes in the heats of reaction, $-\Delta H_{r1}$ and $-\Delta H_{r2}$, the activation energies, E_1 and E_2, and the Arrhenius coefficients, k_{01} and k_{02}, are not considered in the sensitivity studies.

The following list itemizes the parameters considered in the sensitivity studies including justification of the selection. Note that some parameters appear in groups as they always appear in the model equations given in Figure 6.13. For instance, a change in the reactor heat transfer coefficient, U, affects the controller in the same manner as a change in the heat transfer area, A. Therefore, these two terms are grouped together as the term UA in the sensitivity studies.

UA The heat transfer coefficient of the reactor may change due to fouling on the inside of the reactor or on the walls of the jacket surrounding the jacket.

$c_p\rho$ The heat capacity or the density of the reactor contents may change because of feed temperature or feed composition changes.

$c_{pc}\rho_c$ The heat capacity or the density of the coolant in the jacket surrounding may change during the operation of the reactor due to changes in the type of coolant used or changes in the inlet coolant temperature.

V The reactor contents volume may change due to flucuations in the reactor flow rate if level control fails to compensate for the flucuations.

V_c The volume of the coolant jacket contents may change because of excessive fouling or problems with the coolant flow control valve.

α_{cat} The catalyst activity may degrade during the operation of the reactor.

C_{bi} The inlet concentration of the secondary reaction reactant affects the material and energy balance of the reactor.

Figure 6.15 shows the results of simulating the CSTR using the nonlinear optimal controller with discrepancies in each of the model parameters. In each case, the specified model parameter is changed by a magnitude of +10% and a magnitude of -10% from the CSTR parameter value unless otherwise indicated in the figure. The sensitivity data includes controller performance criteria, average reactor temperature value, and reactor temperature standard deviation determined over a period of about 3 hours of simulation time.

Model Parameters			Simulation Results			
Changed Parameter	Parameter Value	Percent Change	SSE	SAE	Average Temperature	Temperature Std. Dev.
Uc	3,905.00	10%	325	315	88.958	0.3336
Uc	3,195.00	-10%	567	425	86.722	0.3095
Cp	199,650.00	10%	994	553	89.672	0.4966
Cp	163,350.00	-10%	885	530	86.408	0.3785
Cpc	4,602.40	10%	185	236	88.721	0.2953
Cpc	3,765.60	-10%	202	253	87.238	0.2423
V	7.788	10%	215	261	87.214	0.2443
V	6.372	-10%	297	301	88.915	0.3304
activity	1.20	20%	844	521	85.434	0.3211
activity	0.80	-20%	1,553	702	90.117	0.487
Cbi	0.12	*	952	543	89.641	0.4745
Cbi	-0.12	*	852	520	86.436	0.3613
Vc	2.002	10%	0.483	9.1	88.002	0.2113
Vc	1.638	-10%	0.504	9.2	88.001	0.2119

Figure 6.15: CSTR Model parameter sensitivity results.

Based on the sensitivity results presented in Figure 6.15, the following model parameters are selected to train the neural network to identify discrepancies between their respective process parameters:

1. $U_{cm}A_m$,
2. $c_{pm}\rho_m$,
3. $c_{pcm}\rho_{cm}$,
4. V_m,
5. α_{cat_m}, and
6. C_{bim}.

Discrepancies in the coolant jacket contents volume, V_c, do not affect the nonlinear optimal controller significantly compared to the other model parameter discrepancies.

After selecting these six model parameters to train the neural network to recognize discrepancies in, historical data is generated representing changes in each of the parameters. In each of the six cases, two datasets are generated: one representing a positive discrepancy between the process and model parameter and one representing a negative deviation between the process and model parameter. Also, a historical dataset representing normal operation is generated for a total of 13 historical datasets. The length of each simulation is about 28 hours and includes the input variable disturbances shown in Figures 6.4 through 6.7. Noise is added to all measurements sent to

the nonlinear optimal controller. The nonlinear optimal controller removes as much noise as possible using the adaptive filter algorithm.

A feedback cultivation dataset is generated for each of the 13 historical datasets. Each feedback cultivation data point consists of the nine feedback cultivation variables shown in Figure 6.14. The resulting feedback cultivation neural network training datasets consist of 125 points with nine variables each.

The structure of the neural network trained for the CSTR example consists of 9 input nodes, one for each feedback cultivation variable, and 13 output nodes, one for each class of feedback cultivation training data. Each of the two hidden layers in the network contains 50 nodes. The learning rate parameter, variable α in Equation 3.5, has a value of 0.45 during training.

Training of the neural network proceeded until 100% correct classification of the training data is achieved for most of the training datasets. Figure 6.16 shows the results of training the neural network for a CPU time of about 72 hours on a DEC VAX 6000-440. The results indicate that the network has difficulty distinguishing changes in $c_p\rho$ and, to a small degree, C_{bi}. As indicated in Figure 6.16, a significant portion of the training data representing positive and negative changes in $c_p\rho$ classified incorrectly as changes in C_{bi} with a few points getting NULL classification. This means that the feedback cultivation algorithm may modify the term C_{bim} or the bias terms when changes in the $c_p\rho$ term occur. Different choices for the feedback cultivation variables may improve the network's ability to distinguish

Training Dataset	Classification Set	Percent in Class
Uc+	Uc+	100.00%
Uc-	Uc-	100.00%
Cp+	Cp+	24.40%
	Cbi+	62.20%
	Null	13.40%
Cp-	Cp-	39.40%
	Cbi-	49.60%
	Null	11.00%
Cpc+	Cpc+	100.00%
Cpc-	Cpc-	100.00%
V+	V+	100.00%
V-	V-	100.00%
activity+	activity+	100.00%
activity-	activity-	100.00%
Cbi+	Cbi+	100.00%
Cbi-	Cbi-	94.40%
	Null	5.50%
Normal	Normal	100.00%

Figure 6.16: Neural network training results for the CSTR example.

between these parameter discrepancies and correct the problem.

Training of the network with fewer than 50 nodes in each hidden layer were attempted with little success. Using 20, 30, and 40 nodes in each hidden layer resulted in poorer performance by the network. The most significant difference between the network with 50 nodes per hidden layer and the network with fewer hidden layer nodes was in the magnitude of some of the node-to-node weighting factor values. For a network with fewer than 50 nodes per hidden layer, some of the weighting factor values were 15.0 or higher, up to a value of 30.0. These high weighting factor values caused problems for some sets of test data resulting in arithmetic overflow errors during pattern classification with the network. The network with 50 nodes per hidden layer has weighting factor values of 10.0 or less. Also, in spite of the greater number of weighting factors to determine, the training time for the network with 50 nodes per hidden layer is less than that for the training of the network with fewer hidden layer nodes.

Once trained, the neural network classifies feedback cultivation datasets during simulations of the CSTR. During each classification, 5 overlapping feedback cultivation datapoints are classified. The model re-identification algorithm interprets the results from the neural network and re-identifies specific inaccurate parameters. Model re-identification only occurs if all 5 points fall into the same problem class.

Just as in the first-order example presented in Chapter 5, three possible methods exist for determining the initial condition values of the state

variables (C_{am}, T_m, and T_{cm}) for the CSTR model:

1. the latest measurements after filtering with the adaptive filter can be used as the initial condition values for the three state variables in the CSTR model,

2. the initial conditions can be search variables along with the inaccurate parameter in the model re-identification optimization, or

3. a very large model re-identification horizon can be used to minimize the effect of inaccurate initial conditions.

The results presented in Section 6.2.1 include model re-identification results using each of these three initial condition determination methods.

6.1.2.4 Adaptive Filtering

The presence of noise in the process disturbance and process output measurements degrades the performance of the nonlinear optimal controller. The nonlinear optimal control algorithm uses the adaptive filter algorithm presented in Chapter 3 to help reduce the effect of noisy signals on the controller.

As shown in Section 3.4.2.3, the adaptive filter algorithm has two adjustable parameters: (1) n, the number of weights, and (2) μ, the LMS convergence constant. Setting these parameters is a matter of trial-and-error just as is true for setting a first-order filter constant. The advantage of the adaptive filter algorithm is that the adaptive filter output is less sensitive to

the values of these parameters. Once the filter weights have been adjusted from the initial values of 0.0, the weights do not change much. The number of weights and the LMS convergence constant merely determine the rate of filter adaptation and the degree in which the filter learns the measurement signal's dynamic characteristics.

In the CSTR simulations presented in this chapter, the four process disturbance variables of C_{ai}, T_i, T_{ci}, and F are adaptively filtered before using the signals to update the values of C_{aim}, T_{im}, T_{cim}, and F_m in the CSTR model. Also, the three process output variables of C_a, T, and T_c are adaptively filtered before saving the variables in a historical database for use with the feedback cultivation algorithm.

The adaptive filter weights for each of the seven measurement signals are uniquely determined. The filter parameters are the same for all seven measurement signals. The nonlinear optimal control algorithm uses a value of $n = 10$ and a value of $\mu = 0.05$ for all seven measurement signal filters.

Section 6.2.2 presents results comparing the performance of the nonlinear optimal control algorithm using the adaptive filter algorithm to the controller performance using a first-order filter. The first-order filter algorithm is presented in Chapter 3. The results present comparisons of the adaptive filter algorithm to first-order filters with "guessed" filter constants. Also, the results compare the adaptive filter algorithm to first-order filters with optimally tuned filter constants. This latter comparison is possible because the non-noisy measurement signals are available from the CSTR

computer simulation. This section also shows how the weights for some of the measurement signals vary during CSTR simulations.

6.2 CSTR Results

This section presents results demonstrating the complete nonlinear optimal control algorithm including feedback cultivation and adaptive filtering. Section 6.2.1 presents simulations of the CSTR with a variety of process-to-model parameter mismatches. Section 6.2.2 presents results of the comparison of the adaptive filter algorithm to a first-order filter algorithm for the CSTR example.

6.2.1 CSTR Simulations

Results for seven different simulations for each of the three methods of determining the state variable initial conditions are presented in this section. During each simulation, reactor parameters are modified one at a time or, in some cases, simultaneously. Some simulations assume that the energy term accounting for heat losses to the surroundings equals zero. Results for all three initial condition determination methods are presented together for each simulation so that comparisons can be made more easily.

The CSTR simulations use the disturbances presented in Figures 6.4 through 6.7. The measurements of the disturbance variables and the output variables are filtered using the adaptive filter algorithm described in

Section 6.1.2.4.

In all the results presented, the CSTR parameters in the equations shown in Figure 6.2 represent the changed parameters in each simulation. The model parameters shown in Figure 6.13 represent the model re-identification search parameters.

In each of the seven simulations, results for three initial condition determination methods are presented. Initial condition determination method #1 involves setting the initial values of the state variables equal to the available filtered measurements. In other words, the initial condition for C_{am} equals the latest filtered measurement of C_a adjusted by the term $bias_1$ according to Equation 6.3. Initial values for T_m and T_{cm} are determined similarly. Initial condition method #1 is possible only if measurements are available for the state variables. In the simulations presented, measurements of the three state variables are assumed available with measurement lags. Initial condition method #2 involves searching for the new values of the initial state variable values simultaneously with the inaccurate model parameter or bias terms. This is similar to the method presented by Jang [36] where the model re-identification search variables include all the model parameters and the initial condition values. Initial condition method #3 involves using a large identification horizon during the model re-identification optimization so that inaccuracies in the state variable initial values do not affect the optimization significantly.

The objective function for the model re-identification optimization

involves the square sums of the differences between the output variable measurements and the model predicted values of the output variables:

$$\Phi = \sum_{j=1}^{I}(T_j - T_{mj})^2 + \sum_{j=1}^{I}(T_{cj} - T_{cmj})^2 + \sum_{j=1}^{I}(C_{aj} - C_{amj})^2 \qquad (6.8)$$

where I is the model re-identification horizon. The horizon, I, represents the number of points from the historical database considered in the summations shown in Equation 6.8. For initial condition determination methods #1 and #2, the model re-identification horizon is $I = 50$, i.e. the latest 50 points are considered in the summations in Equation 6.8. The time represented by the 50 points is 1500.0 seconds since the points are recorded every 30 seconds which is the control action determination step size. For initial condition determination method #3, a model re-identification horizon of $I = 250$ is used which represents using historical data for the past 7500.0 seconds in the summations in Equation 6.8.

The next seven sections present the results of simulations of the CSTR using the conditions presented above and in Section 6.1.

6.2.1.1 Simulation 1

The first CSTR simulation involves changes in five of the parameters of the CSTR. At the beginning of the simulation, a -10% change is made in the UA term. At a time of 20000.0 seconds, a -10% change is made in the reactor volume, V. A 20% drop in the catalyst activity is imposed at a time of 40000.0 seconds. At a time of 60000.0 seconds, the inlet impurity concentration, C_{bi}, changes from 0.0 to 0.12 kgmoles/m^3. And finally, at a

time of 80000.0 seconds, a -10% change is made in the heat capacity, c_p. The duration of the simulation is 100000.0 seconds or approximately 28 hours.

In this simulation, the term accounting for heat losses to the surroundings is ignored. The value of constant term K_1 is set to 0.0 in the reactor jacket energy balance equation shown in Figure 6.2.

Figure 6.17 shows the variation in the reactor outlet temperature and the coolant flow rate for the simulation using initial condition method #1. Figure 6.18 shows the model re-identification results for the simulation using initial condition method #1. The variations in the plot of the reactor outlet temperature shown in Figure 6.17 correlate directly to the changes in the CSTR parameters and the subsequent model parameter corrections. Each time a CSTR parameter is changed, the reactor temperature deviates from the setpoint. After each model re-identification, the reactor temperature returns to the setpoint.

The model re-identification results shown in Figure 6.18 indicate that the feedback cultivation algorithm correctly diagnoses and corrects the change in the heat transfer coefficient U, the change in the reactor volume V, and the change in the catalyst activity.

The results in Figure 6.18 show that the feedback cultivation algorithm incorrectly treats the change in the inlet impurity concentration, C_{bi}, as an unmeasured disturbance. The data presented in Figure 6.16 indicate that some data representing changes in the inlet impurity concentration may be classified in the null class. The null class is treated as an unmeasured

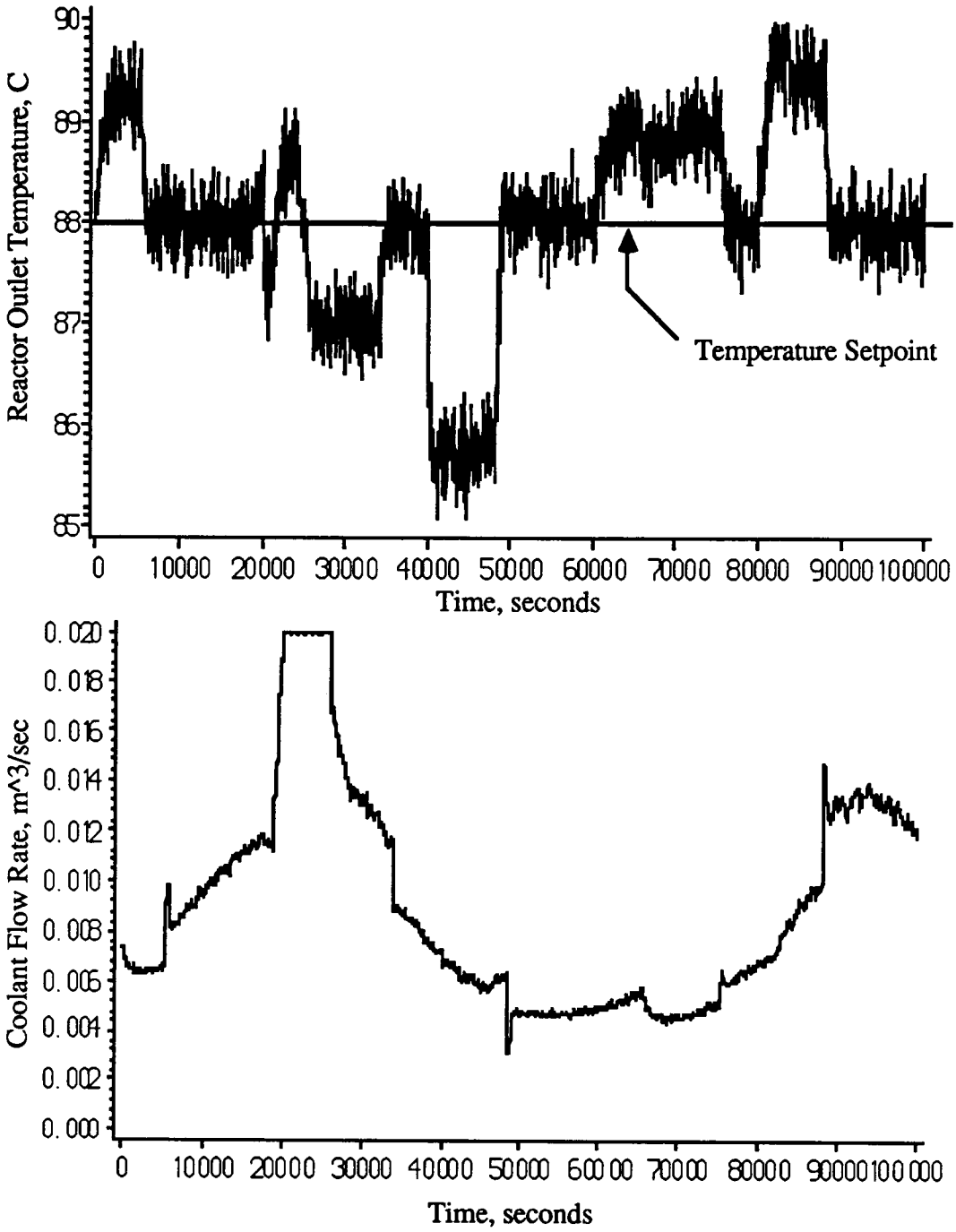


Figure 6.17: Reactor temperature and coolant flow rate for simulation 1 using initial condition method #1.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
20000.00	*V*	7.08	6.372
40000.00	*activity*	1.00	0.80
60000.00	*Cbi*	0.00	0.12
80000.00	*cp*	181500.00	163350.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*Uc*	3550.00	3187.00	3195.00	0.25%
33212.00	*V*	7.08	6.372	6.372	0.00%
48212.00	*activity*	1.00	0.7931	0.80	0.88%
69212.00	*bias1*	0.00	-0.0075	n.a.	1.39%
	bias2	0.00	0.4521	n.a.	0.29%
	bias3	0.00	0.2119	n.a.	0.05%
84962.00	*Cbi*	0.00	0.1722	0.12	43.50%

n.a. = not applicable, percent deviation based on difference between actual measurement and model initial condition plus bias term

Figure 6.18: Feedback cultivation results for using initial condition method #1 during simulation 1.

disturbance by the feedback cultivation algorithm. The feedback cultivation algorithm also incorrectly treats the change in the heat capacity of the reactor contents, c_p, as a change in the inlet impurity concentration. The data presented in Figure 6.16 indicate that this is most likely to occur.

In spite of the incorrect classifications of the change in the inlet impurity concentration and the change in the reactor contents heat capacity, the plot of the reactor outlet temperature in Figure 6.17 indicates that the model parameter changes made by the feedback cultivation algorithm return the temperature to the setpoint.

In this simulation and the six that follow, the results for the model re-identification of the bias terms due to an unmeasured disturbance show the value of the three bias terms before and after the optimization. The three bias terms correspond to the bias terms indicated in Equations 6.3 to 6.5. Also shown is the percent deviation between the bias corrected model output variable and the actual measurement of the variable:

$$\text{percent deviation} = \frac{[y - (y_m + bias)]}{y} * 100 \tag{6.9}$$

where y is the actual measurement of an output variable (C_a, T, or T_c) and y_m is the model predicted value for the same output variable. The smaller the deviation, the more closely the model output predictions match the actual output values at the current time. The accuracy of the output variables predicted into the future depends on the how accurately the model parameters match the process parameters.

Figures 6.19 and 6.20 show the results for the same simulation using initial condition determination method #2. The plot of the reactor outlet temperature in Figure 6.19 indicates that the simulation behaves in a similar manner as the simulation using initial condition method #1 until a time of about 80000.0 seconds. Another difference is the slight offset of about 0.3 ^{o}C between the setpoint and the average value of the reactor temperature after each model re-identification.

After imposing the -10% change in the reactor contents heat capacity at a time of 80000.0 seconds, the setpoint offset becomes a significant 2.0 ^{o}C. Unfortunately, too many things may be wrong at this point for the feedback cultivation algorithm to pinpoint one parameter to update. In other words, during neural network classification, all 5 of the feedback cultivation points classified never get classified as the same class of model parameter error. Changing the requirement of 100% classification or implementing more detailed analysis of the neural network output may improve the performance of the feedback cultivation algorithm in the case of multiple parameter errors.

The results shown in Figure 6.20 indicate that the feedback cultivation algorithm correctly diagnoses the change in the heat transfer coefficient, the change in reactor volume, and the change in the catalyst activity. For these three problem, the feedback cultivation algorithm behaves in the same fashion as during the simulation using initial condition method #1 but with less accuracy. The heat transfer coefficient problem is corrected for with about the same accuracy using initial condition method #2 as using initial condition

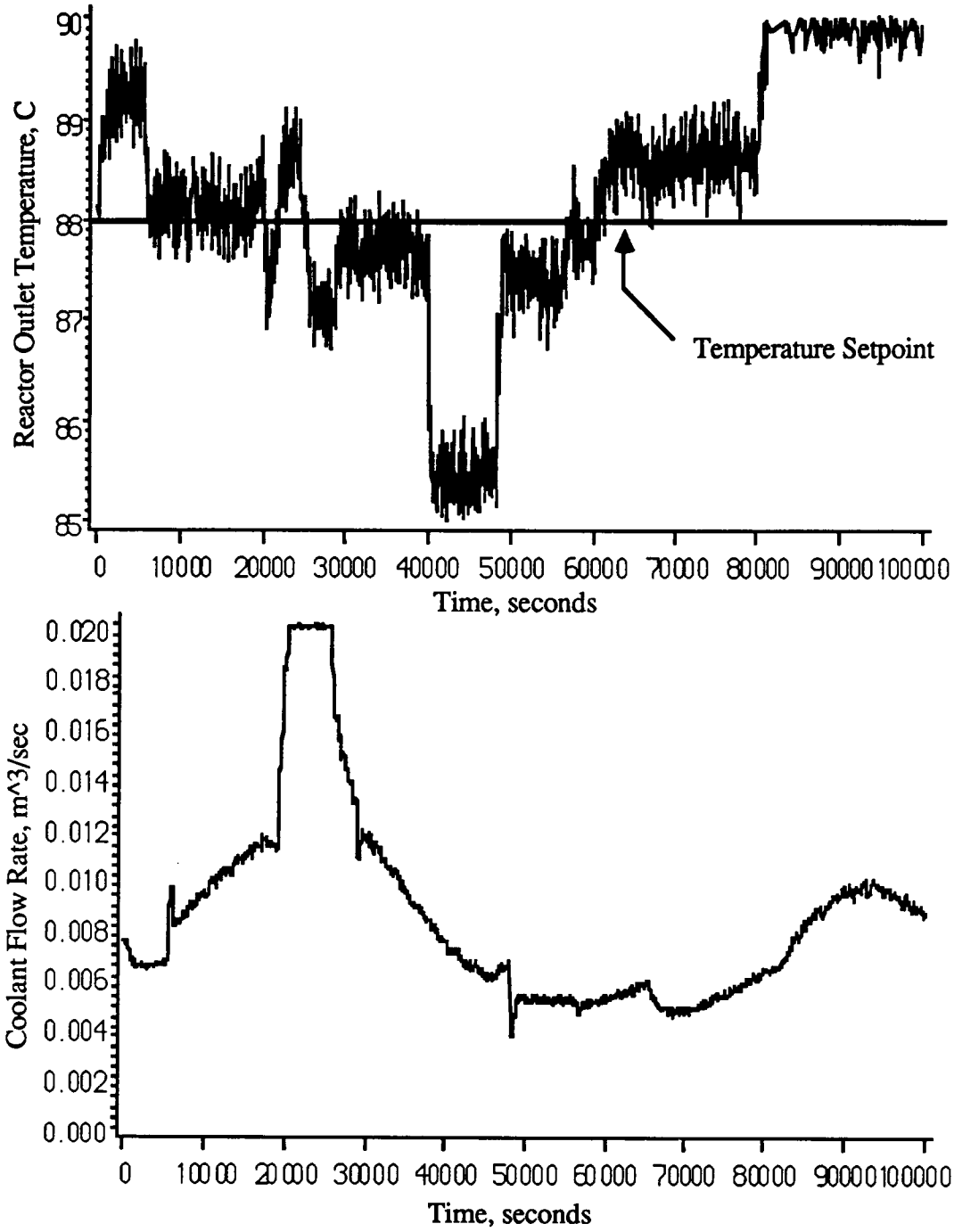


Figure 6.19: Reactor temperature and coolant flow rate for simulation 1 using initial condition method #2.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
20000.00	*V*	7.08	6.372
40000.00	*activity*	1.00	0.80
60000.00	*Cbi*	0.00	0.12
80000.00	*cp*	181500.00	163350.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*Uc*	3550.00	3211.00	3195.00	0.50%
28712.00	*V*	7.08	6.614	6.372	3.79%
48212.00	*activity*	1.00	0.82	0.80	2.52%
56462.00	*V*	6.614	6.343	6.372	0.46%

Figure 6.20: Feedback cultivation results for using initial condition method #2 during simulation 1.

method #1. The results in Figure 6.20 show that the model re-identification algorithm modifies the model reactor volume term twice before reducing the difference between the CSTR reactor volume and the model reactor volume to 0.46%. This is compared to 0.0% error using initial condition method #1. Also, the error for the catalyst activity re-identification is 2.52% for initial condition method #2 compared to 0.88% using initial condition method #1.

Figures 6.21 and 6.22 show the results for using initial condition method #3 for the same simulation. As the plot of the reactor temperature shown in Figure 6.21 indicates, this initial condition method works as well as initial condition method #1 with the exception of about 0.3 ^{0}C offset after the model re-identification at a time of 87722.0 seconds.

Examining the plot between a time of 60000.0 seconds and 80000.0 seconds indicates that using initial condition method #3 results in slightly better performance than using initial condition method #1. The reactor temperature returns more quickly to setpoint after the change in the inlet impurity concentration is detected.

The feedback cultivation results shown in Figure 6.22 show that changes in the inlet impurity concentration are diagnosed as an unmeasured disturbance. Also, the change in the reactor contents heat capacity is treated as a change in the inlet impurity concentration. These are the same results achieved using initial condition method #1. The difference between using initial condition method #3 and initial condition method #1 is the model re-identification accuracy of the model parameters. Using method #3 results

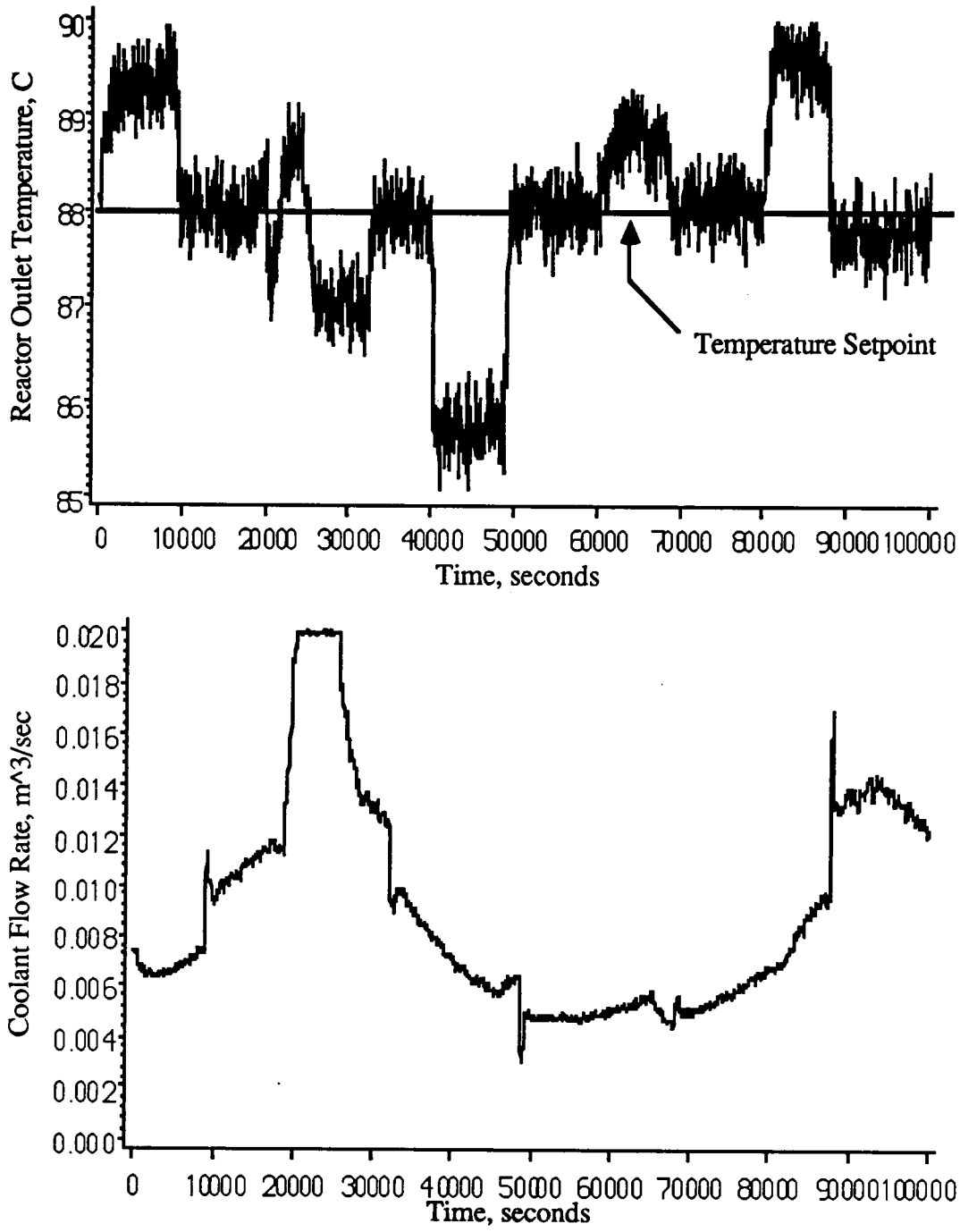


Figure 6.21: Reactor temperature and coolant flow rate for simulation 1 using initial condition method #3.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
20000.00	*V*	7.08	6.372
40000.00	*activity*	1.00	0.80
60000.00	*Cbi*	0.00	0.12
80000.00	*cp*	181500.00	163350.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
8972.00	*Uc*	3550.00	3189.00	3195.00	0.19%
32222.00	*V*	7.08	6.372	6.372	0.00%
48722.00	*activity*	1.00	0.7971	0.80	0.36%
68222.00	*bias1*	0.00	-0.0033	n.a.	0.16%
	bias2	0.00	0.7467	n.a.	0.01%
	bias3	0.00	0.3479	n.a.	0.29%
87722.00	*Cbi*	0.00	0.1489	0.12	24.10%

n.a. = not applicable, percent deviation based on difference between actual measurement and calculated model initial condition plus bias term

Figure 6.22: Feedback cultivation results for using initial condition method #3 during simulation 1.

in more accurate values for the model heat transfer coefficient, the catalyst activity, two of the three bias terms, and the inlet impurity concentration compared to method #1. The same accuracy is achieved for the re-identification of the model reactor volume.

For this simulation, using initial condition method #3 works the best of the three initial condition methods. Initial condition method #1 also works well. Initial condition method #2 has difficulty from the beginning and bogs down completely after imposing the change in the heat capacity of the reactor contents.

6.2.1.2 Simulation 2

Simulation 2 is exactly the same as simulation 1 except that the energy changes due to heat losses to the surroundings are accounted for in the reactor jacket energy balance equation shown in Figure 6.2. In this simulation, the constant term K_1 has a value of 3000.0 selected arbitrarily. During the simulation, the value of the ambient temperature, T_{amb} remains constant at 27.0 ^{0}C. But the term accounting for the heat losses to the surroundings varies during the simulation because of the variation of the inlet coolant temperature (shown in Figure 6.6) and the variation of the coolant temperature in the reactor jacket.

Figures 6.23 and 6.24 show the results for using initial condition method #1 during the simulation. Figures 6.25 and 6.26 show the results for using initial condition method #2 during the simulation. And Figures 6.27

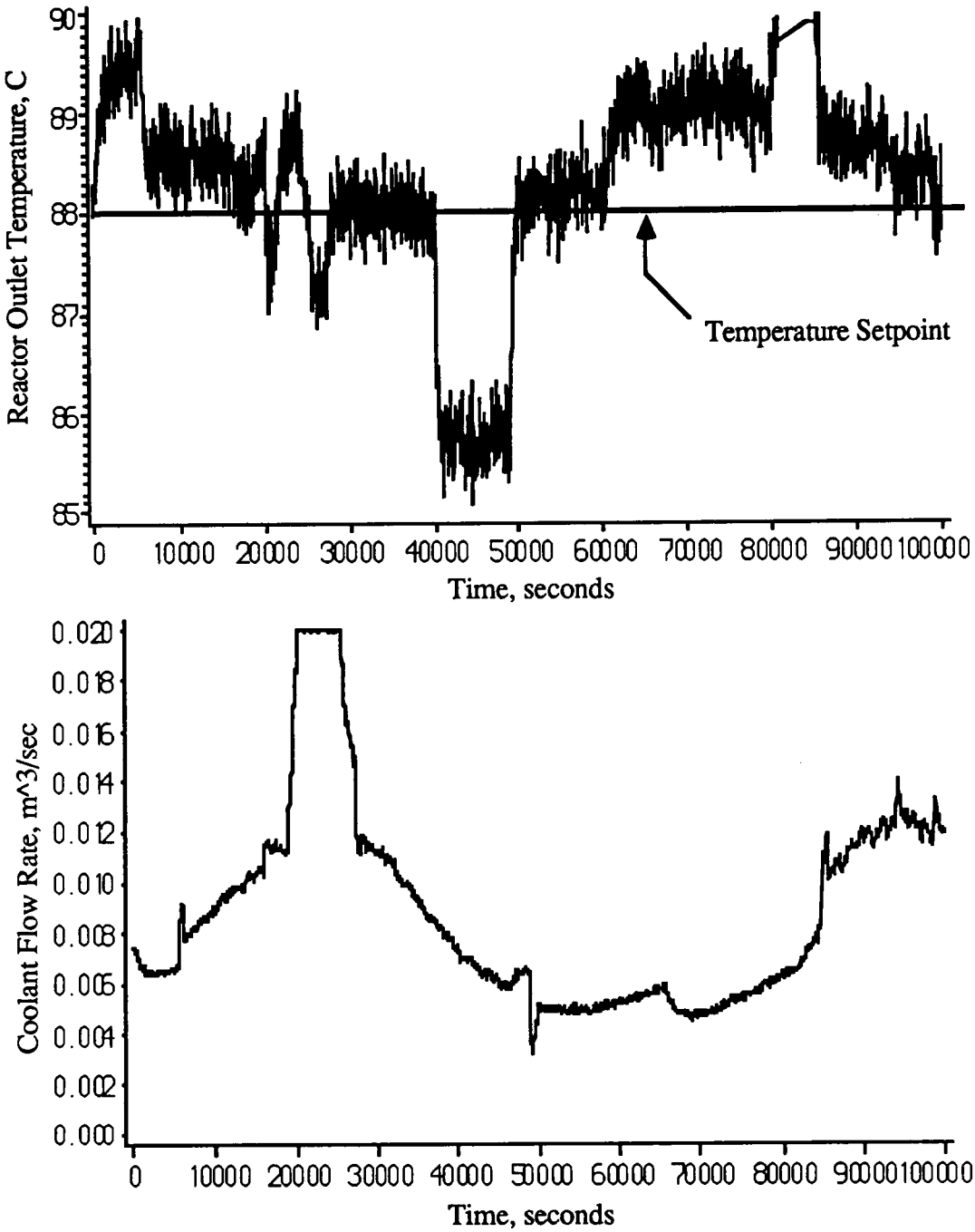


Figure 6.23: Reactor temperature and coolant flow rate for simulation 2 using initial condition method #1.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
20000.00	*V*	7.08	6.372
40000.00	*activity*	1.00	0.80
60000.00	*Cbi*	0.00	0.12
80000.00	*cp*	181500.00	163350.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*Uc*	3550.00	3263.00	3195.00	2.10%
15962.00	*bias1*	0.00	-0.0012	n.a.	0.07%
	bias2	0.00	0.2915	n.a.	0.15%
	bias3	0.00	0.2379	n.a.	0.43%
27212.00	*V*	7.08	6.484	6.372	1.76%
48962.00	*activity*	1.00	0.8025	0.80	0.31%
84962.00	*bias1*	-0.0012	-0.0017	n.a.	0.02%
	bias2	0.2915	1.613	n.a.	0.72%
	bias3	0.2379	0.6863	n.a.	0.28%
93962.00	*bias1*	-0.0017	0.0032	n.a.	0.80%
	bias2	1.613	1.929	n.a.	0.24%
	bias3	0.6863	0.4673	n.a.	0.42%
98462.00	*bias1*	0.0032	0.0007	n.a.	0.85%
	bias2	1.929	2.289	n.a.	0.13%
	bias3	0.4673	0.6508	n.a.	0.15%

n.a. = not applicable, percent deviation based on difference between actual measurement and model initial condition plus bias term

Figure 6.24: Feedback cultivation results for using initial condition method #1 during simulation 2.

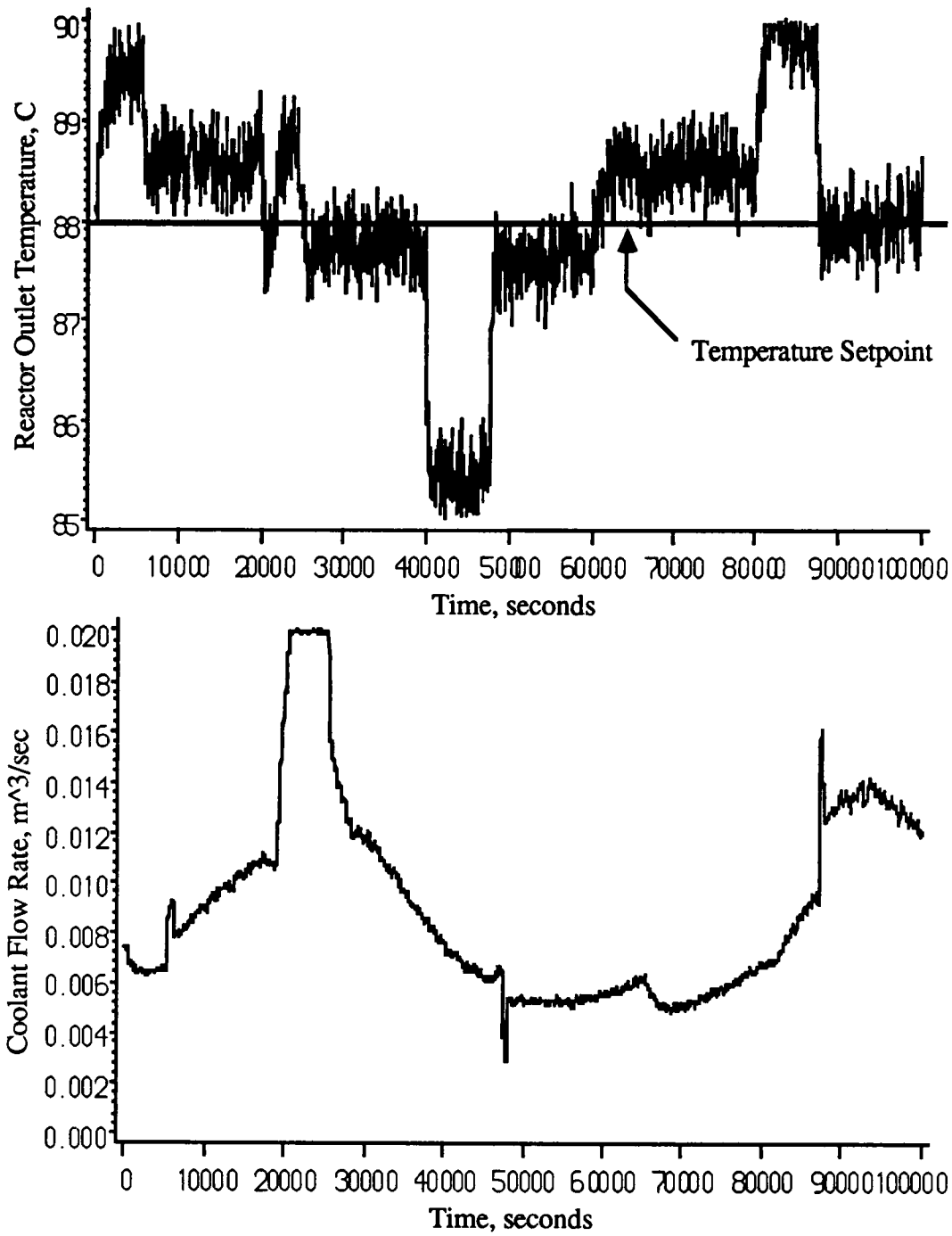


Figure 6.25: Reactor temperature and coolant flow rate for simulation 2 using initial condition method #2.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
20000.00	*V*	7.08	6.372
40000.00	*activity*	1.00	0.80
60000.00	*Cbi*	0.00	0.12
80000.00	*cp*	181500.00	163350.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*Uc*	3550.00	3264.00	3195.00	2.16%
25712.00	*V*	7.08	6.90	6.372	8.29%
34712.00	*bias1*	0.00	0.0428	n.a.	0.41%
	bias2	0.00	-0.0046	n.a.	0.17%
	bias3	0.00	0.0071	n.a.	0.13%
47462.00	*activity*	1.00	0.8183	0.80	2.29%
55712.00	*bias1*	0.0428	0.0515	n.a.	0.52%
	bias2	-0.0046	-0.0126	n.a.	0.31%
	bias3	0.0071	0.0214	n.a.	0.73%
60212.00	*bias1*	0.0515	0.0512	n.a.	0.64%
	bias2	-0.0126	-0.0195	n.a.	0.18%
	bias3	0.0214	0.0444	n.a.	0.80%
87212.00	*Cbi*	0.00	0.1404	0.12	16.67%

n.a. = not applicable, percent deviation based on difference between actual measurement and model initial condition plus bias term

Figure 6.26: Feedback cultivation results for using initial condition method #2 during simulation 2.

and 6.28 show the results for using initial condition method #3 during the simulation.

The model re-identification results for all three initial condition determination methods indicate that the bias terms are re-identified more frequently than during simulation 1. This shows that more unmeasured disturbances are detected by the feedback cultivation algorithm. Since the term accounting for heat losses to the surroundings varies during the simulation, more unmeasured disturbances occur during the simulation.

The plots of the reactor outlet temperature for all three initial condition methods indicate that the feedback cultivation algorithm keeps the temperature close to the setpoint after model re-identification. Even using initial condition method #2 keeps the temperature at the setpoint after the change in the heat capacity of the reactor contents at a time of 80000.0 seconds. This is compared to the failure of the feedback cultivation algorithm to correct for this change in simulation 1 using initial condition method #2.

The results in shown in Figure 6.24 show that using initial condition method #1 corrects for the changes in the heat transfer coefficient, changes in the reactor volume, and the catalyst activity in the same fashion as during simulation 1 but with less accuracy. The heat transfer coefficient in the model varies by 2.1% from the actual value in the CSTR for simulation 2 compared to 0.25% in simulation 1. The model reactor volume varies the actual CSTR volume by 1.76% during simulation 2 compared to 0.0% during simulation 1. However, the model value of the catalyst activity varies by

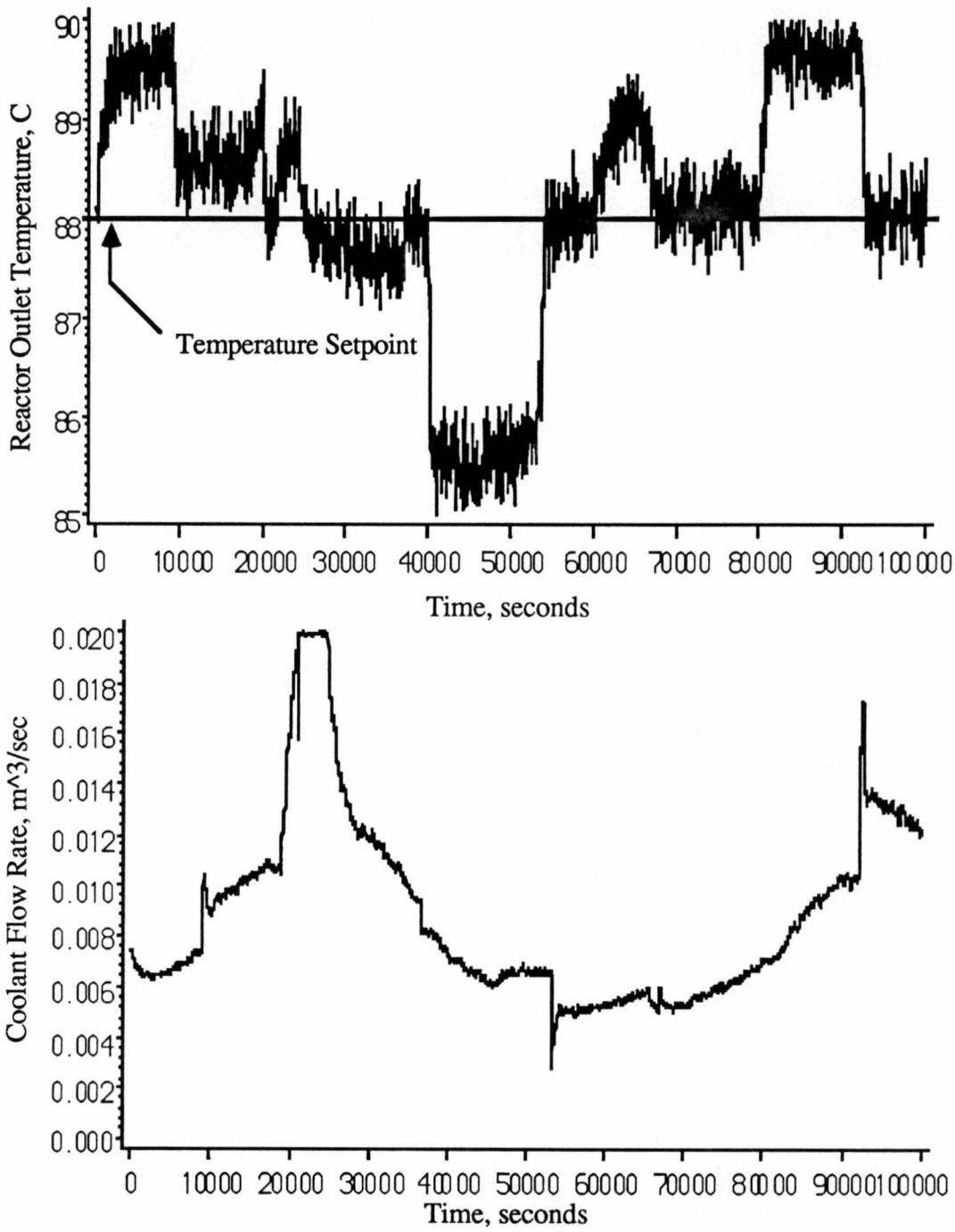


Figure 6.27: Reactor temperature and coolant flow rate for simulation 2 using initial condition method #3.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
20000.00	*V*	7.08	6.372
40000.00	*activity*	1.00	0.80
60000.00	*Cbi*	0.00	0.12
80000.00	*cp*	181500.00	163350.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
8972.00	*Cbi*	0.00	0.0847	0.00	>100%
36722.00	*V*	7.08	6.704	6.372	5.21%
53222.00	*bias1*	0.00	0.1351	n.a.	1.19%
	bias2	0.00	-2.382	n.a.	0.03%
	bias3	0.00	-1.973	n.a.	0.16%
66722.00	*bias1*	0.1351	0.1297	n.a.	0.18%
	bias2	-2.382	-1.637	n.a.	0.17%
	bias3	-1.973	-1.401	n.a.	0.32%
71222.00	*bias1*	0.1297	0.1292	n.a.	0.25%
	bias2	-1.637	-1.473	n.a.	0.01%
	bias3	-1.401	-1.277	n.a.	0.28%
75722.00	*bias1*	0.1292	0.128	n.a.	0.32%
	bias2	-1.473	-1.494	n.a.	0.07%
	bias3	-1.277	-1.291	n.a.	0.47%
80222.00	*bias1*	0.128	0.1291	n.a.	0.64%
	bias2	-1.494	-1.416	n.a.	0.47%
	bias3	-1.291	-1.331	n.a.	0.21%
92222.00	*bias1*	0.1291	0.1199	n.a.	0.15%
	bias2	-1.416	0.241	n.a.	0.10%
	bias3	-1.331	-1.073	n.a.	0.09%
96722.00	*bias1*	0.1199	0.1207	n.a.	0.55%
	bias2	0.241	0.2576	n.a.	0.04%
	bias3	-1.073	-0.9674	n.a.	0.65%

n.a. = not applicable, percent deviation based on difference between actual measurement and calculated model initial condition plus bias term

Figure 6.28: Feedback cultivation results for using initial condition method #3 during simulation 2.

0.31% during simulation 2 which is slightly more accurate than the variation of 0.88% that occurred during simulation 1.

After the change in the inlet impurity concentration at a time of 60000.0 seconds, the feedback cultivation algorithm fails to diagnose the change in the inlet impurity concentration or the subsequent change in the reactor contents heat capacity at a time of 80000.0 seconds as indicated by the deviation from setpoint by the reactor temperature shown in Figure 6.23. After modifying the bias terms three times, the reactor temperature appears to be returning to the setpoint at the end of the simulation.

The results shown in Figure 6.26 show that the feedback cultivation algorithm works in a similar fashion as in simulation 1. The main difference is that the feedback cultivation algorithm recognizes the change in the reactor heat capacity as a change in the inlet impurity concentration. This relates to the results for simulation 1 using initial condition method #1 shown in Figure 6.18.

Although the plot of the reactor temperature shown in Figure 6.27 is similar to those shown in Figures 6.23 and 6.25, the feedback cultivation results shown in Figure 6.28 indicate that the feedback cultivation algorithm using initial condition method #3 deals with the process parameter changes in a completely different manner. The only similarity is the modification of the model value of the reactor volume. But even this model parameter re-identification occurs much later in the simulation than for using method #1 or #2. Also, the resulting value of the model reactor volume varies from

the CSTR volume by 5.21% which is much larger than the variation during the same simulation using method #1 or #2.

In this simulation, it is not clear which initial condition method is superior. By examining the three plots of the reactor outlet temperature, using initial condition method #3 appears slightly superior to the other two methods in terms of returning the reactor temperature to the setpoint without very little offset after model re-identification. The feedback cultivation results indicate that using initial condition method #2 is slightly superior to the other two methods followed closely by using initial condition method #1. The feedback cultivation results using initial condition method #3 deviate substantially from the results using the other two methods during this simulation and the results using all three methods during simulation 1.

6.2.1.3 Simulation 3

Simulation 3 consists of imposing both a -10% change in the CSTR heat transfer coefficient and a -10% change in the reactor contents volume at the beginning of the simulation. During this simulation, the energy term accounting for heat losses to the surroundings is ignored. This simulation tests the feedback cultivation algorithm's ability to deal with multiple problems occuring simultaneously.

Figures 6.29 through 6.34 show the reactor temperature and the feedback cultivation results for simulation 3 using each of the three initial condition methods. The plots of the reactor temperature shown in

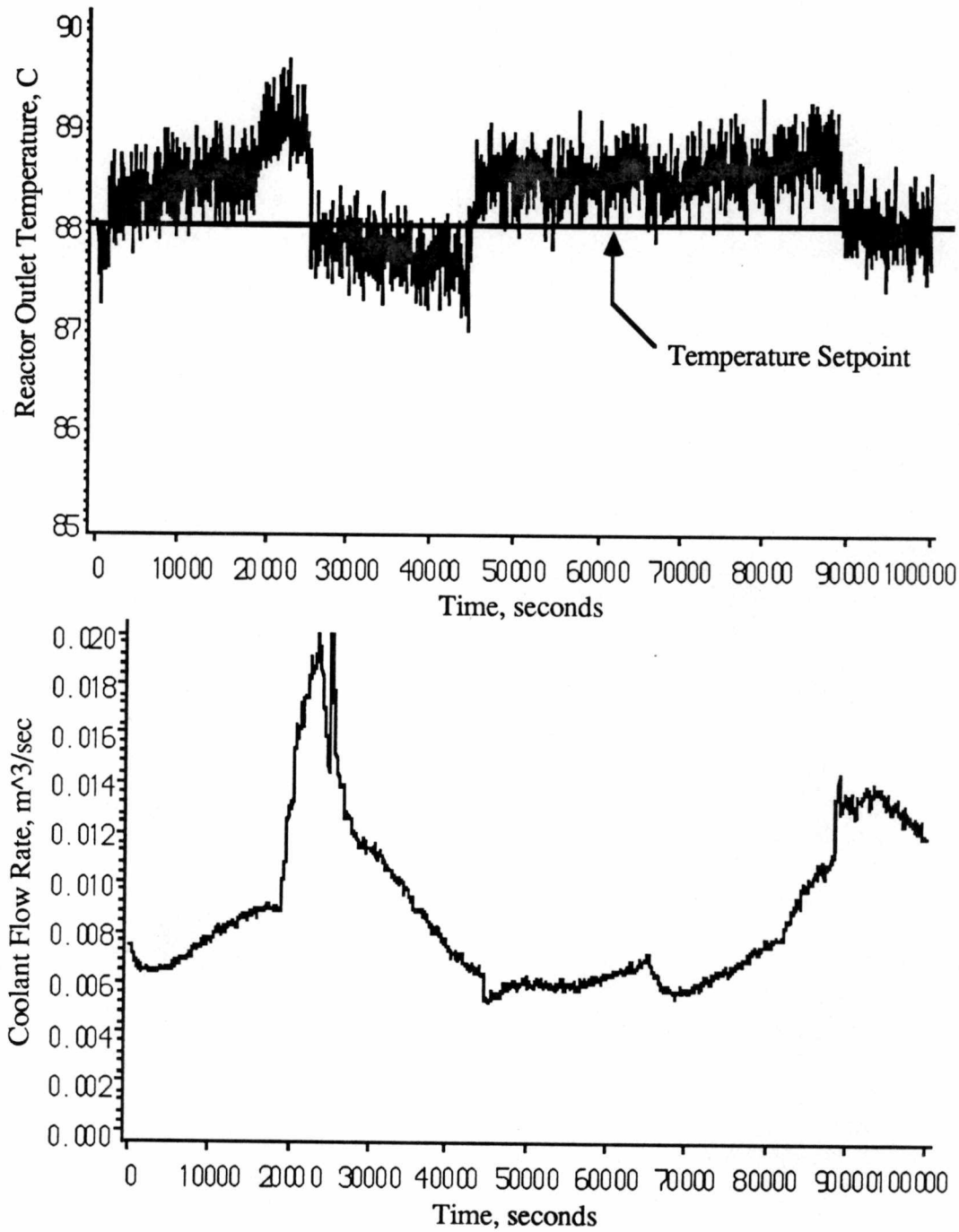


Figure 6.29: Reactor temperature and coolant flow rate for simulation 3 using initial condition method #1.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
	V	7.08	6.372

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
24962.00	*Uc*	3550.00	3359.00	3195.00	5.13%
44462.00	*V*	7.08	6.458	6.372	1.35%
88712.00	*Uc*	3359.00	3213.00	3195.00	0.56%

Figure 6.30: Feedback cultivation results for using initial condition method #1 during simulation 3.

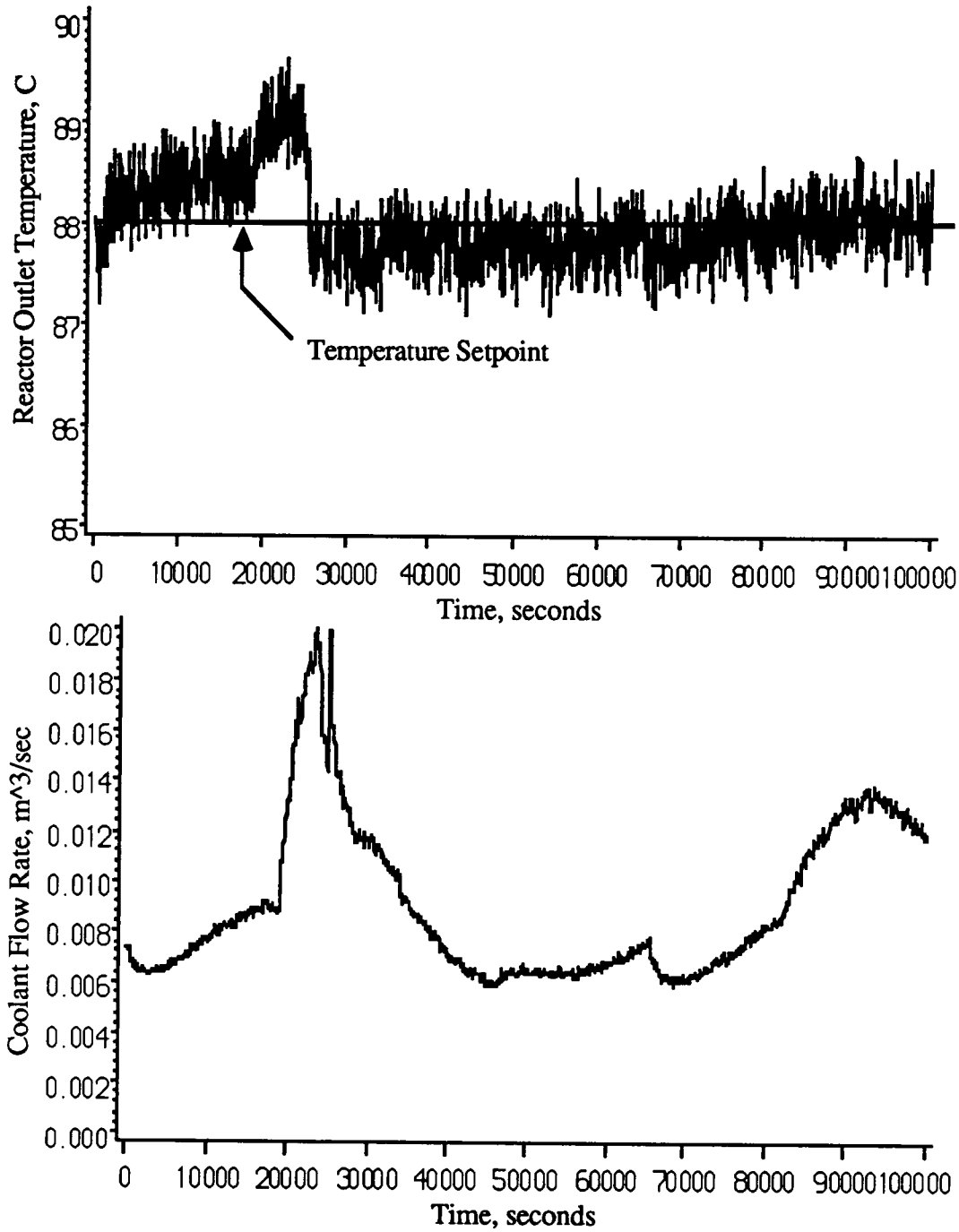


Figure 6.31: Reactor temperature and coolant flow rate for simulation 3 using initial condition method #2.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
	V	7.08	6.372

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
24962.00	*Uc*	3550.00	3333.00	3195.00	4.32%
33962.00	*V*	7.08	6.877	6.372	7.93%

Figure 6.32: Feedback cultivation results for using initial condition method #2 during simulation 3.

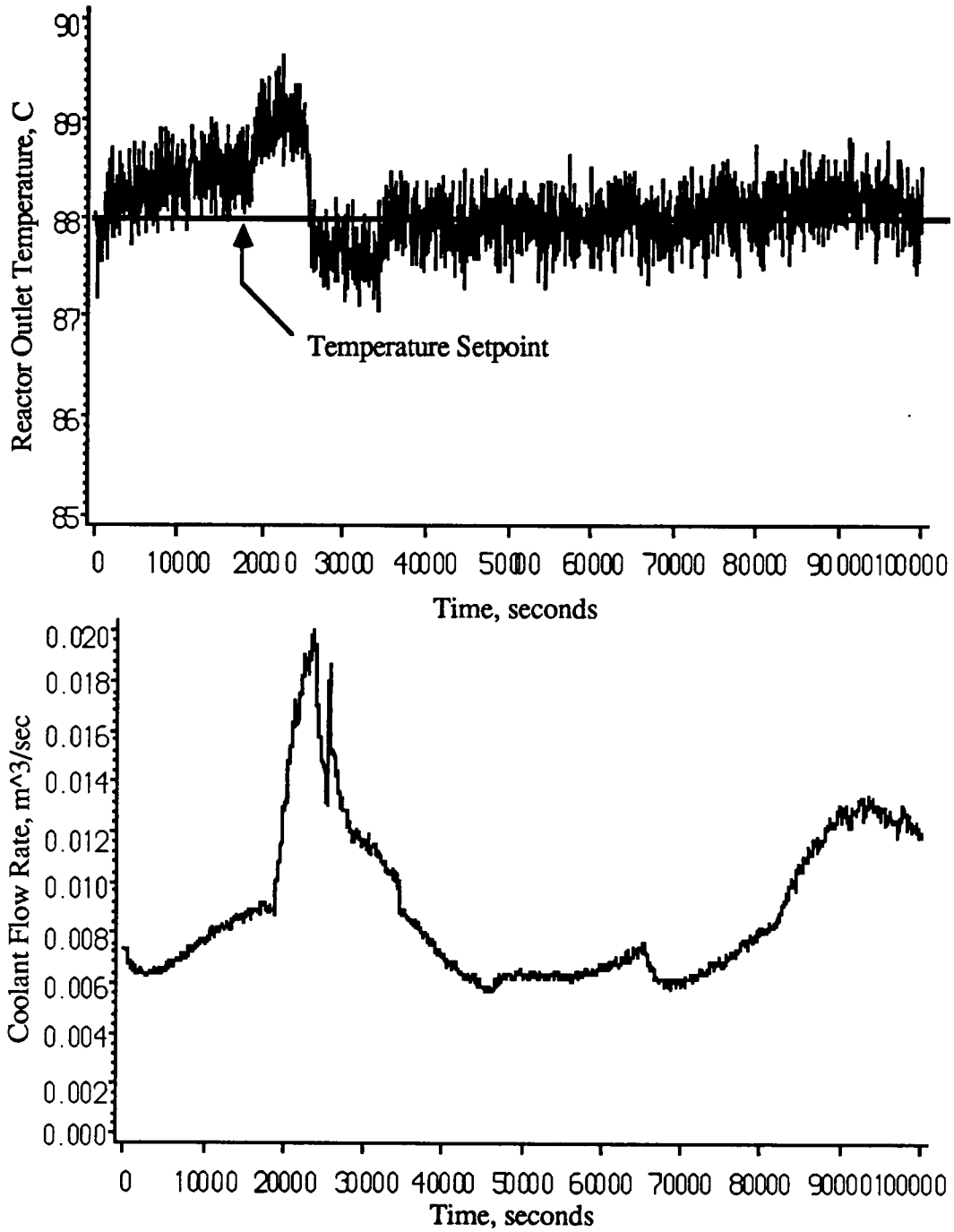


Figure 6.33: Reactor temperature and coolant flow rate for simulation 3 using initial condition method #3.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
	V	7.08	6.372

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
25472.00	*Uc*	3550.00	3325.00	3195.00	4.07%
34472.00	*V*	7.08	6.735	6.372	5.70%
97472.00	*bias1*	0.00	0.0236	n.a.	0.05%
	bias2	0.00	0.1761	n.a.	0.15%
	bias3	0.00	-0.3952	n.a.	0.06%

n.a. = not applicable, percent deviation based on difference between actual measurement and calculated model initial condition plus bias term

Figure 6.34: Feedback cultivation results for using initial condition method #3 during simulation 3.

Figures 6.29, 6.31, and 6.33 indicate that the feedback cultivation algorithm returns the temperature to the setpoint value during the simulation. Using initial condition method #1 results in offset from the setpoint until near the end of the simulation. Using both methods #2 and #3 returns the temperature to the setpoint value between 30000.0 and 40000.0 seconds into the simulation.

The feedback cultivation results shown in Figures 6.30, 6.32, and 6.34 indicate that the feedback cultivation algorithm correctly determines that the model heat transfer coefficient and the model reactor volume need modification. In all three cases, the heat transfer coefficient is modified first at a time of around 25000.0 seconds. Also in all three cases, the model reactor volume term is modified at a time of 44000.0 seconds using initial condition method #1 and at a time of around 34000.0 seconds using initial condition methods #2 and #3. Near the end of the simulation using initial condition method #1, the heat transfer coefficient is re-identified again. Also near the end of the simulation, the feedback cultivation algorithm using initial condition method #3 modifies the bias terms.

In spite of the apparent deviation of the reactor temperature from setpoint during most of the simulation, the feedback cultivation algorithm using initial condition method #1 determines the most accurate values for the model heat transfer coefficient and the model reactor volume term. After the second re-identification of the model heat transfer coefficient, the deviation from the CSTR heat transfer coefficient is 0.56%. The deviation of the model

reactor volume from the CSTR volume is 1.35% after model re-identification.

The deviations from the CSTR parameter values of the model heat transfer coefficient and the model reactor volume using method #2 are 4.32% and 7.93% respectively. The discrepancies from the correct values seem to offset each other as indicated by the return of the reactor temperature to setpoint shown in Figure 6.31.

The model parameters after re-identification, are slightly more accurate using initial condition method #3 than using method #2 as shown in Figure 6.34. The discrepancies from the correct values offset each other until near the end of the simulation where the feedback cultivation algorithm modifies the bias terms to bring the reactor temperature back to setpoint.

Again, for this simulation as for simulation 2, the results do not indicate a superior method for determing the initial conditions in the feedback cultivation algorithm. Examining the plots of the reactor temperature shows that initial condition methods #2 and #3 result in a similar response to the changes in the CSTR parameters and that both methods work better than initial condition method #1. The feedback cultivation results indicate that using initial condition method #1 determines model parameters more accurate than the parameters determined by the feedback cultivation algorithm using methods #2 and #3.

6.2.1.4 Simulation 4

Simulation 4 is exactly the same as simulation 3 except that heat losses to the surroundings are not ignored. Figures 6.35 through 6.40 show the results for using each of the three initial condition methods in the feedback cultivation algorithm during the simulation.

The plots of the reactor temperature shown in Figures 6.35, 6.37, and 6.39 indicate that the temperature returns to setpoint with little offset after a time of about 25000.0 seconds. For the case of using initial condition method #3, the reactor temperature has an offset of about 0.4 ^{0}C as indicated in Figure 6.39.

The feedback cultivation results shown in Figures 6.36, 6.38, and 6.40 indicate the feedback cultivation does a poor job of re-identifying the model parameters. During all three simulations, no unmeasured disturbances are diagnosed in spite of the presence of unmeasured disturbances due to accounting for heat losses to the surroundings in the CSTR equations.

The feedback cultivation algorithm does correctly diagnose which model parameters need modification using initial condition methods #2 and #3. Figures 6.38 and 6.40 indicate that first the model heat transfer coefficient is modified at a time of about 23000.0 seconds. The model value of the reactor volume is modified next. In the case of using initial condition method #2, the model value of the reactor volume is modified twice. Figure 6.36 shows that only the model heat transfer coefficient is modified during the simulation using initial condition method #1, yet the reactor

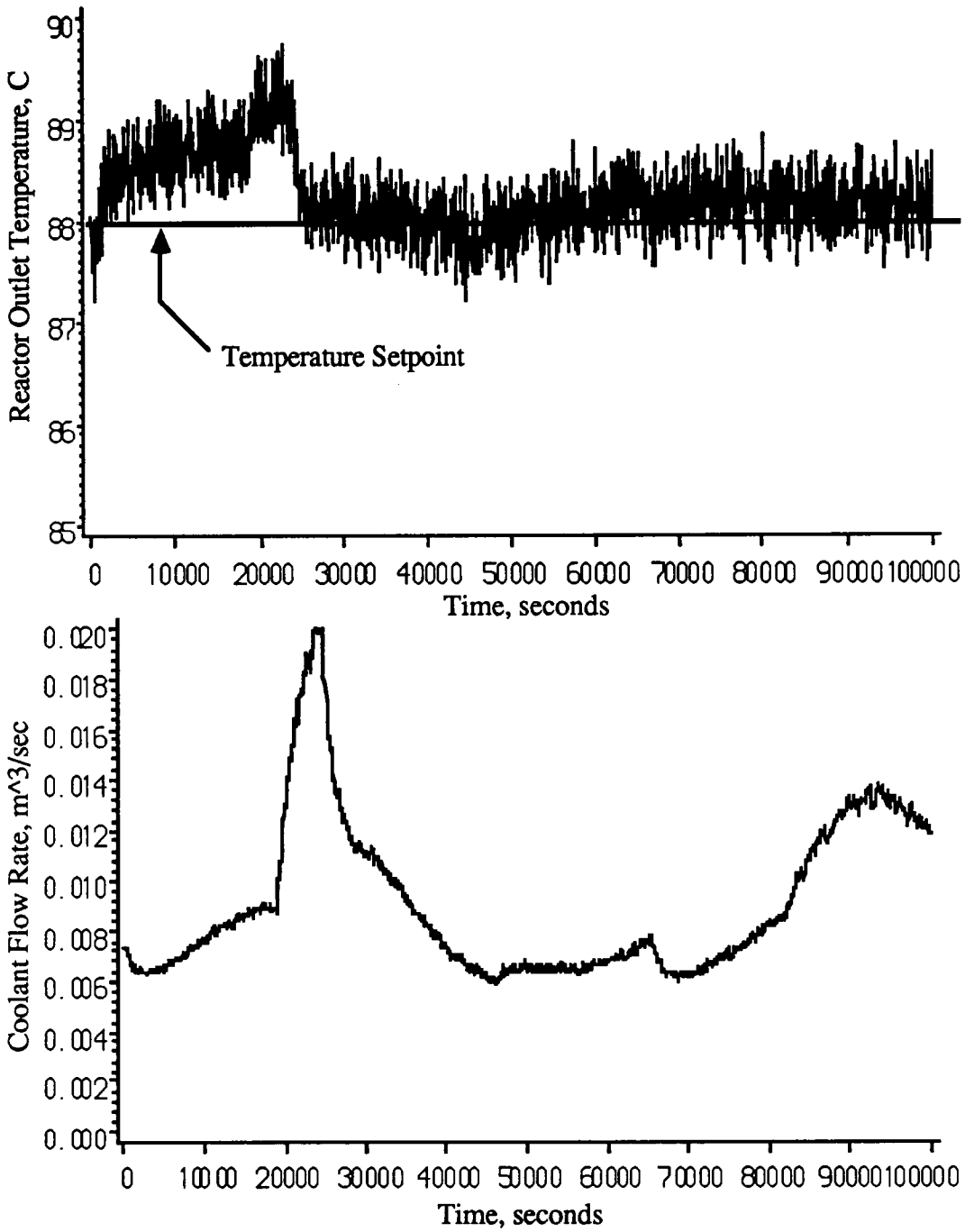


Figure 6.35: Reactor temperature and coolant flow rate for simulation 4 using initial condition method #1.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
	V	7.08	6.372

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
23462.00	*Uc*	3550.00	3384.00	3195.00	5.91%

Figure 6.36: Feedback cultivation results for using initial condition method #1 during simulation 4.

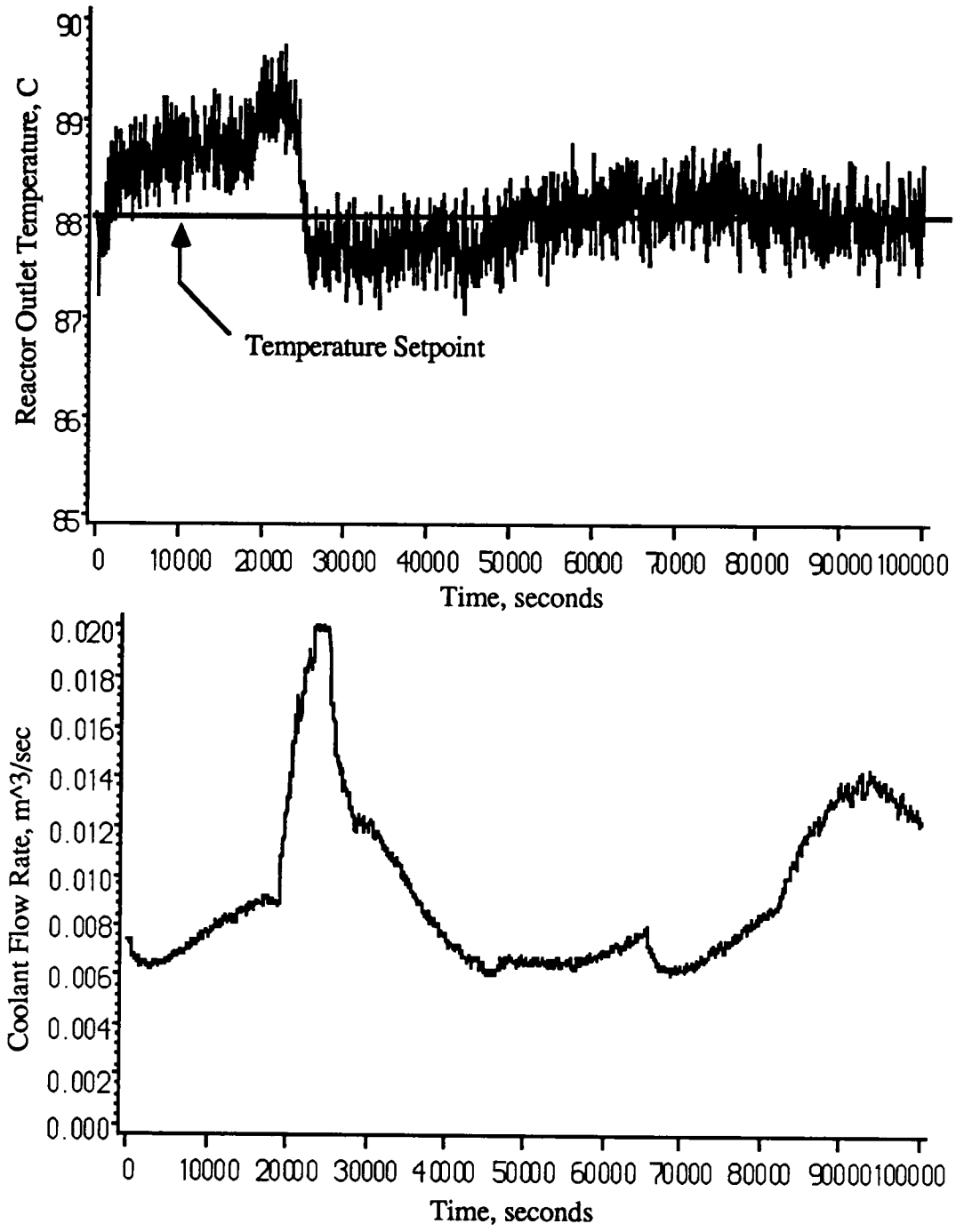


Figure 6.37: Reactor temperature and coolant flow rate for simulation 4 using initial condition method #2.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
	V	7.08	6.372

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
23462.00	*Uc*	3550.00	3292.00	3195.00	3.03%
36212.00	*V*	7.08	6.956	6.372	9.16%
48962.00	*V*	6.956	6.85	6.372	7.50%

Figure 6.38: Feedback cultivation results for using initial condition method #2 during simulation 4.

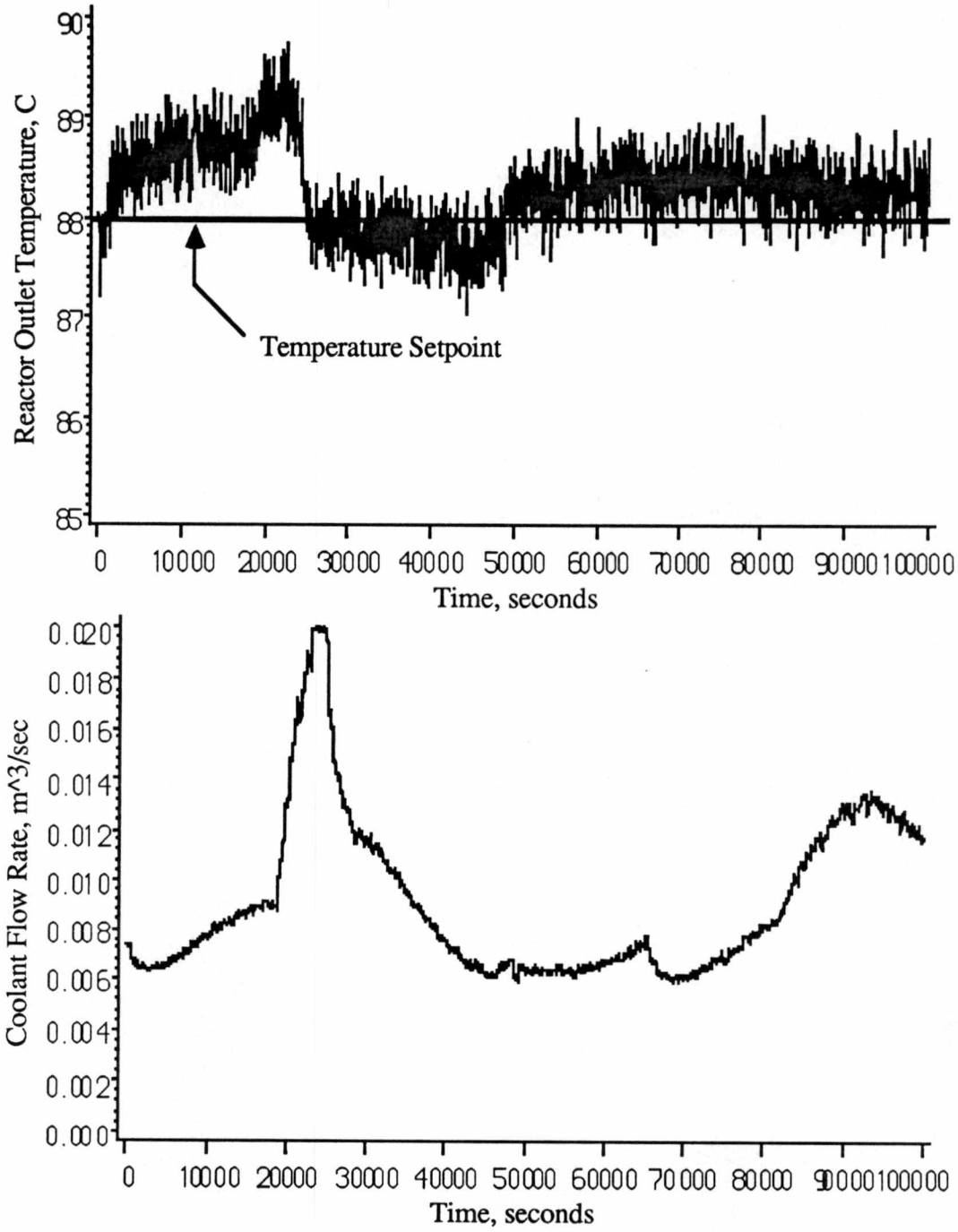


Figure 6.39: Reactor temperature and coolant flow rate for simulation 4 using initial condition method #3.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Uc*	3550.00	3195.00
	V	7.08	6.372

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
23222.00	*Uc*	3550.00	3335.00	3195.00	4.38%
48722.00	*V*	7.08	6.803	6.372	6.76%

Figure 6.40: Feedback cultivation results for using initial condition method #3 during simulation 4.

temperature shown in Figure 6.35 seems to track the setpoint the best after the one model re-identification compared to the cases using initial condition methods #2 and #3.

6.2.1.5 Simulation 5

In simulation 5, the only change made is a 5.0 ^{0}C increase in the ambient temperature, T_{amb}. The values for all comparable model parameters equal the values of the CSTR parameters.

Figures 6.41 through 6.46 show the results for using each of the three initial condition methods during the simulation. Note that results for using initial condition method #1 shown in Figures 6.41 and 6.42 stop after a simulation time of 50000.0 seconds instead of continuing to 100000.0 seconds as the other two cases do.

The feedback cultivation results shown in Figures 6.42, 6.44, and 6.46 indicate that the deviations of the reactor temperature from setpoint are diagnosed correctly as unmeasured disturbances. The one exception appears in Figure 6.44 at a time of 73000.0 seconds where the model value of the coolant heat capacity, c_{pcm} is modified by the feedback cultivation algorithm using initial condition method #2.

The reactor temperature in Figures 6.41, 6.43, and 6.45 tends to drift away from the setpoint until modification of the bias terms by the feedback cultivation algorithm brings it back to setpoint. This drift is due to the variation of the heat loss term in the reactor jacket energy balance equation

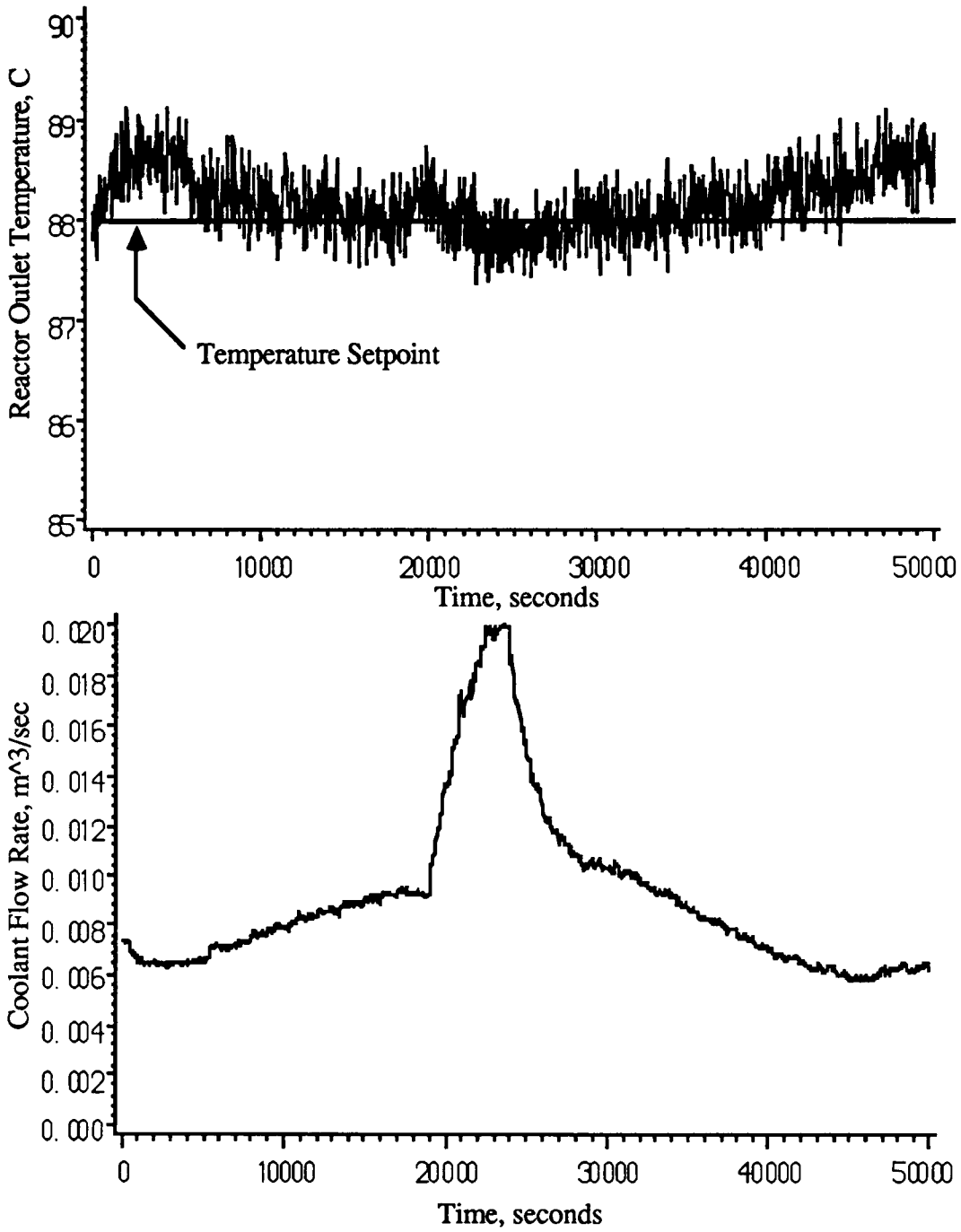


Figure 6.41: Reactor temperature and coolant flow rate for simulation 5 using initial condition method #1.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	32.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*bias1*	0.00	0.001	n.a.	0.18%
	bias2	0.00	0.2346	n.a.	0.18%
	bias3	0.00	0.8633	n.a.	0.23%

n.a. = not applicable, percent deviation based on difference between actual ment and model initial condition plus bias term

Figure 6.42: Feedback cultivation results for using initial condition method #1 during simulation 5.

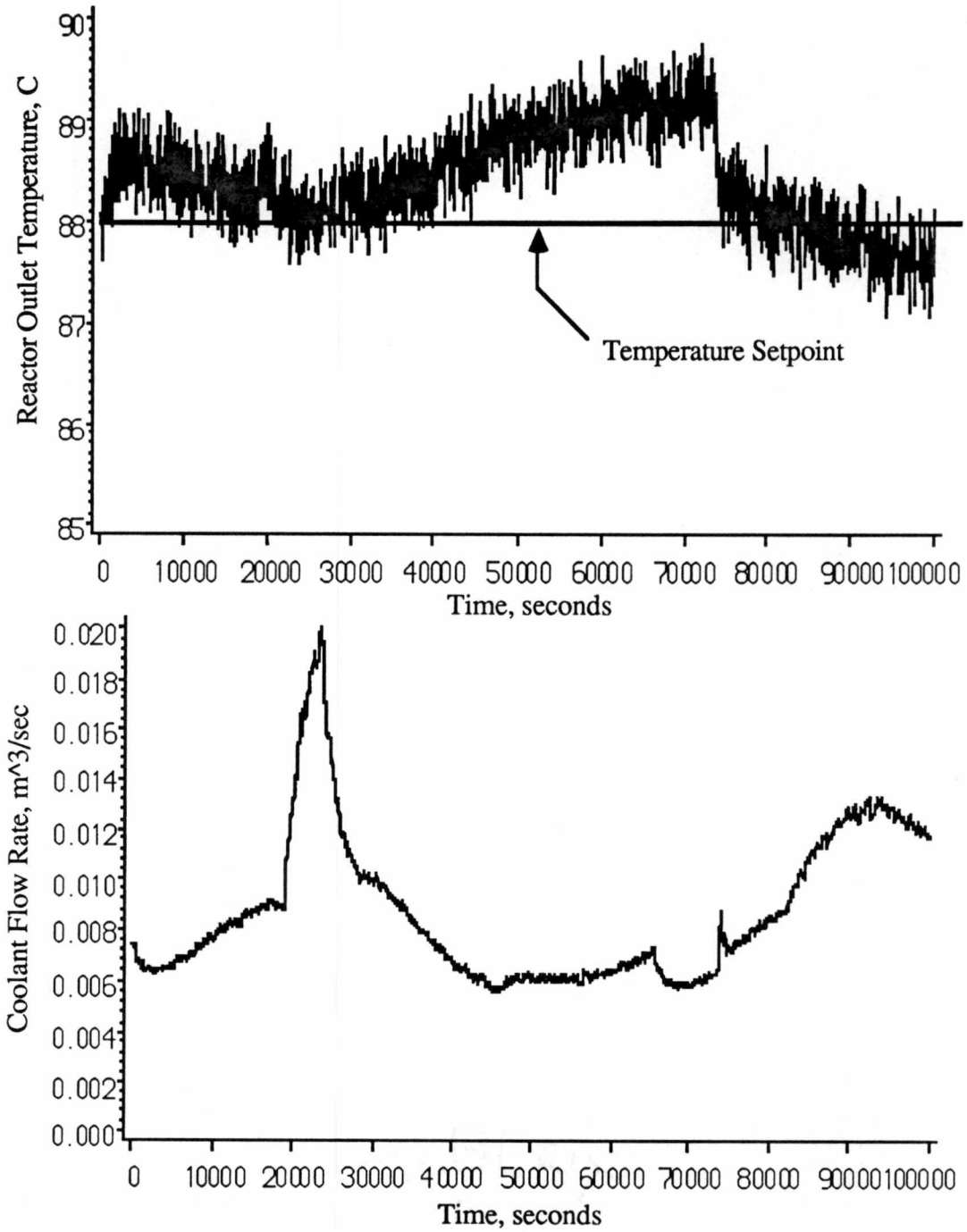


Figure 6.43: Reactor temperature and coolant flow rate for simulation 5 using initial condition method #2.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	32.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*bias1*	0.00	0.0086	n.a.	0.41%
	bias2	0.00	0.0091	n.a.	0.32%
	bias3	0.00	0.0467	n.a.	1.64%
56462.00	*bias1*	0.0086	-0.0352	n.a.	3.80%
	bias2	0.0091	0.0227	n.a.	0.66%
	bias3	0.0467	0.1345	n.a.	4.25%
73712.00	*cpc*	4184.00	3786.00	4184.00	9.51%

n.a. = not applicable, percent deviation based on difference between actual measurement and model initial condition plus bias term

Figure 6.44: Feedback cultivation results for using initial condition method #2 during simulation 5.

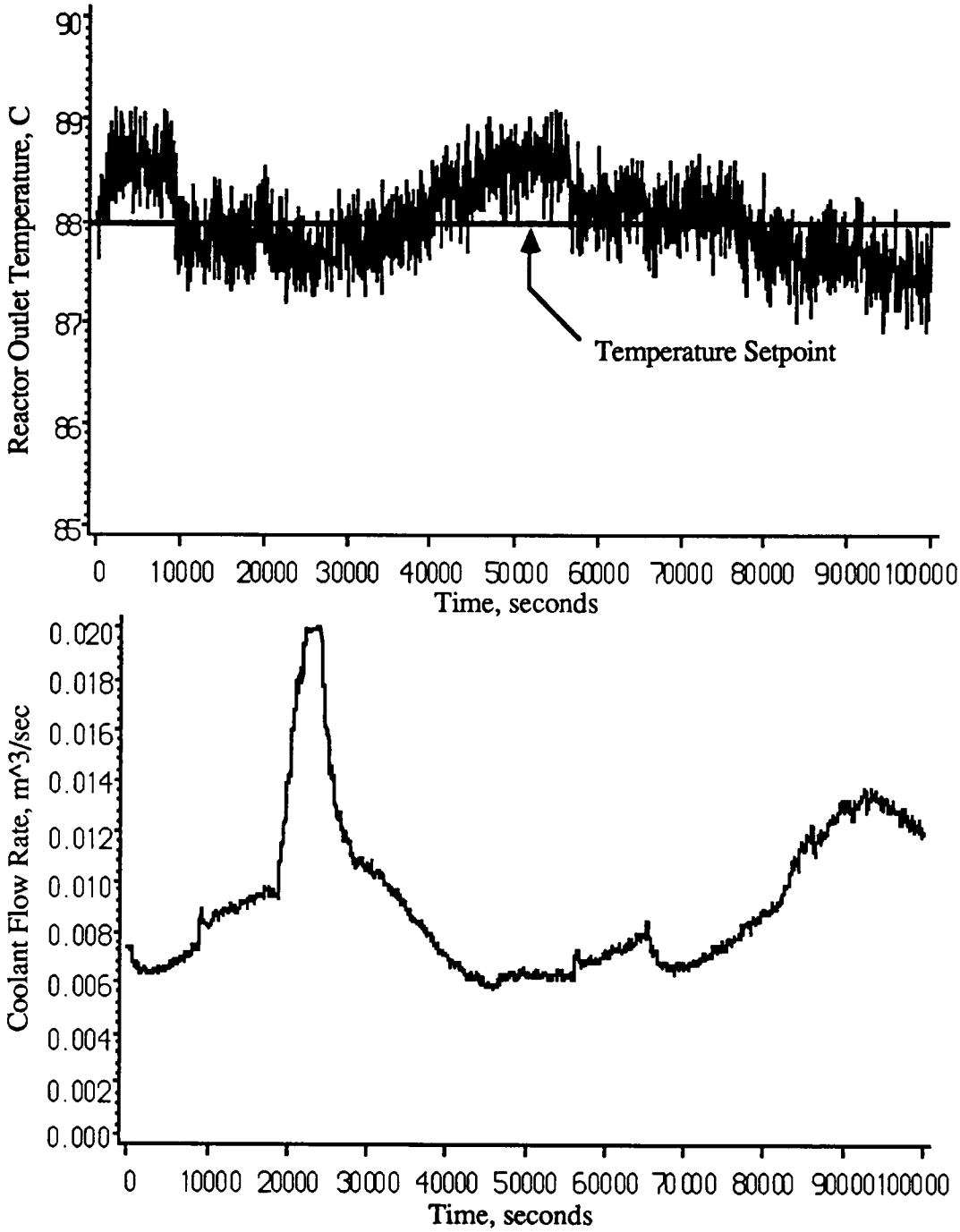


Figure 6.45: Reactor temperature and coolant flow rate for simulation 5 using initial condition method #3.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	32.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
8972.00	*bias1*	0.00	-0.0011	n.a.	0.71%
	bias2	0.00	0.5415	n.a.	0.10%
	bias3	0.00	1.158	n.a.	0.47%
13472.00	*bias1*	-0.0011	-0.0002	n.a.	0.72%
	bias2	0.5415	0.4241	n.a.	0.13%
	bias3	1.158	0.8642	n.a.	0.61%
38222.00	*bias1*	-0.0002	-0.0012	n.a.	0.69%
	bias2	0.4241	0.296	n.a.	0.08%
	bias3	0.8642	0.6395	n.a.	0.53%
56222.00	*bias1*	-0.0012	-0.0043	n.a.	0.30%
	bias2	0.296	0.8836	n.a.	0.08%
	bias3	0.6395	1.965	n.a.	0.42%
60722.00	*bias1*	-0.0043	-0.0067	n.a.	0.94%
	bias2	0.8836	0.9794	n.a.	0.10%
	bias3	1.965	2.176	n.a.	0.17%
65222.00	*bias1*	-0.0067	-0.0061	n.a.	0.54%
	bias2	0.9794	1.068	n.a.	0.17%
	bias3	2.176	2.398	n.a.	0.02%
77222.00	*bias1*	-0.0061	-0.0045	n.a.	0.76%
	bias2	1.068	1.125	n.a.	0.10%
	bias3	2.398	2.471	n.a.	0.89%
86222.00	*bias1*	-0.0045	-0.0041	n.a.	0.98%
	bias2	1.125	0.7976	n.a.	0.30%
	bias3	2.471	1.65	n.a.	1.02%

n.a. = not applicable, percent deviation based on difference between actual measurement and calculated model initial condition plus bias term

Figure 6.46: Feedback cultivation results for using initial condition method #3 during simulation 5.

shown in Figure 6.2. The variation is caused by the cyclic oscillation of the inlet coolant temperature shown in Figure 6.6.

In spite of the constant variation of the heat loss term, the feedback cultivation keeps the reactor temperature within 1.0 ^{0}C of the setpoint for all three initial condition methods. The feedback cultivation results shown in Figure 6.44 indicate that the model re-identification encounters more difficulty when the search variables of the optimization include both the bias terms and the initial conditions using initial condition search method #2. The percent deviations of the model initial state variable values from the actual CSTR output measurements are higher for this case than for the other two cases. This may be due to the increase in optimization search variables from just the three bias terms using initial condition methods #1 and #3 to six search variables using initial condition method #2. More search variables causes more difficulty during the optimization step.

6.2.1.6 Simulation 6

Simulation 6 involves decreasing the ambient temperature 5.0 ^{0}C for each of the three initial condition methods. All comparable model parameters equal the CSTR parameters for this simulation.

Figures 6.47 through 6.52 show the results of the simulation with feedback cultivation using each of the three initial condition methods. Note that for the case of using initial condition method #1 shown in Figures 6.47 and 6.48 stop after a time of 50000.0 seconds instead of proceeding to

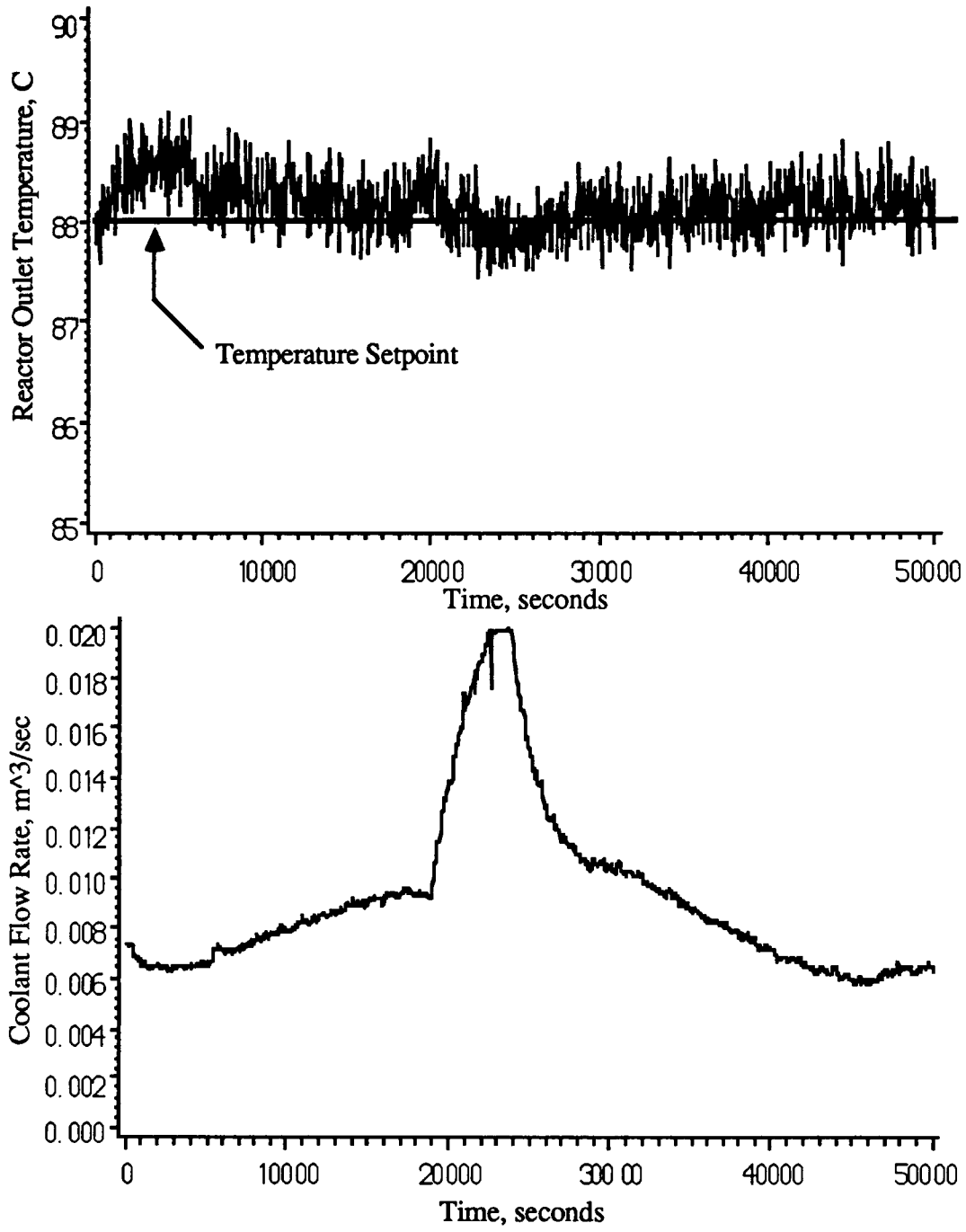


Figure 6.47: Reactor temperature and coolant flow rate for simulation 6 using initial condition method #1.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	22.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*bias1*	0.00	0.0009	n.a.	0.18%
	bias2	0.00	0.2452	n.a.	0.20%
	bias3	0.00	0.9083	n.a.	0.30%
12692.00	*cpc*	4184.00	4135.00	4184.00	1.17%

n.a. = not applicable, percent deviation based on difference between actual measurement and model initial condition plus bias term

Figure 6.48: Feedback cultivation results for using initial condition method #1 during simulation 6.

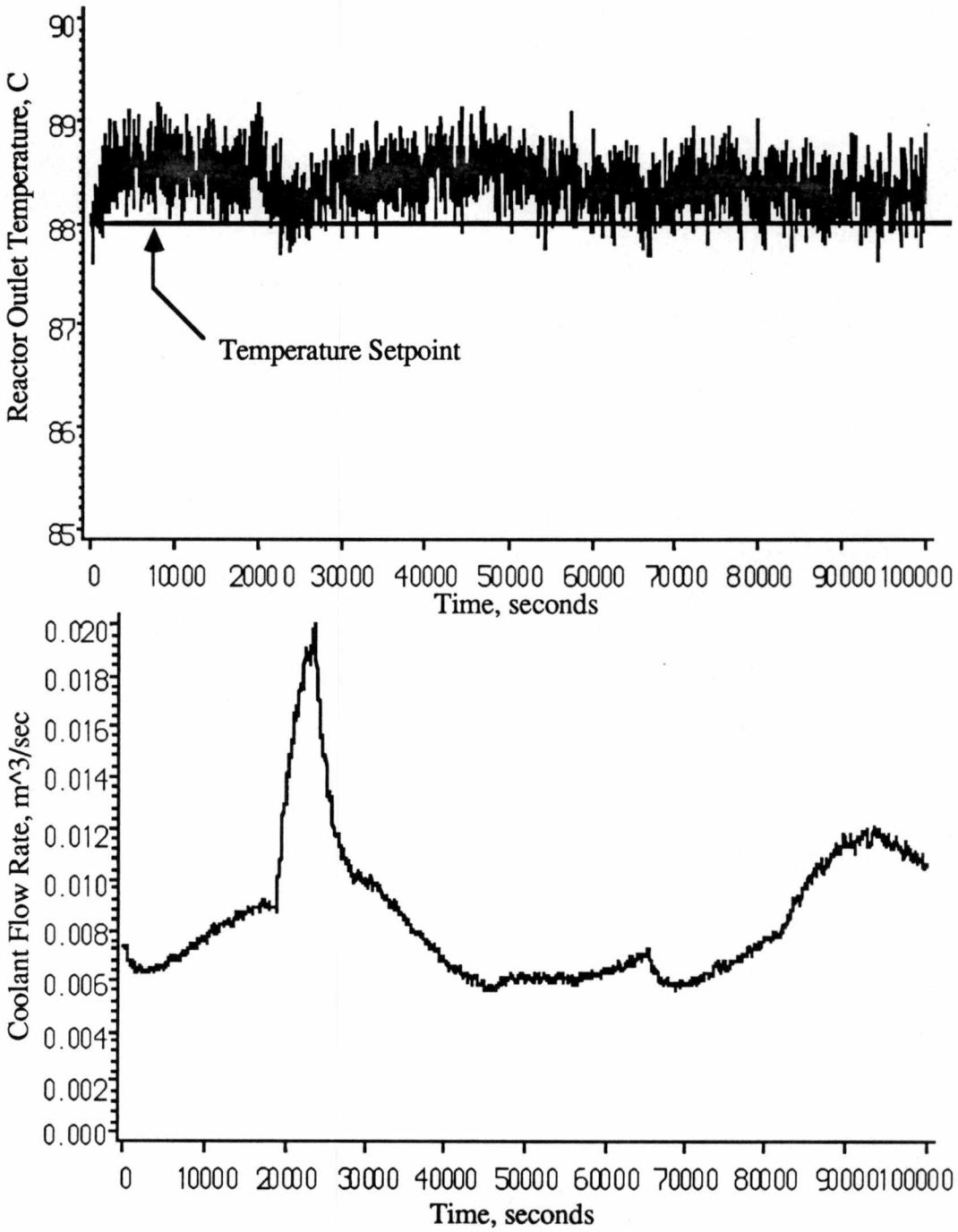


Figure 6.49: Reactor temperature and coolant flow rate for simulation 6 using initial condition method #2.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	22.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*bias1*	0.00	0.0085	n.a.	0.39%
	bias2	0.00	0.0091	n.a.	0.35%
	bias3	0.00	0.4737	n.a.	1.81%
42212.00	*bias1*	0.0085	-0.0084	n.a.	1.36%
	bias2	0.0091	0.0185	n.a.	0.39%
	bias3	0.4737	0.0916	n.a.	1.73%
73712.00	*bias1*	-0.0084	0.0307	n.a.	2.81%
	bias2	0.0185	0.0248	n.a.	0.34%
	bias3	0.0916	0.1228	n.a.	1.20%
87212.00	*bias1*	0.0307	-0.0132	n.a.	1.35%
	bias2	0.0248	0.0293	n.a.	0.15%
	bias3	0.1228	0.1403	n.a.	0.86%

n.a. = not applicable, percent deviation based on difference between actual measurement and model initial condition plus bias term

Figure 6.50: Feedback cultivation results for using initial condition method #2 during simulation 6.

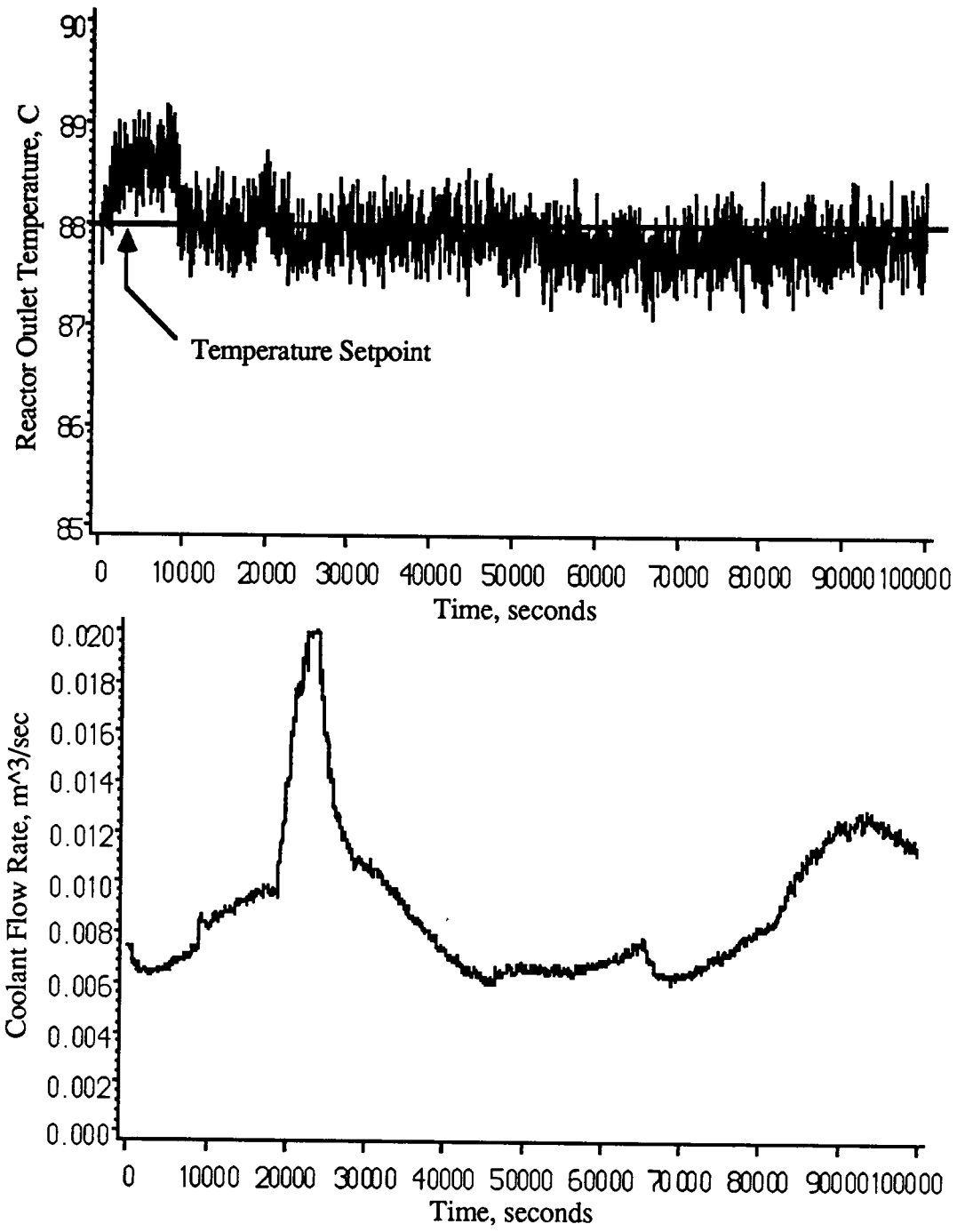


Figure 6.51: Reactor temperature and coolant flow rate for simulation 6 using initial condition method #3.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	22.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
8972.00	*cpc*	4184.00	3906.00	4184.00	6.64%

Figure 6.52: Feedback cultivation results for using initial condition method #3 during simulation 6.

100000.0 seconds as in the other two cases.

For all three initial condition methods, the feedback cultivation algorithm keeps the reactor temperature within 1.0 ^{0}C during the simulations. All three plots of the reactor temperature indicate slight offset for most of the simulation duration. Unlike the case of increasing the ambient temperature in simulation 5, the reactor temperature does not tend to drift up or down as significantly during the simulation.

The feedback cultivation results for using initial condition method #1 shown in Figure 6.48 indicate that the bias terms are modified once during the simulation correctly indicating the presence of an unmeasured disturbance. At a time of about 12000.0 seconds, the feedback cultivation algorithm incorrectly modifies the model value of the coolant heat capacity resulting in a 1.17% deviation from the CSTR value of the coolant heat capacity.

During the simulation using initial condition method #2, the feedback cultivation algorithm modifies only the bias terms four times as shown in Figure 6.50. During the simulation, the offset of the reactor temperature from setpoint seems to be declining slightly as the bias terms are modified as indicated in Figure 6.49.

The feedback cultivation results using initial condition method #3 in Figure 6.52 show that only the model value of the coolant heat capacity is modified during the simulation. After the modification, the setpoint offset is nearly eliminated with only slight drift as shown in Figure 6.51.

The results presented in this section and also in Section 6.2.1.5 for simulation 5 show that unmeasured disturbances are corrected for in most cases by the feedback cultivation algorithm modifying the bias terms of the model. Several times, the model value of the coolant heat capacity is modified. Since the CSTR coolant heat capacity appears in the heat loss term in the reactor jacket energy balance equation shown in Figure 6.2, modification of the model value of the coolant heat capacity does not seem unusual. As the results shown in Figures 6.51 and 6.52 indicate, modification of the model value of the coolant heat capacity effectively returns the reactor temperature to setpoint.

As with previous simulations, the feedback cultivation algorithm does not seem to perform better for any of the initial condition methods. All three initial condition methods in this simulation provide model parameters that maintain the reactor temperature within 1.0 ^{0}C of the setpoint.

6.2.1.7 Simulation 7

The final simulation presented involves a simultaneous change in the ambient temperature of +5.0 ^{0}C and a -10% change in the CSTR heat transfer coefficient. The change in each of the parameters occurs at the beginning of the simulation.

Figures 6.53 through 6.58 show the results for this simulation using each of the three initial condition methods. In all three cases, after an initial offset of almost 2.0 ^{0}C, the feedback cultivation algorithm maintains the

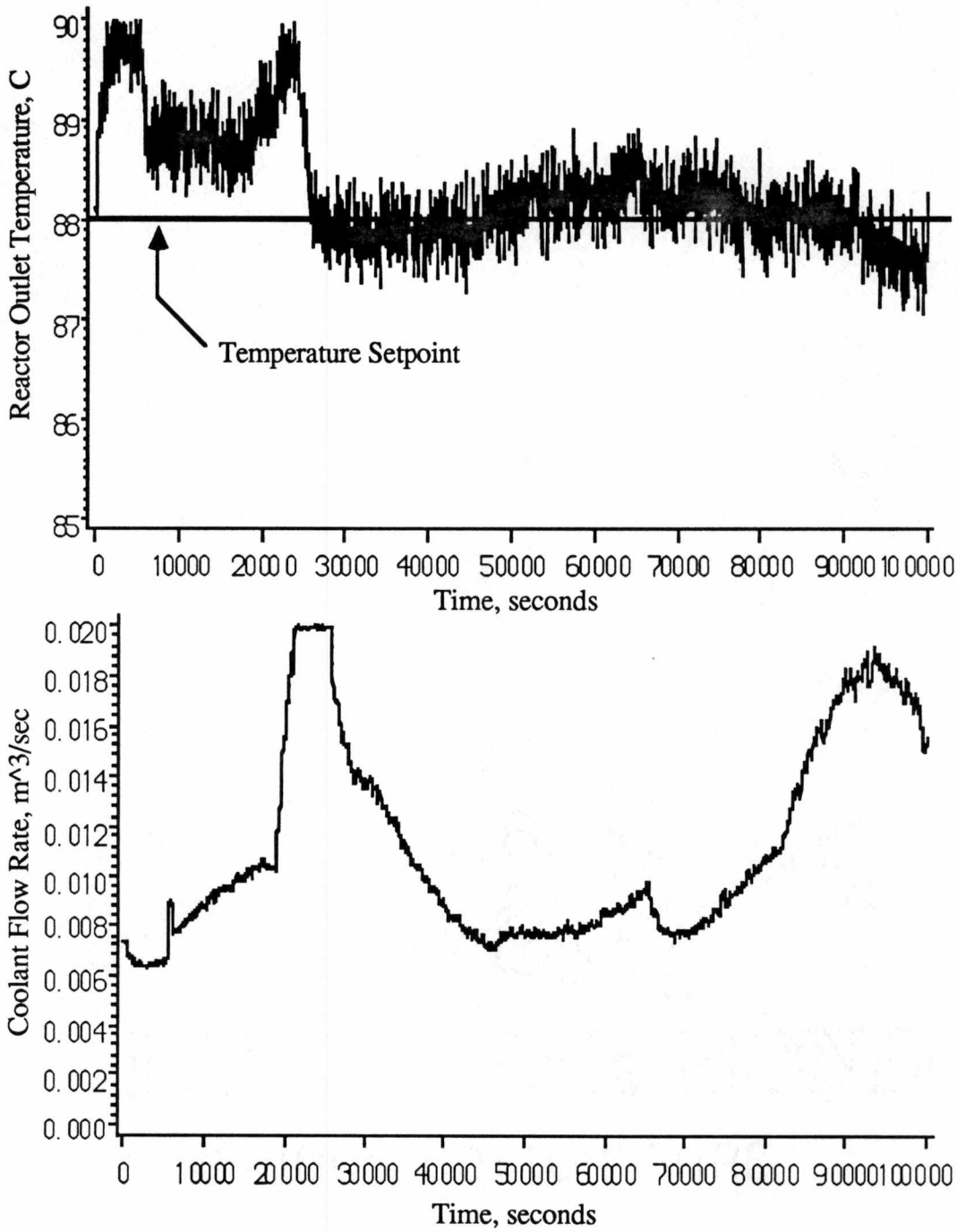


Figure 6.53: Reactor temperature and coolant flow rate for simulation 7 using initial condition method #1.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	32.00
	Uc	3550.00	3195.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*bias1*	0.00	-0.0018	n.a.	0.02%
	bias2	0.00	0.9974	n.a.	0.53%
	bias3	0.00	-0.2155	n.a.	9.60%
21212.00	*Cbi*	0.00	0.0722	0.00	>100%
33962.00	*bias1*	-0.0018	0.0002	n.a.	0.55%
	bias2	0.9974	0.9225	n.a.	0.01%
	bias3	-0.2155	-0.5081	n.a.	0.07%
54962.00	*bias1*	0.0002	-0.0051	n.a.	0.45%
	bias2	0.9225	1.113	n.a.	0.13%
	bias3	-0.5081	0.3972	n.a.	0.41%
59462.00	*bias1*	-0.0051	-0.0096	n.a.	0.41%
	bias2	1.113	1.321	n.a.	0.18%
	bias3	0.3972	0.8682	n.a.	0.30%
74462.00	*Cbi*	0.0722	0.09	0.00	>100%
99212.00	*bias1*	-0.0096	-0.0069	n.a.	0.19%
	bias2	1.321	1.079	n.a.	0.22%
	bias3	0.8682	-0.342	n.a.	0.53%

n.a. = not applicable, percent deviation based on difference between actual measurement and model initial condition plus bias term

Figure 6.54: Feedback cultivation results for using initial condition method #1 during simulation 7.

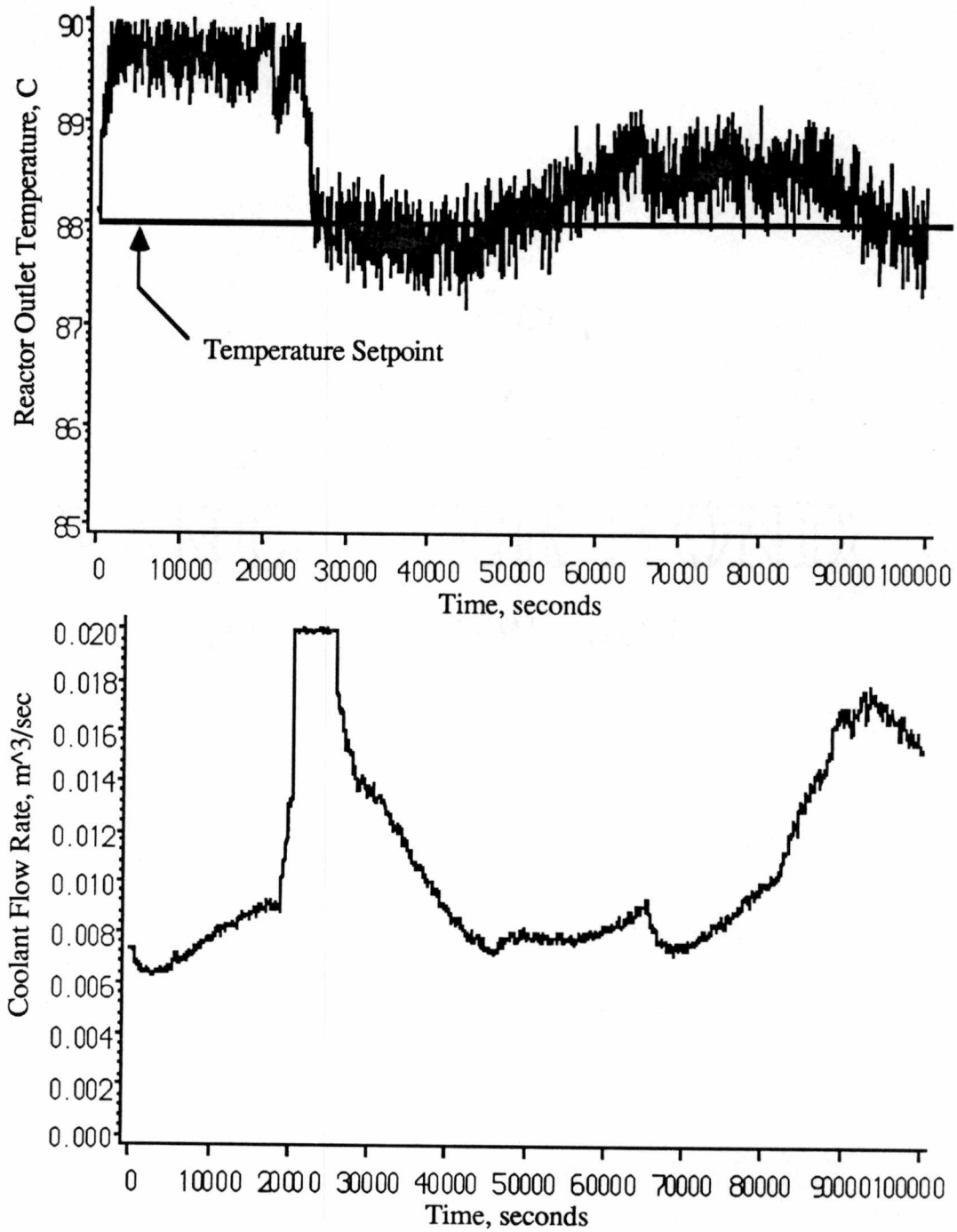


Figure 6.55: Reactor temperature and coolant flow rate for simulation 7 using initial condition method #2.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	32.00
	Uc	3550.00	3195.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
5462.00	*bias1*	0.00	-0.0001	n.a.	0.08%
	bias2	0.00	0.0313	n.a.	1.45%
	bias3	0.00	-0.0004	n.a.	0.38%
20462.00	*Cbi*	0.00	0.14	0.00	>100%
66962.00	*bias1*	-0.0001	-0.0098	n.a.	1.22%
	bias2	0.0313	0.0412	n.a.	0.24%
	bias3	-0.0004	0.0464	n.a.	1.82%
88712.00	*V*	7.08	7.269	7.08	2.68%
96962.00	*bias1*	-0.0098	-0.0089	n.a.	1.42%
	bias2	0.0412	0.0412	n.a.	0.01%
	bias3	0.0464	0.0209	n.a.	1.74%

n.a. = not applicable, percent deviation based on difference between actual measurement and model initial condition plus bias term

Figure 6.56: Feedback cultivation results for using initial condition method #2 during simulation 7.

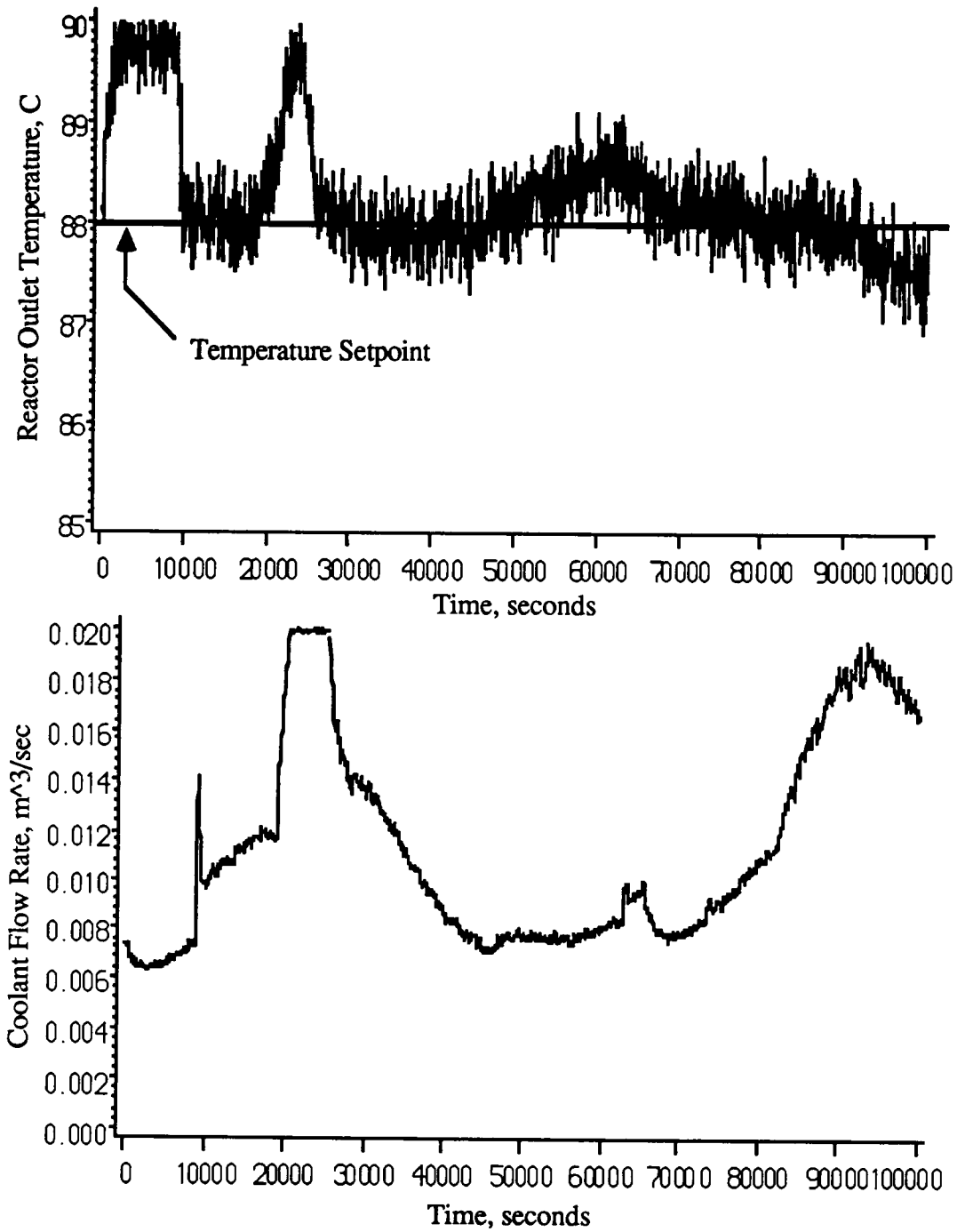


Figure 6.57: Reactor temperature and coolant flow rate for simulation 7 using initial condition method #3.

	Plant Parameters		
Time, sec	Changed Parameter	Old Value	New Value
10.00	*Tamb*	27.00	32.00
	Uc	3550.00	3195.00

	Model Parameters				
ID Search Time, sec	Search Variable	Old Value	New Value	Plant Parameter	Percent Deviation
8972.00	*bias1*	0.00	-0.0054	n.a.	1.14%
	bias2	0.00	1.761	n.a.	0.04%
	bias3	0.00	0.1305	n.a.	0.49%
28472.00	*Cbi*	0.00	0.012	0.00	>100%
62972.00	*bias1*	-0.0054	-0.0119	n.a.	0.17%
	bias2	1.761	2.266	n.a.	0.15%
	bias3	0.1305	1.278	n.a.	0.87%
73472.00	*bias1*	-0.0119	-0.012	n.a.	0.35%
	bias2	2.266	2.433	n.a.	0.09%
	bias3	1.278	1.62	n.a.	0.15%
77972.00	*bias1*	-0.012	-0.0105	n.a.	0.13%
	bias2	2.433	2.463	n.a.	0.09%
	bias3	1.62	1.507	n.a.	0.37%
82472.00	*bias1*	-0.0105	-0.0099	n.a.	0.62%
	bias2	2.463	2.452	n.a.	0.36%
	bias3	1.507	1.269	n.a.	1.04%

n.a. = not applicable, percent deviation based on difference between actual measurement and calculated model initial condition plus bias term

Figure 6.58: Feedback cultivation results for using initial condition method #3 during simulation 7.

reactor temperature within 1.0 ^{0}C of setpoint after a time of 30000.0 seconds. The reactor temperature displays the same tendency to drift up or down after the model re-identification returns the temperature to the setpoint. The same tendency occurs in simulation 5 when only the ambient temperature is increased as explained in Section 6.2.1.5.

Figures 6.53 and 6.54 show the results for using initial condition method #1 in the feedback cultivation algorithm during the simulation. For the most part, the feedback cultivation algorithm modifies the model bias terms to return and maintain the reactor temperature at the setpoint. And except for an initial deviation of 9.6% between the actual coolant temperature and the model predicted value of the coolant temperature at a time of 5462.0 seconds, the feedback cultivation algorithm keeps the deviation between the actual measured variable values and the model predicted values of the variables to about 0.50% or less. Twice during the simulation though, the feedback cultivation algorithm incorrectly modifies the model value of the inlet impurity concentration.

Figures 6.55 and 6.56 show the results for using initial condition method #2 in the feedback cultivation algorithm. In this case, the feedback cultivation algorithm also modifies the bias terms to return the reactor temperature to setpoint. But the first time the bias terms are modified at a time of 5500.0 seconds, the change seems to have no effect on the reactor temperature. Only after the model value of the inlet impurity concentration is incorrectly modified at a time of 20000.0 seconds does the reactor

temperature return to the setpoint. The feedback cultivation algorithm also incorrectly modifies the model value of the reactor volume near the end of the simulation.

In the case of using initial condition method #2, the feedback cultivation algorithm does not keep the deviations between the model predicted values of the state variable values as close to the actual measured values as when initial condition method #1 is used. The deviations range from very small to almost 2.0% as shown in Figure 6.56.

Figures 6.57 and 6.58 show the results for using initial condition method #3 in the feedback cultivation algorithm. In this case the feedback cultivation algorithm returns and maintains the reactor temperature to setpoint by modifying the bias terms for the most part. After modifying the bias terms at a time of 9000.0 seconds, the reactor temperature returns to setpoint and remains at setpoint until a time of 20000.0 seconds. At this time, the reactor temperature rises sharply until the feedback cultivation algorithm incorrectly modifies the model value of the inlet impurity concentration. This happens because the values of the disturbance variables between 20000.0 and 30000.0 seconds change significantly and cause the controller to fully open the coolant flow control valve. This changes the operating conditions of the reactor significantly and since the model parameters do not match the CSTR parameters, the controller cannot maintain the reactor temperature at the setpoint value. This problem is not so obvious in the other two cases.

6.2.1.8 PI-Controller Comparisons

The response of the CSTR for a simulation with the PI-controller described in Smith and Corripio [65] is compared to the nonlinear optimal control algorithm for the simulation described in Section 6.2.1.2. In this simulation, the plant parameters follow the schedule of change shown in Figure 6.24. Figure 6.59 shows the reactor outlet temperature and the coolant flow rate for this simulation. Comparing the coolant flow rate in Figure 6.59 to the coolant flow rate in Figure 6.23 indicates that the effect of noise in the measurment signals on the reactor outlet temperature for the nonlinear optimal control algorithm is less than for the PI-controller. Comparing the reactor outlet temperatures in the two plots shows that the PI-controller keeps the temperature at the setpoint more effectively than the nonlinear optimal control algorithm. This is due to the way the two controllers deal with feedback errors. The PI-controller makes no distinction between different sources of feedback error. The PI-controller constantly attempts to force the controlled variable, in this case, the reactor temperature, back to the setpoint value. The nonlinear optimal control algorithm waits until the source of the feedback error has been determined before taking corrective action. During the delay, the controlled variable may deviate from the setpoint.

The comparison of the PI-controller to the more complex nonlinear optimal control algorithm demonstrates that the PI-controller provides adequate control of the CSTR as effectively or more effectively than the nonlinear optimal control algorithm. With a more complex process or a

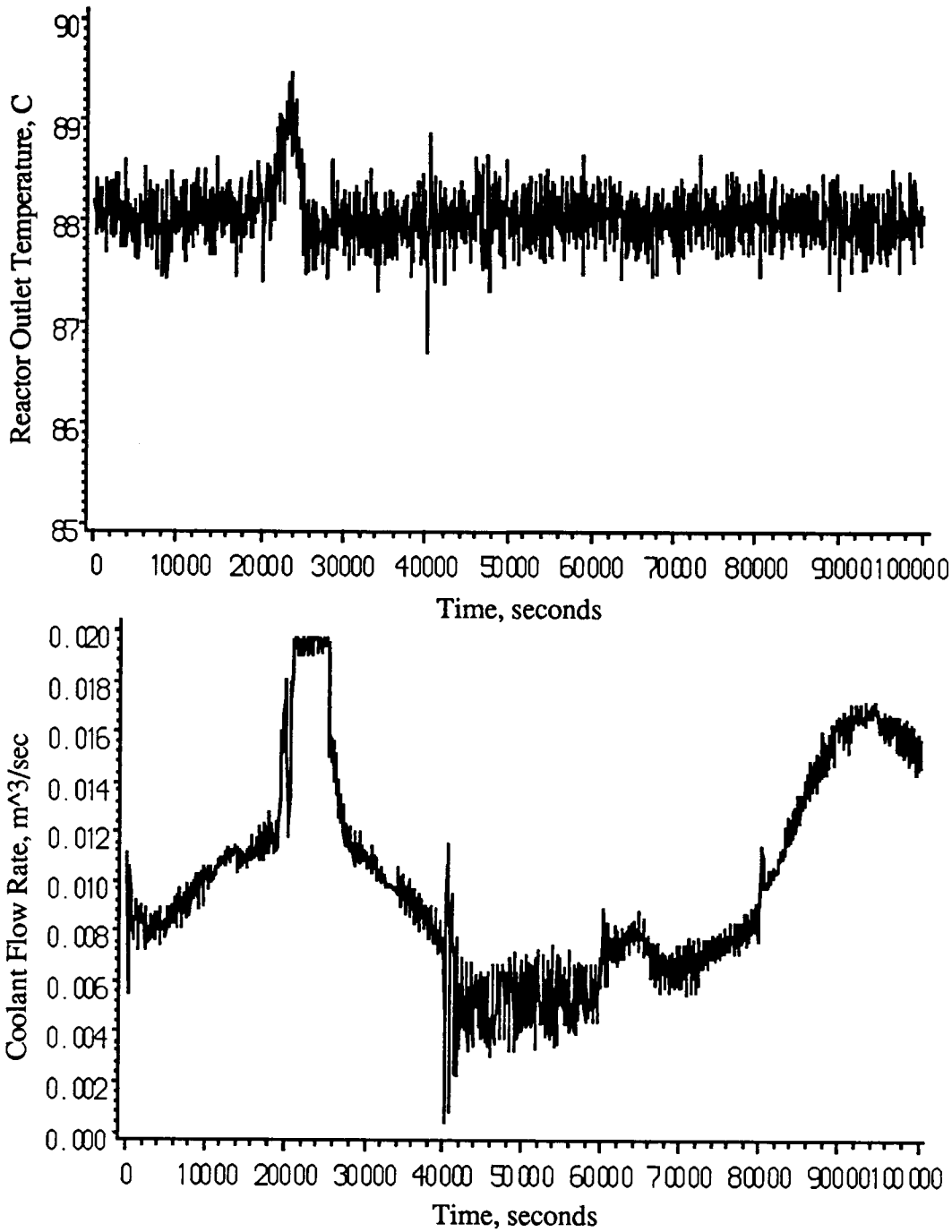


Figure 6.59: Results of a simulation of the CSTR with a PI-controller and the plant parameter disturbances used for simulation 2.

multivariable process, the PI-controller may not function as well as it does for the CSTR.

The results also show that the PI-controller is somewhat more sensitive to measurement signal noise. Sensitivity to noise can cause excessive flucuations of the manipulated variable – in this case, the coolant flow rate. Excessive flucuations cause undesirable wear and tear on control elements such as flow control valves. The nonlinear optimal control algorithm limits the flucuations of the manipulated variable which reduces the possibility of damage to control elements.

6.2.2 Adaptive Filter Comparisons

In this section, results of comparing the adaptive filter algorithm to two first-order filters are presented. Also presented is a brief analysis of the variation of the adaptive filter weight values during a CSTR simulation for each of the disturbance and output variable measurements. The CSTR simulation used to produce these results contained no unmeasured disturbances or model parameter errors. The values of the four disturbance variables, F, C_{ai}, T_i, and T_{ci} varied during the simulation as shown in Figures 6.4 through 6.7.

6.2.2.1 Filter Algorithm Comparisons

The performance of the adaptive filter algorithm is compared to the performance of a first-order filter with "guessed" filter constants and to the

performance of a first-order filter with optimally determined filter constants. Section 6.1.2.4 gives the values of the adaptive filter parameters. The filter constants used for each measurement signal in the first-order filter with "guessed" filter constants is $\alpha = 0.9$.

As discussed in Section 6.1.2.4, comparison to an optimally tuned first-order filter is possible because the non-noisy measurement signals are available during the simulation. The filter constant for each measurement signal is optimized by minimizing the square difference between the non-noisy measurement signal value and the noisy measurement signal value for each CSTR disturbance and output measurement signal.

Figure 6.60 presents performance criteria for simulations of the CSTR using the adaptive filter algorithm and the two first-order algorithms. The figure also lists the average and standard deviation of the reactor temperature occurring during the simulation for each of the filter algorithms. Since the non-noisy signals are available during the simulation, a comparison is made between the non-noisy signals and the resulting filtered signals. These results are reported as the sum of square differences and the sum of absolute differences for the reactor temperature only in Figure 6.60. And to demonstrate that any filtering is better than no filtering, all of the results described above are presented for simulating the CSTR without any signal filtering.

The performance criteria indicate that the adaptive filter algorithm results in less variation of the reactor temperature from setpoint and, for the

Filter Type	SSE	SAE	SSDU	SADU
Adaptive	0.582	10.2	0.0002505	0.162
First Guess	2.07	22.1	0.0003875	0.15
First Optimal	1.29	17.7	0.0004258	0.169
No Filter	14.7	54.4	0.02114	1.98

			Pure T - Filtered T	
Filter Type	Average T	T Std. Dev.	Square Sum	Absolute Sum
Adaptive	88.001	0.2148	0.627	10.53
First Guess	88.035	0.2146	0.616	11.61
First Optimal	88.042	0.2117	0.4595	9.952
No Filter	88.012	0.2109	14.00	53.06

T = Reactor outlet temperature
Tset = Setpoint temperature
SSE = Sum of squares of (Tset - T)
SAE = Sum of absolute values of (Tset - T)
SSDU = Sum of square changes in control move
SADU = Sum of absolute values of control move changes
Pure T = Reactor outlet temperature signal before noise added
Filtered T = Reactor outlet temperature signal after filtering

Figure 6.60: Results of comparing the adaptive filter algorithm to two first-order filter algorithms and to using no filtering

most part, slightly smaller changes in the control moves during the simulation. The optimally tuned first-order filter algorithm performs slightly better than the first-order filter algorithm with guessed filter constants. All of the filter algorithms perform better in terms of the performance criteria than using no filtering.

When using the adaptive filter algorithm, the average value of the reactor temperature is nearly equal to the setpoint value of 88.0 ^{0}C. The first-order filters result in slightly worse average values than the case of using no filtering. The standard deviations are nearly equal for all the filter algorithms including no filtering which has the smallest standard deviation.

The results of comparing the filtered reactor temperature to the non-noisy reactor temperature show that the optimally tuned first-order filter performs the best. But the adaptive filter algorithm results are similar to the results for the optimally tuned first-order filters and the first-order filters with guessed filter constants.

6.2.2.2 Adaptive Filter Weight Variation

As discussed in Section 6.1.2.4, the adaptive filter algorithm uses 10 weights for all seven measurements. In the adaptive filter algorithm, the last weight value, W_{10}, is multiplied by the latest measurement on down to multiplying the oldest measurement by W_1. Because of this, the weights are reported in reverse order.

Figure 6.61 shows the variation of all 10 weights for the reactor

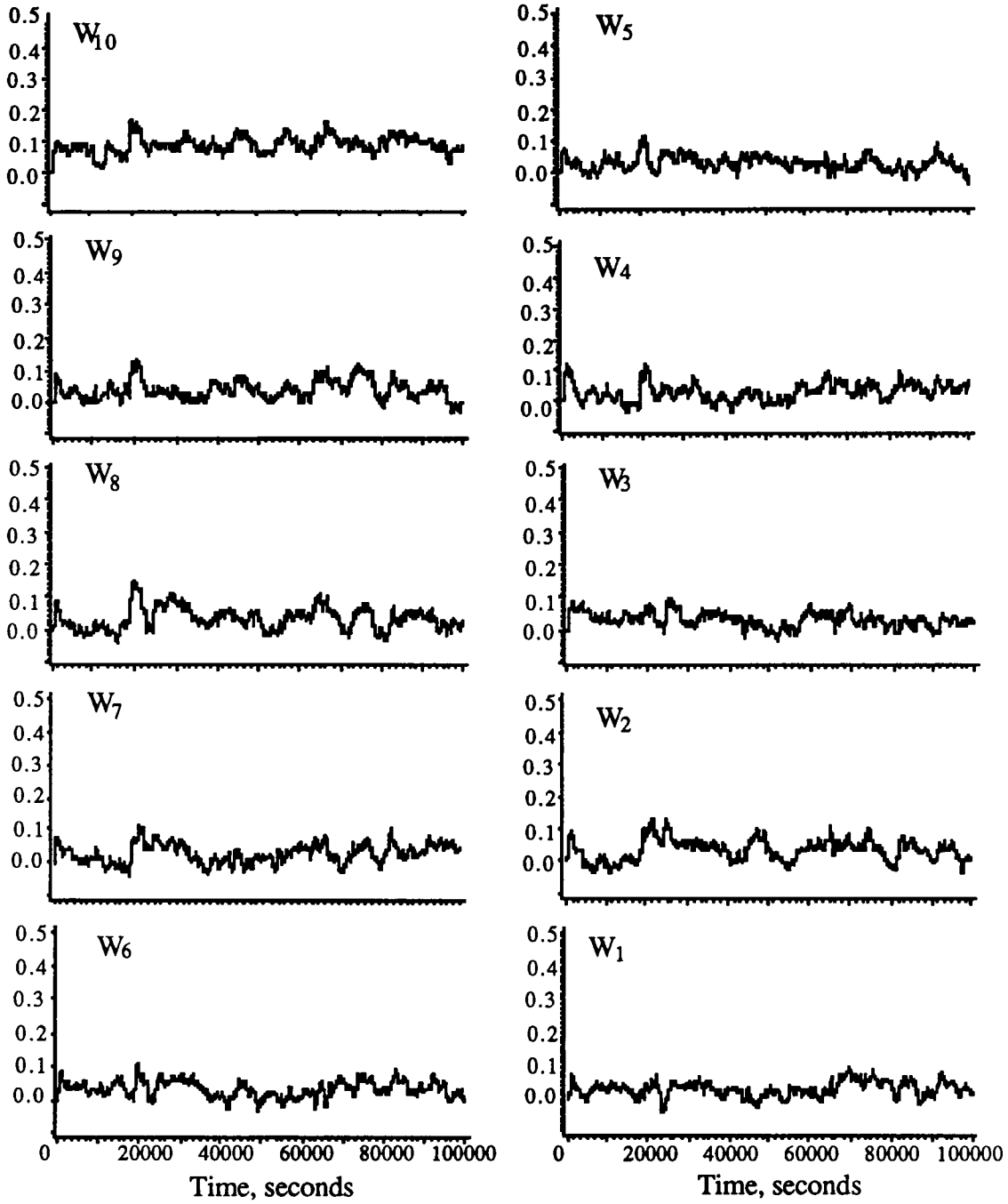


Figure 6.61: Adaptive filter weights for the reactor temperature measurement signal.

temperature filter during the CSTR simulation. All of the weights vary between 0.0 and 0.1 during the simulation.

Figure 6.62 shows the variation of all 10 weights for the outlet reactant concentration measurement filter. The weights for the C_a filter vary more slowly and with less oscillation than the weights for the T filter. The first couple of weights, W_{10} and W_9, have not reached nearly constant values by the end of the simulation as have some of the other weights such as W_1 and W_2. A larger value for the adaptive filter LMS convergence factor, μ, may be needed for the adaptive filter algorithm for C_a.

Figures 6.63 through 6.67 show only the value of W_{10} for each of the other five measurement filter algorithms. The plots of W_{10} for the inlet feed temperature, T_i, the inlet coolant temperature, T_{ci}, and the jacket coolant temperature, T_c shown in Figures 6.63 to 6.65 respectively, compare with the plot of W_{10} for the reactor temperature, T, shown in Figure 6.61. Figure 6.66 shows that the variation of W_{10} for the inlet concentration filter is similar to the variation of W_{10} for the outlet concentration filter shown in Figure 6.62.

The weight factor W_{10} varies more for the feed flow rate, F, filter than for the temperature filters or the concentration filters as shown in figure 6.67. A smaller value of the LMS convergence factor, μ may be appropriate for this adaptive filter algorithm.

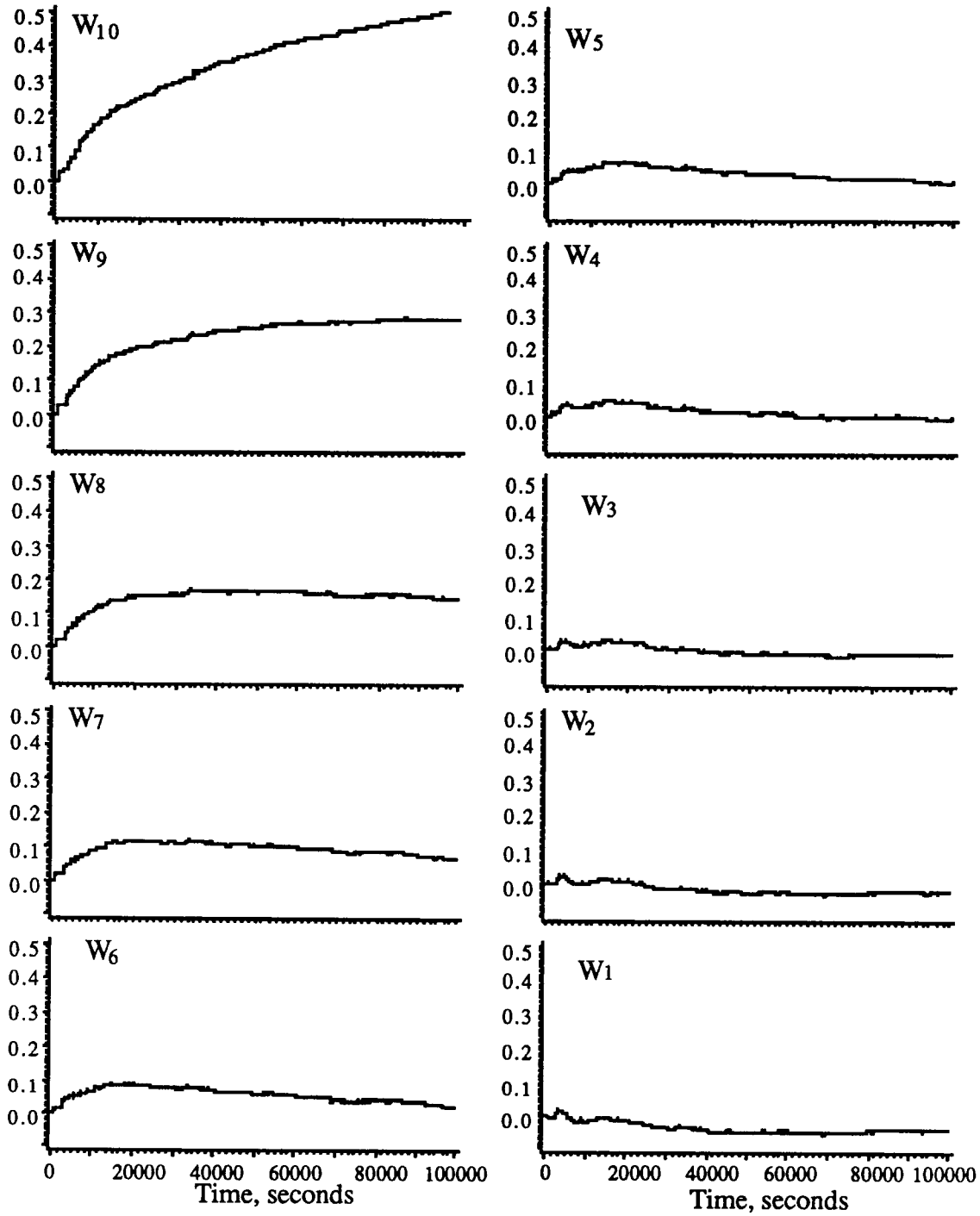


Figure 6.62: Adaptive filter weights for the outlet reactant concentration measurement signal

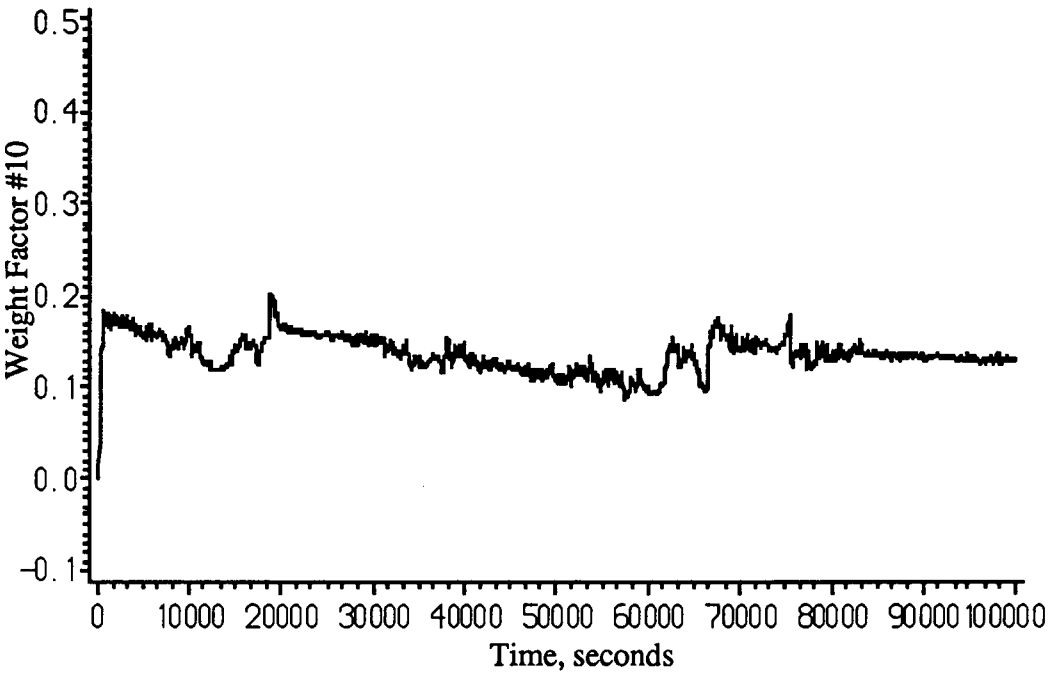


Figure 6.63: Adaptive filter weight W_{10} for the inlet temperature measurement signal

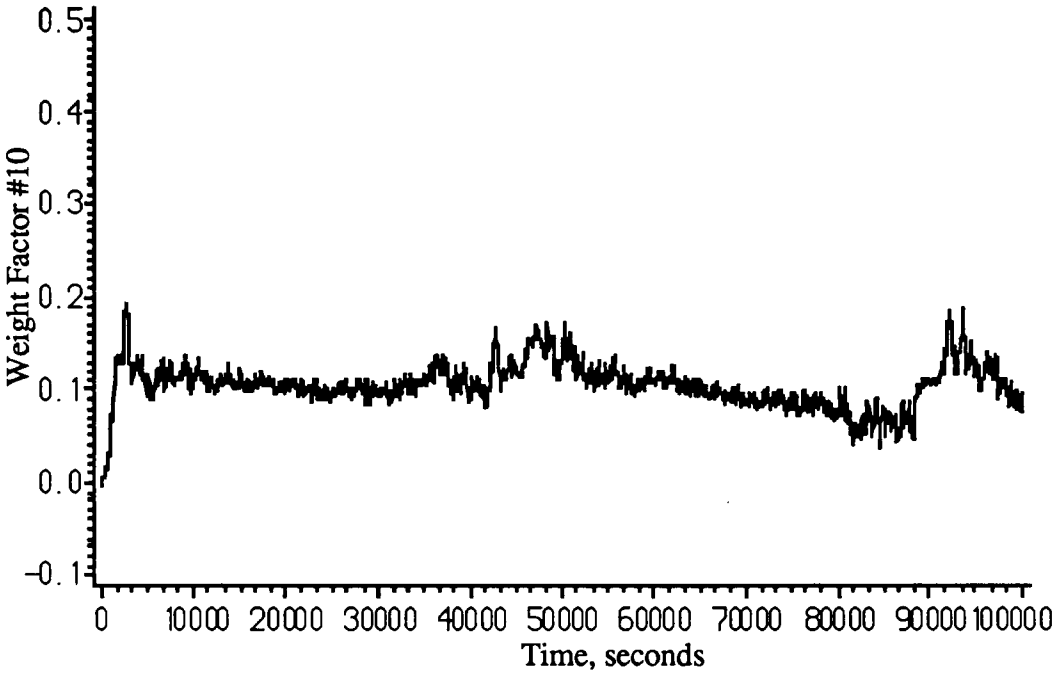


Figure 6.64: Adaptive filter weight W_{10} for the inlet coolant temperature measurement signal

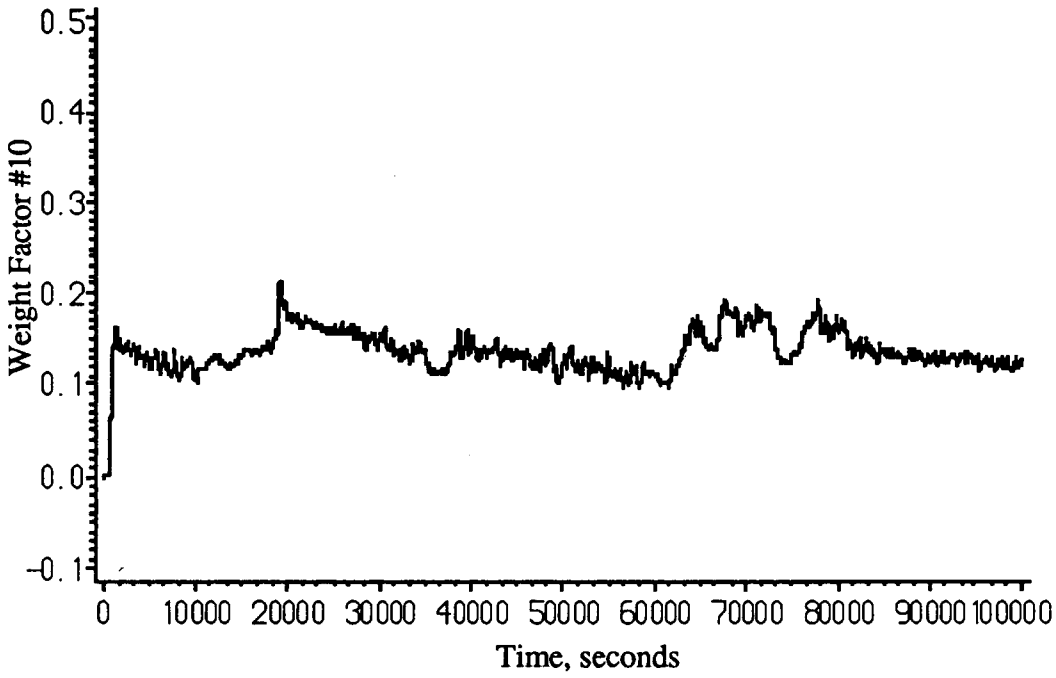


Figure 6.65: Adaptive filter weight W_{10} for the reactor jacket coolant temperature measurement signal

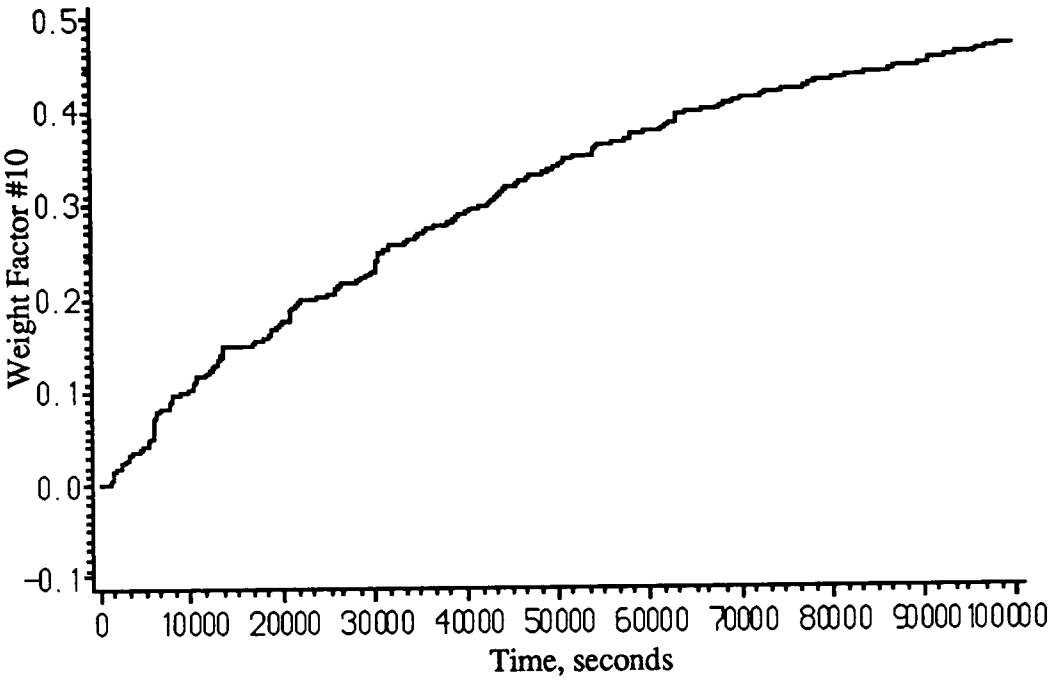


Figure 6.66: Adaptive filter weight W_{10} for the inlet reactant concentration measurement signal

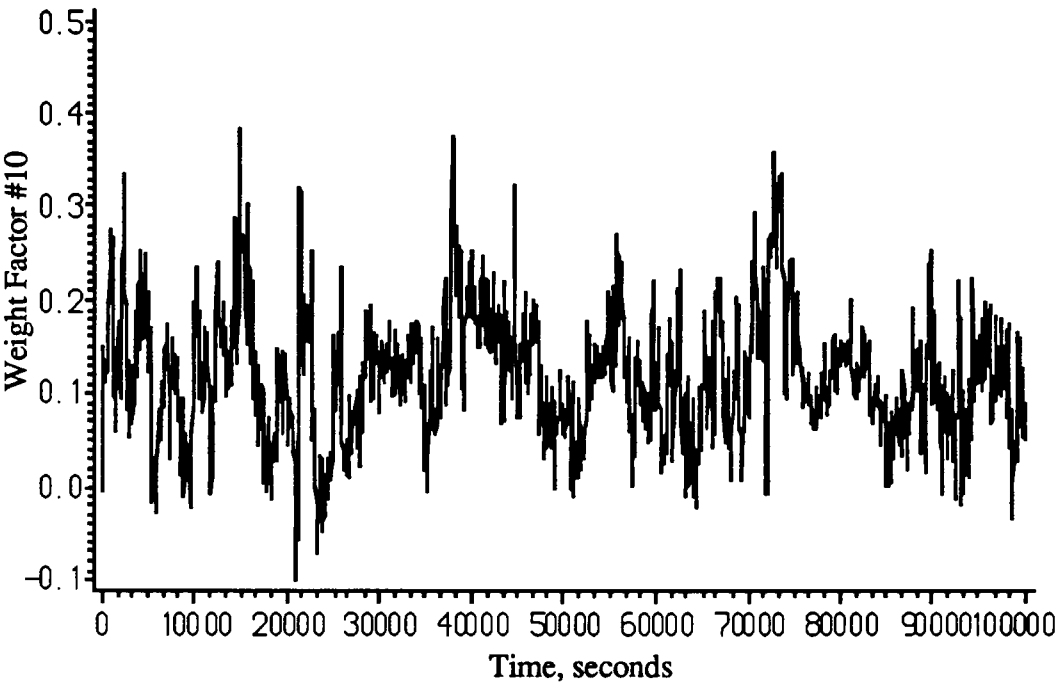


Figure 6.67: Adaptive filter weight W_{10} for the inlet feed flow rate measurement signal

6.2.3 Results Discussion

The results presented in Section 6.2.1 demonstrate that the nonlinear optimal control algorithm functions in the face of measurement signal noise, measured and unmeasured disturbances, and process-to-model mismatch. The results also indicate that using any of the three initial condition methods in the feedback cultivation algorithm provides adequate control in most cases. In only one instance, using initial condition method #2 during simulation 1, did the feedback cultivation algorithm fail to compensate for deviations between the reactor temperature and the setpoint. Had the simulation proceeded further, the feedback cultivation may have eventually modified a model parameter to compensate for the deviation in this case.

The feedback cultivation results presented for the CSTR in this chapter differ from the feedback cultivation results for the first-order example presented in Chapter 5 in that no one initial condition determination method clearly works better than the others. In Chapter 5, using initial condition method #1 and selectively modifying the model parameters worked better than any of the other methods. The results presented in Section 6.2.1 indicate that the initial condition method that works best depends on the nature of the simulation.

The results presented in Section 6.2.2 demonstrate that the adaptive filter algorithm works as well and possibly better than a first-order filter algorithm. Also, the variation of the some of the adaptive filter weights for the measurement signals is shown in Section 6.2.2. These results indicate that

the nature of the filter weights depend on the dynamic charateristics of each measurement signal. The adaptive filter algorithm is better suited to take advantage of this dependence than the first-order filter algorithm because some of the adaptive filter parameters are modifiable, namely the weights. First-order filter algorithms have just the one filter constant that is arbitrarily set and held constant.

CHAPTER 7

Conclusions and Future Work

The work presented in this dissertation addresses the issues of dealing with measurement signal noise, unmeasured disturbances, and process-to-model mismatch using a nonlinear optimal control algorithm. The feedback cultivation algorithm demonstrated in this work addresses these issues. The feedback cultivation algorithm incorporates a number of techniques including adaptive filtering, neural networks, and statistical analysis.

This work is by no means complete. A lot of problems remain unsolved concerning the feedback cultivation algorithm. The results presented in Chapters 5 and 6 examine only one set of feedback cultivation variables for the sample simulations. Also, only one variation of the neural network structure was tried in each case.

In spite of this lack of completeness of this work, a number of relevent conclusions can be drawn resulting from the work completed so far:

- The feedback cultivation algorithm can be successfully used to eliminate steady state offset between controlled variables and their respective setpoints. The results presented in Chapter 6 especially show this to be true. The feedback cultivation algorithm applied in Chapter 5 tended to be too sensitive to deviations between the model parameters and the

plant parameters including deviations between the model predicted value of the state variable and the actual measured value of the variable.

- The feedback cultivation results presented in Chapter 6 demonstrated that even if the wrong model parameters are modified, deviation of the controlled variable from setpoint can be eliminated.

- The back propagation perceptron neural network algorithm can be trained to distinguish between various process-to-model parameter deviations. In both examples, the neural network had difficulty distinguishing between certain classes of data during both training and operating modes. More judicious selection of the feedback cultivation variables included in the input to the neural network may correct the difficulty.

- An adaptive filtering algorithm can be used in place of a first-order filter algorithm with equal or better performance characteristics. The results in Chapter 6 show that the adaptive filter algorithm causes the nonlinear optimal control algorithm to function better in terms of the overall process simulation performance criteria.

- A nonlinear optimal control algorithm along with any model predictive control scheme must be tuned just like any other controller. The results in Chapter 5 show this for the first-order process example. The disadvantage of having to tune a model predictive control algorithm is

that the rules for manipulating the various tuning parameters are still in the infancy of development.

- Coupling the feedback cultivation algorithm with the nonlinear optimal control algorithm provides a more logical and intelligent way of incorporating feedback into the nonlinear optimal control algorithm. Using the feedback cultivation algorithm, the sources of deviations from setpoint are determined and eliminated. Unmeasured disturbances are treated as unmeasured disturbances, not as process-to-model mismatch like other methods of incorporating feedback.

- The nonlinear optimal control algorithm is very sensitive to parameter discrepancies. The first-order example demonstrates that the nonlinear optimal control algorithm fails to keep the process output at the setpoint when process-to-model mismatch exists.

- The feedback cultivation algorithm compensates for the nonlinear optimal control algorithm's sensitivity to parameter mismatch but a lot of historical process data is required to perform the feedback cultivation. During the data accumulation before the feedback cultivation algorithm can be implemented, offset from setpoint may be present.

- Comparisons of the PI-controller to the nonlinear optimal control algorithm for the CSTR demonstrates that for this particular process, the PI-controller is superior with respect to both performance and simplicity. In spite of the nonlinearities in the CSTR, the dynamics are

such that the PI-controller adequately deals with the disturbances encountered by the process.

The results presented in this work represent only the beginning of analysing the nonlinear optimal control algorithm with feedback cultivation. Before these techniques find use in an industrial setting, a lot more research must be performed using the algorithm with simulated processes. The most significant items that need to be addressed include the following:

- Different sets of feedback cultivation variables need to be tried for input to the neural network to better distinguish between process-to-model mismatch errors and unmeasured disturbances.

- Different perceptron neural network structures and also completely different neural network algorithms need to be tried to attempt to improve the feedback cultivation datapoint classification phase of the feedback cultivation algorithm.

- The complete nonlinear optimal control algorithm needs to be demonstrated using a multivariable process. The algorithm is very well suited to for extrapolation to the multivariable case.

- The adaptive filter algorithm needs more in-depth analysis to determine whether or not it is a good replacement for first-order filter algorithms or any other types of measurement signal filtering algorithms.

The above list mentions only the most significant suggestions for future work

involving the nonlinear optimal control algorithm. Many other aspects of the nonlinear optimal control algorithm contain weaknesses that need to be addressed before any sort of industrial application can occur.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Abdelmalek, N. N., "A Simplex Algorithm for Minimum Fuel Problems of Linear Discrete Control Systems", *Int. J. Control*, **26**(4):635–642, 1977.

[2] Balchen, J. and Mumme, K., *Process Control Structures and Applications*, Van Nostrand Reinhold Co. Inc., New York, 1988.

[3] Bazarra, M. S. and Shetty, C. M., *Nonlinear Programming*, John Wiley and Sons, New York, New York, 1979.

[4] Bhat, N. and McAvoy, T., "Dynamic Process Modeling Via Neural Computing", In *1989 Annual Meeting of the AIChE*, San Francisco, CA, 1989.

[5] Bhat, N.V., Minderman, P.A., and McAvoy, T., "Modeling Chemical Process Systems via Neural Computation", *IEEE Control Systems Magazine*, 24–30, 1990.

[6] Biegler, L., "Solution of Dynamic Optimization Problems by Successive Quadratic Programming and Orthogonal Collocation", *Computers and Chemical Engineering*, **8**(3/4):243–248, 1984.

[7] Bourdache-Siguerdidjane, H. and Fliess, M., "Optimal Feedback Control of Nonlinear Systems", *Automatica*, **23**(3):365–372, 1987.

[8] Brosilow, C., Zhao, G. Q., and Rao, K. C., "A Linear Programming Approach to Constrained Multivariable Control", In *ACC*, Boston, MA, 1985.

[9] Bryson, Arthur E. Jr. and Ho, Yu-Chi, *Applied Optimal Control*, Hemisphere Publishing Corp., Washington D.C., revised edition, 1975.

[10] Chang, T. S. and Seborg, D. E., "A Linear Programming Approach for Multivariable Feedback Control with Inequality Constraints", *Int. J. Control*, **37**(3):583–597, 1983.

[11] Chang, T. S. and Seborg, D. E., "Process Control in the Presence of Constraints", In *ACC*, Boston, MA, 1985.

[12] Choi, K-Y. and Khan, A. A., "Optimal State Estimation in the Transesterification Stage of a Continuous Polyethylene Terephthalate Condensation Polymerization Process", *Chemical Engineering Science*, **43**(4):749–762, 1988.

[13] Clarke, D. W., "Application of Generalized Predictive Control to Industrial Processes", *IEEE Control Systems Magazine*, 49–55, April 1988.

[14] Clarke, D. W., Mohtadi, C., and Tuffs, P. S., "Generalized Predictive Control - Part 1: The Basic Algorithm", *Automatica*, **23**(2):137–148, 1987.

[15] Clarke, D. W., Mohtadi, C., and Tuffs, P. S., "Generalized Predictive Control - Part 2: Extensions and Interpretations", *Automatica*, **23**(2):149–160, 1987.

[16] Cuthrell, J. E. and Biegler, L. T., "On the Optimization of Differential-Algebraic Process Systems", *AIChE Journal*, **33**(8):1257–1270, August 1987.

[17] Cuthrell, James E., *On the Optimization of Differential-Algebraic Systems of Equations in Chemical Engineering*, PhD thesis, Carnegie-Mellon University, 1986.

[18] Cutler, C. R. and Finlayson, S. G., "Design Considerations for a Hydrocracker Preflash Column Multivariable Constraint Controller", In *IFAC Workshop on Model Based Process Control*, Atlanta, June 1988.

[19] Cutler, C. R. and Ramaker, B. L, "Dynamic Matrix Control — A Computer Control Algorithm", In *Proceedings of the JACC*, San Francisco, CA, 1980.

[20] Eaton, J. W., Rawlings, J. B., and Edgar, T. F., "Model Predictive Control and Sensitivity Analysis for Constrained Nonlinear Processes", In *IFAC Workshop on Model Based Process Control*, Atlanta, June 1988.

[21] Economou, C. G., *An Operator Theory Approach to Nonlinear Controller Design*, PhD thesis, Caltech, 1985.

[22] Farell, A. E., *Application of Process Control Cultivation to a Liquid-Liquid Heat Exchanger*, Master's thesis, The University of Tennessee, Knoxville, 1987.

[23] Farell, A. E., *System Cultivation: Process Monitoring and Control Using the Historical Database*, PhD thesis, The University of Tennessee, Knoxville, 1990.

[24] Garcia, C. E. and Morari, M., "IMC. Part 3. Multivariable Control Law Computation and Tuning Guidelines", *IEC Proc. Des. Dev.*, **24**:484–494, 1985.

[25] Garcia, C. E. and Morari, M., "Internal Model Control. Part 1. A Unifying Review and Some New Results", *IEC Proc. Des. Dev.*, **21**:308–323, 1982.

[26] Garcia, C. E. and Morari, M., "Optimal Processing Systems. Part 1: Open-Loop On-Line Optimizing Control", *AIChE Journal*, **27**(6):960–968, November 1981.

[27] Garcia, C. E. and Morshedi, A. M., "Quadratic Programming Solution of Dynamic Matrix Control (QDMC)", 1985, Submitted to *Chemical Engineering Communications.*

[28] Glad, S. T., "Robustness of Nonlinear State Feedback - A Survey", *Automatica*, **23**(4):425–435, 1987.

[29] Grimble, M. J., "Discrete Dual-Rate Stochastic Optimal Control", *Trans. Inst. M. C.*, **10**(1):41–49, 1988.

[30] Grosdidier, P. and Froisy, B., "The IDCOM-M Controller", In *IFAC Workshop on Model Based Process Control*, Atlanta, June 1988.

[31] Gupta, Y. P., "Suboptimal Control Through Model Reduction", *Canadian Journal of ChE*, **66**:141–144, February 1988.

[32] Harris, T. J. and Macgregor, J. F., "Design pf Multivariable Linear-Quadratic Controllers Using Transfer Functions", *AIChE Journal*, **33**(9):1481–1495, September 1987.

[33] Hecht-Nielson, R., "Kolmorgorov's Mapping Neural Network Existence Theorem", In *Proceedings of IEEE Conference on Neural Networks*, San Diego, CA, 1987.

[34] Hoskins, J. C. and Himmelblau, D. M., "Fault Detection and Diagnosis Using Artificial Neural Networks", 123–160, 1990.

[35] Jakoby, W. and Pandit, M., "A Prediction-Error-Method for Recursive Identification of Nonlinear Systems", *Automatica*, **23**(4):491–496, 1987.

[36] Jang, Shi-Shang, *Control and Optimization of Constrained Multivariable Processes*, PhD thesis, Washington University, 1986.

[37] Jang, Shi-Shang, Joseph, B., and Mukai, H., "Comparison of Two Approaches to On-Line Parameter and State Estimation of Nonlinear Systems", *IEC Proc. Des. Dev.*, **25**:809–814, 1986.

[38] Jang, Shi-Shang, Joseph, B., and Mukai, H., "Control of Constrained Multivariable Nonlinear Processes Using a Two-Phase Approach", *IEC Research*, **26**:2106–2114, 1987.

[39] Jang, Shi-Shang, Joseph, B., and Mukai, H., "On-Line Optimization of Constrained Multivariable Chemical Processes", *AIChE Journal*, **33**(1):26–35, January 1987.

[40] Jorgensen, S. B., "An Approach to Multivariable Process Identification", In *IFAC Workshop on Model Based Process Control*, Atlanta, June 1988.

[41] Kalman, R. E. and Bucy, R. S., "New Results in Linear Filtering and Prediction Theory", *Trans. ASME J. Basic Engineering*, **82D**:95, 1961.

[42] Lee, P. L. and Sullivan, G. R., "Generic Model Control - Theory and Applications", In *IFAC Workshop on Model Based Process Control*, Atlanta, June 1988.

[43] Mehra, R. K., Rouhani, R., Eterno, J., Richalet, J., and Rault, A., "Model Algorithm Control (MAC) ; Review and Recent Developments", In *Chemical Process Control II*, 1980.

[44] Minsky, M. and Papert, S., *Perceptrons*, MIT Press, Cambridge, 1969.

[45] Minsky, M. L., *Perceptrons*, The MIT Press, 1972.

[46] Moore, B. C., "Multivariable Challenges in the Petrochemical Industry", 1986, Technical Report Number 8601.

[47] Moore, B.C., "Systematic Attention to Detail by Production Organizations for Sustained Improvement", In *Second Nat. Sym. on Statistical Process Control*, Tempe, Az, 1987.

[48] Moore, B.C., "What and Who is in Control?", In *The Shell Process Control Workshop*, 1988.

[49] Moore, C. F., Smith, C. L., and Murrill, P. W., "Improved algorithms for direct digital control", *Instruments and Control Systems*, ():70–74, 1970.

[50] Morari, M., Garcia, C. E., and Prett, D. M., "Model Predictive Control: Theory and Practice", In *IFAC Workshop on Model Based Process Control*, Atlanta, June 1988.

[51] Morshedi, A. M., "Universal Dynamic Matrix Control", In Morari, M. and McAvoy, T. J., editors, *Chemical Process Control — CPCIII*, pp. 547–577, New York, 1986.

[52] Morshedi, A. M., Cutler, C. R., and Skrovanek, T. A., "Optimal Solution of Dynamic Matrix Control with Linear Programming Techniques (LDMC)", In *ACC*, Boston, MA, 1985.

[53] Ogunnaike, B. A., "Dynamic Matrix Control: A Nonstochastic, Industrial Process Control Technique with Parallels in Applied Statistics", *IEC Fundamentals*, **25**:712–718, 1986.

[54] Parlos, A. G., Henry, A. F., Schweppe, F. C., Gould, L. A., and Lanning, D. D., "Nonlinear Multivariable Control of Nuclear Power Plants Based on the Unknown-but-Bounded Disturbance Model", *IEEE Transactions on Automatic Control*, **33**(2):130–137, 1988.

[55] Peterson, T., Hernandez, E., Arkun, Y., and Schork, F. J., "Nonlinear Predictive Control of a Semi Batch Polymerization Reactor by an Extended DMC", 1988, unpublished report.

[56] Pontryagin, L. S. *et al.*, *The Mathematical Theory of Optimal Processes*, Interscience, New York, New York, 1962.

[57] Prett, D. M. and Gillette, R. D., "Optimization and Constrained Multivariable Control of a Catalytic Cracking Unit", In *AIChE National Meeting*, Houston, 1989.

[58] Richalet, J., Rault, A., Testud, J. L., and Papon, J., "Model Predictive Heuristic Control: Applications to Industrial Processes", *Automatica*, **14**:413–428, 1978.

[59] Ricker, N. L., "Use of Quadratic Programming for Constrained Internal Model Control", *IECPDD*, **24**:925–936, 1985.

[60] Ricker, N. L., Subrahmanian, T., and Sim, T., "Case Studies of Model Predictive Control in Pulp and Paper Production", In *IFAC Workshop on Model Based Process Control*, Atlanta, June 1988.

[61] Rosenblatt, F., "The Perceptron: A Probilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, **65**:386, 1958.

[62] Rumelhart, D. and McClelland, J., "An Interactive Activation Model of Context Effects in Letter Perception: Part 2. The Contextual Enhancement Effect and Some Tests and Extensions of the Model", *Psychological Review*, **89**:60, 1982.

[63] Rumelhart, D. and McClelland, J., *Parallel Distributed Processing, Vol. I and II*, MIT Press, Cambridge Mass., 1987.

[64] Sen, S. and Yakowitz, S. J., "A Quasi-Newton Differential Dynamic Programming Algorithm for Discrete-Time Optimal Control", *Automatica*, **23**(6):749–752, 1987.

[65] Smith, C. A. and Corripio, A. B., *Principles and Practice of Automatic Process Control*, John Wiley & Sons, New York, 1985.

[66] Smith, O. J. M., "Closer control of loops with dead time", *Chemical Engineering Progress*, **53**(5):217–219, 1957.

[67] von Westerholt, E., Melsheimer, S. S., and Beard, J., "Neural Network Modeling of Chemical Engineering Processes", 1990, unpublished report.

[68] Widrow, B., "Adaptive Filters", 563–587, 1982.

[69] Widrow, B. and Stearns, S., *Adaptive Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, 1985.

[70] Widrow, B. and Winter, R., "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition", *IEEE Computer*, **18**(3):25–39, 1988.

[71] Widrow, B., Winter, R. G., and Baxter, R. A., "Learning Phenomena in Layered Neural Networks", In *IEEE 1st Int. Conf. on Neural Networks. Proceedings*, pp. 411–429, San Diego, 1987.

[72] Winter, R. and Widrow, B., "Madeline Rule II: A Training Algorithm for Neural Networks", In *IEEE 2nd Int. Conf. on Neural Networks. Proceedings*, pp. 401–408, San Diego, 1988.

[73] Zafiriou, E., "Robustness and Tuning of On-line Optimizing Control Algorithms", In *IFAC Workshop on Model Based Process Control*, Atlanta, June 1988.

VITA

Suzanne D. Roat was born March 14, 1963 in Camden, New Jersey. She attended high school in Wilmington, Delaware graduating magna cum laude. Suzanne attended Clemson University in Clemson, South Carolina while pursuing a Bachlor's degree in chemical engineering. She graduated from Clemson University in May 1985 summa cum laude with Senior Departmental Honors. Suzanne relocated to Knoxville to attend the University of Tennessee to pursue a Master's degree and PhD in chemical engineering.