# PRODUCT OPTIMIZATION IN VINYL ACETATE MONOMER PROCESS USING MODEL PREDICTIVE CONTROL

AAINAA IZYAN BINTI NAFSUN

UNIVERSITI MALAYSIA PAHANG

**KOLEJ UNIVERSITI KEJURUTERAAN & TEKNOLOGI MALAYSIA**

# BORANG PENGESAHAN STATUS TESIS

JUDUL: **PRODUCT OPTIMIZATION IN VINYL ACETATE MONOMER PROCESS USING MODEL PREDICTIVE CONTROL**

**SESI PENGAJIAN**: **2007/2008**

Saya **AAINAA IZYAN BINTI NAFSUN**
**(HURUF BESAR)**

mengaku membenarkan kertas projek ini disimpan di Perpustakaan Universiti Malaysia Pahang dengan syarat-syarat kegunaan seperti berikut :

1. Hakmilik kertas projek adalah di bawah nama penulis melainkan penulisan sebagai projek bersama dan dibiayai oleh UMP, hakmiliknya adalah kepunyaan UMP.
2. Naskah salinan di dalam bentuk kertas atau mikro hanya boleh dibuat dengan kebenaran bertulis daripada penulis.
3. Perpustakaan Universiti Malaysia Pahang dibenarkan membuat salinan untuk tujuan pengajian mereka.
4. Kertas projek hanya boleh diterbitkan dengan kebenaran penulis. Bayaran royalti adalah mengikut kadar yang dipersetujui kelak.
5. *Saya membenarkan/tidak membenarkan Perpustakaan membuat salinan kertas projek ini sebagai bahan pertukaran di antara institusi pengajian tinggi.
6. ** Sila tandakan (✓ )

| | | |
|---|---|---|
| ☐ | SULIT | (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972) |
| ☐ | TERHAD | (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan). |
| ☑ | TIDAK TERHAD | |

Disahkan oleh

(TANDATANGAN PENULIS) (TANDATANGAN PENYELIA)

Alamat Tetap:
**NO.15, TAMAN SRI INDAH,**
**JALAN SEPANGGAR, MENGGATAL,**
**88450 KOTA KINABALU,**
**SABAH.**

**NOOR ASMA FAZLI ABDUL SAMAD**
Nama Penyelia

Tarikh: Tarikh:

CATATAN:
* Potong yang tidak berkenaan.
** Jika Kertas Projek ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai SULIT atau TERHAD.
♦ Tesis ini dimaksudkan sebagai tesis bagi Ijazah Doktor Falsafah dan Sarjana secara penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM)

"I hereby declare that I have read this thesis and in
my opinion, this thesis is sufficient in terms of scope and
quality for the award of degree of Bachelor of Chemical Engineering"

Signature :

Name of Supervisor : NOOR ASMA FAZLI ABDUL SAMAD

Date: : May 2008

# PRODUCT OPTIMIZATION IN VINYL ACETATE MONOMER PROCESS USING MODEL PREDICTIVE CONTROL

AAINAA IZYAN BINTI NAFSUN

A thesis submitted in fulfillment of the
requirements for the award of the degree of
Bachelor of Chemical Engineering

Faculty of Chemical and Natural Resources Engineering
University Malaysia Pahang

MAY 2008

I declare that this thesis entitled “PRODUCT OPTIMIZATION IN VINYL ACETATE MONOMER PROCESS USING MODEL PREDICTIVE CONTROL” is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature :

Name : AAINAA IZYAN BINTI NAFSUN

Date : May 2008

For my beloved parents, sisters and brothers……

## ACKNOWLEDGEMENT

Assalamualaikum wbkt,

A very grateful to the Almighty, for give me strength and guidance to finish my final year project.

I would like to express my deepest thanks and gratitude to my supervisor Mr. Noor Asma Fazli Abdul Samad for his valuable comment, encouragement, guidance, useful suggestions as well as assistance throughout this research. Without his support, this thesis could not be finish.

My sincere appreciation also extends to all my colleagues and group members who helped me in many different ways at various instances of my research. Their support and views are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my members in Universiti Malaysia Pahang.

# ABSTRACT

The needs for effective control performance in the face of highly process interactions have call for better plantwide process control system synthesis method. As a practical illustration, a vinyl acetate monomer plant was considered. The aim was to develop a suitable control model and then its performance was analyzed. This research underwent several stages. First, data was generated from the simulation of vinyl acetate monomer process. This studies was performed using MATLAB71. This was followed by analyses of dynamic response of the process. Transfer functions was developed using First Order Plus Time Delay (FOPTD) equation. These transfer function are then used in development of Model Predictive Control (MPC). Lastly, model testing of vinyl acetate monomer process is done and followed by tuning process. The optimum value of Prediction horizon (P) and Control horizon (M) is determined from the tuning process. The result lead to the conclusion that the Model Predictive Control is better than PI controller specifically in optimize the desired production of vinyl acetate.

# ABSTRAK

Keperluan kepada kawalan yang efektif dalam menghadapi interaksi proses yang tinggi memerlukan kaedah- kaedah sintesis system kawalan seluruh loji yang baik. Sebagai ilustrasi yang praktikal, sebuah loji penghasilan monomer vinyl acetate telah digunakan. Tujuannya adalah untuk menghasilkan satu kawalan yang sesuai dan menganalisa prestasinya. Penyelidikan ini dilaksanakan melalui beberapa peringkat. Pertama, data- data telah dihasilkan daripada simulasi proses monomer vinyl acetate. Kajian ini telah dijalankan menggunakan perisian MATLAB7.1. Ini diikuti dengan menganalisis reaksi dinamik proses. Fungsi pemindahan kemudian dicipta menggunakan persamaan ‘First Order Plus Time Delay (FOPTD)’. Fungsi pemindahan ini kemudian digunakan di dalam pembangunan ‘Model Predictive Control (MPC)’. Akhir sekali, ujian ke atas model yang dihasilkan dilakukan dan ini dilakukan dengan menguabahsuai nilai P dan M di dalam model. Nilai optimum P dan M ditentukan melalui ujian ini. Keputusan simulasi membawa kepasa kesimpulan bahawa MPC adalah lebih baik dari pengawal PI terutama dalam memgoptimumkan penghasilan produk vinyl acetate.

# TABLE OF CONTENT

# LIST OF TABLES

# LIST OF FIGURES

# ABBREVIATIONS AND NOMENCLATURE

## Abbreviations

| | |
|---|---|
| MPC | - Model Predictive Control |
| MISO | - Multi Input Single Output |
| VAM | - Vinyl Acetate Monomer |
| IDCOM | - Identification And Command |
| MAC | - Model Algorithmic Control |
| MPHC | - Model Predictive Heuristic Control |
| MV | - Manipulated Variables |
| DV | - Disturbance Variables |
| CV | - Controlled Variables |
| SISO | - Single Input, Single Output |
| DMC | - Dynamic Matrix Control |
| QDMC | - Quadratic Dynamic Matrix Control |
| SMOC | - Shell Multivariable Optimizing Control |
| RMPC | - Robust Model Predictive Control |
| RMPCT | - Robust Model Predictive Control Technology |
| SMCA | - Set Point Multivariable Control Architecture |
| DCS | - Distributed Computer System |
| HIECON | - Command-Hierarchical Controller |
| PFC | - Predictive Functional Control |
| OPC | - Optimum Predictive Control |
| MMC | - Modular Multivariable Control |
| MMAC | - Multiple Model Adaptive Control |
| N | - Model Horizon |
| P | - Prediction Horizon |
| M | - Control Horizon |
| T | - Sample Time |

| | |
|---|---|
| FOPDT | - First Order Plus Time Delay |
| MISO | - Multi Input, Single Output |
| VAC | - Vinyl Acetate |
| HAC | - Acetic Acid |
| VLE | - Vapor liquid equilibrium |
| FEHE | - Feed Effluent Heat Exchanger |

**Nomenclatures**

| | |
|---|---|
| $c_p$ | - heat capacity in cal/g$^o$C |
| $t$ | - temperature in $^o$C. |
| $P^s$ | - vapor pressure in Psia |
| $Q^{VAP}$ | - external heat flux |
| $V_L^{VAP}$ | - liquid holdup |
| $\Delta$P | - pressure drop |
| $f$ | - constant friction factor |
| $F_V^{VAP}$ | - mass flow rate of the vapor |
| $\rho_1^{RCT}$ | - the mass density of the feed stream in reactor (kg/m$^3$) |
| $\upsilon_1^{RCT}$ | - volumetric flowrate of the feed stream in reactor (m$^3$/min) |
| $\phi_i$ | - catalyst activity |
| $\theta_{1,j}, \theta_{2,j}$ | - stoichiometric coefficients for component *j* in the two reactions |
| $r_{1,i}, r_{2,i}$ | - reaction rates in section *i* |
| $E_1, E_2$ | - heats of reactions |
| $Q_i^{RCT}$ | - external heat flux per unit volume in section *i* in rector |
| $UA$ | - total thermal resistance |
| $T_S$ | - shell temperature, $^o$C |
| $F_1^{FEHE}$ | - mass flow rate of the cold stream |

$F_2^{FEHE}$ - mass flow rate of the hot stream

$\gamma$ - compressor coefficient

$\rho^{COM}$ - compressor inlet stream density

$N_i$ - molar flow rate of component *i* (kmol/min)

$N_{MT}$ - constant mass transfer coefficien

$y_i$ - mole fraction of component *i* in the vapor inlet stream

$Q_{MT,j}$ - constant heat transfer coefficient

$T_{V,j}$ - temperature of the vapor inlet stream

$T_{L,j}$ - temperature of the liquid phase

$F_{CO_2}$ - $CO_2$ inlet stream flow rate (kmol/min)

$x_{CO_2}$ - mole fraction of $CO_2$ in the inlet stream

## LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

## 1.1 Background of Study

In the past, process design had been approached in a hierarchical fashion where design problems were solved initially by developing simple solutions, followed by addition of successive levels of details to the process. Consequently, dynamic properties of the process were not considered until the final stage when the control system formulation was considered. This has changed recent years. The introduction of high speed computers has facilitated the use of optimization techniques in the formulation of plant configurations leading to more efficient design that are complex and integrated. Complicate process control such moves alters the dynamic and steady state behaviors of the individual unit operations, leading to poor process dynamics. Usually the control that conducted on individual unit operation cannot show the actual efficiency of the plant and it is shift to control on interacting units of the whole plant.

This study is inspired by vinyl acetate monomer process. This process contains several standard unit operations that are typical of many chemical plants. Both gas and liquid recycle streams are present as well as process-to-process heat integration. The process model contains 246 states, 26 manipulated variables, and 43 measurements. This polymerization process is difficult to control because it is involve molecular weight distribution and it is also highly interacting process. To tackle these problems, advanced control technique that is Model Predictive Control

(MPC) was chosen to control separator unit as well as optimize the product concentration.

MPC is an advanced technique controllers which is rely on dynamic models of the process, most often linear empirical models obtained by system identification. The models are used to predict the behavior of dependent variables of a dynamical system with respect to changes in the process independent variables.

In chemical processes, independent variables are most often set points of regulatory controllers that govern valve movement (e.g., valve position with or without flow, temperature or pressure controller cascades), while dependent variables are most often constraints in the process (e.g., product purity, equipment safe operating limits). The model predictive controller uses the models and current plant measurements to calculate future moves in the independent variables that will result in operation that honors all independent and dependent variable constraints. The MPC then sends this set of independent variable moves to the corresponding regulatory controller set points to be implemented in the process.

Despite the fact that most real processes are approximately linear within only a limited operating window, linear MPC approaches are used in the majority of applications with the feedback mechanism of the MPC compensating for prediction errors due to structural mismatch between the model and the plant. In model predictive controllers that consist only of linear models, the superposition principle of linear algebra enables the effect of changes in multiple independent variables to be added together to predict the response of the dependent variables. This simplifies the control problem to a series of direct matrix algebra calculations that are fast and robust.

## 1.2 Problem Statement

In a large scale plant, the output variables may be influenced by many input variables in a way which is not easy to predict. Complicated process control since such moves alter the dynamic and steady state behaviors of the individual unit operations, leading to poor process dynamics. This situation is further exacerbated by today's production criteria, which are increasingly difficult to satisfy. Product specifications are now largely more stringent and the plants are subjected to increasingly strict safety and environment standards. Large over-designed margins are rarely permitted leading to tight equipment constrains.

Chemical industries also deal with more complexes, nonlinear and highly interacting process which is hard to control with traditional controller. Interacting behavior is exhibit in the processes with variables that interact with each other or that contain internal feedback of material and energy. In interacting process, the units and variables are relating each other. A change in a unit has an affect on the other units. These pose serious challenge to process control and unless a well-designed control system is in place, the desired plant objectives may not be achievable. The traditional approach of control system design by eliminating conflicts following the completion of the individual unit control system formulation is not seen as viable to be used in these demanding circumstances. An advanced control system that is Model Predictive Control is therefore needed.

## 1.3 Objectives of Study

The aim of this study is to design a Model Predictive Control for a separator in vinyl acetate monomer process with good dynamic performance as well as optimize the product concentration. The scheme is developed and tested by rigorous simulation using MATLAB 7.1.

## 1.4 Scope of Study

The scopes of study addressed in this research are:

i. To study a mathematical modeling on vinyl acetate monomer process
ii. To simulate vinyl acetate monomer process for nominal condition and PI control
iii. Analyses of dynamic response of the process
iv. Determine the effect of set-point tracking and disturbance rejection to the response
v. Development of transfer function
vi. Implementation of Model Predictive Control
vii. Model testing for vinyl acetate monomer process.

## 1.5 Layout of the Thesis

Chapter 2 begins with the introduction of vinyl acetate. This is followed by the description and explanation about MPC.

Chapter 3 start with explanation on process description of vinyl acetate monomer process, followed by data collection and mathematical modeling of the process. Then steady state simulation was done and the result was compared with actual plant data. After that analysis of dynamic behavior of the process was done and sensitivity analysis was done and effect of set point tracking and disturbance rejection was identified.

Chapter 4 commences with development of transfer functions. Then those transfer function is used in MPC implementation. In this chapter, MPC is tuned to get the optimum value of prediction horizon, P and control horizon, M. Sensitivity analysis was made to the MPC using some disturbances. The performance of MPC and disturbance rejection capability is examined in this chapter.

Chapter 5 introduced Multi Input Single Output (MISO) process. Sensitivity analysis of MPC in MISO process has been carried out using two disturbances that are random number and band limited white noise. The MPC performance for MISO process has been studied and the results have been displayed. Then, the MPC performance was discussed. This thesis is concluding by Chapter 6 where the conclusions drawn from the study as well as some recommendations for future works are presented.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Vinyl Acetate

Vinyl acetate is the organic compound with the formula $CH_3COOCH=CH_2$. This colorless liquid with a pungent odor is the precursor to an important polymer polyvinyl acetate. Vinyl acetate monomer (VAM) is an essential chemical building block used in a wide variety of industrial and consumer products. VAM is a key ingredient in emulsion polymers, resins, and intermediates used in paints, adhesives, coatings, textiles, wire and cable polyethylene compounds, laminated safety glass, packaging, automotive plastic fuel tanks, and acrylic fibers.

VAM is flammable and reactive, but can be stored, transported and handled safely if the compound's properties are understood. VAM is not considered to be highly toxic, but exposure can irritate the respiratory tract, eyes and skin. Skin contact may cause sensitization and an allergic skin reaction in a small proportion of individuals. Animal studies found that long-term exposure to VAM can cause a carcinogenic response.

## 2.2 Model Predictive Control

Over the past decade, Model Predictive Control (MPC) has established itself in industry as an important form of advanced control due to its advantages over traditional controllers. MPC displays improved performance because the process

model allows current computations to consider future dynamic events. For example, this provides benefit when controlling processes with large dead times or non-minimum phase behavior. MPC allows for the incorporation of hard and soft constraints directly in the objective function. In addition, the algorithm provides a convenient architecture for handling multivariable control due to the superposition of linear models within the controller. Figure 2.1 and Figure 2.2 illustrates the basic structure and algorithm schematic of MPC.

MPC refers to a family of control algorithms that employ an explicit model to predict the future behavior of the process over an extended prediction horizon. These algorithms are formulated as a performance objective function, which is defined as a combination of set point tracking performance and control effort. This objective function is minimized by computing a profile of controller output moves over a control horizon. The first controller output move is implemented, and then the entire procedure is repeated at the next sampling instance. Figure 2.3 illustrates the 'moving horizon' technique used in model predictive control.

MPC presents some advantages such as:

1) The process model captures the dynamic and static interactions between input, output and disturbance variables
2) Constraints on inputs and outputs are considered in a systematic manner
3) The control calculation can be coordinated with the calculation of optimum set points
4) Accurate model prediction can provide early warnings of potential problems

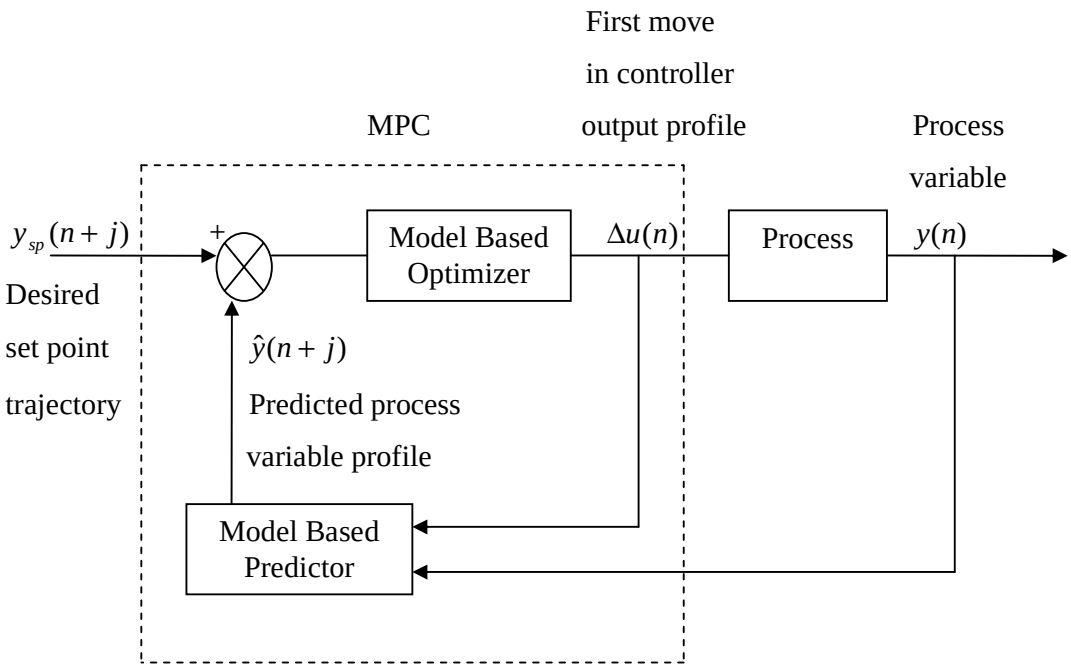


Figure 2.1: Basic structure of MPC strategy

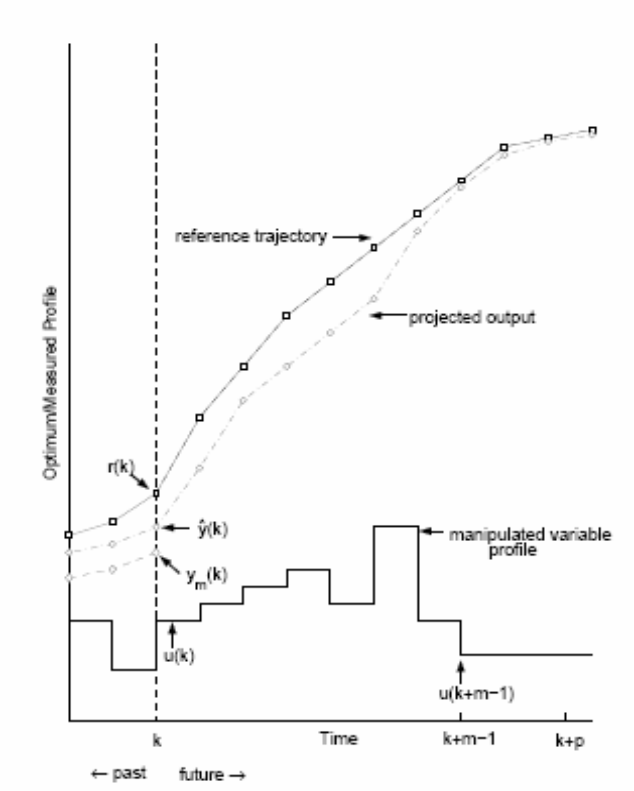


Figure 2.2: MPC Algorithm Schematic (Ogunnaike and Ray, 1994)

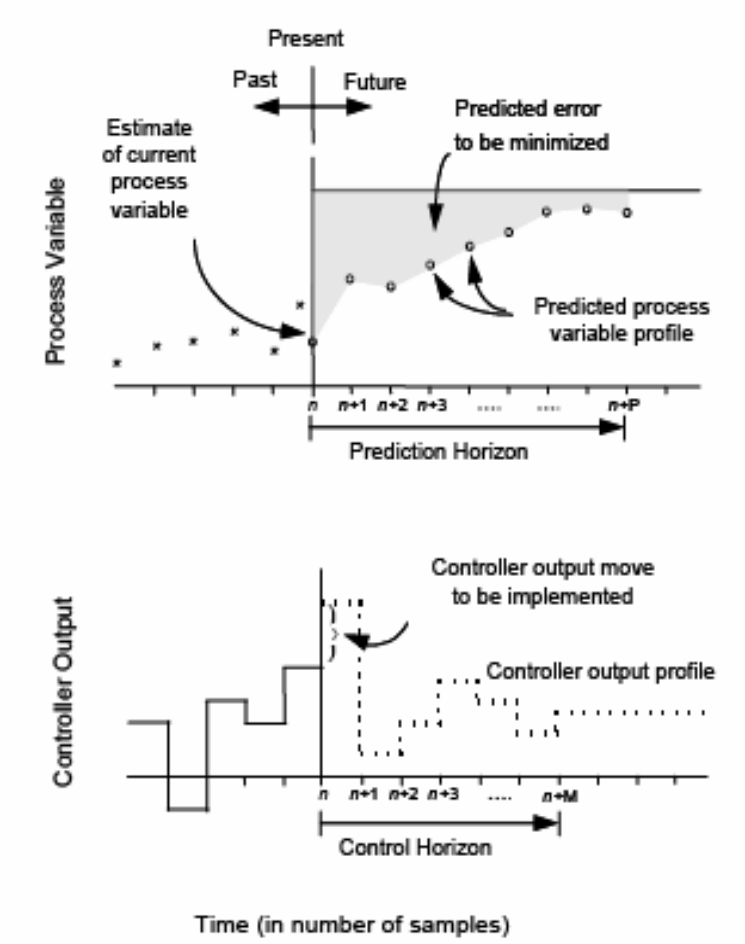


Figure 2.3: The 'moving horizon' concept of MPC (Dougherty and Cooper, 2003)

### 2.2.1 Development of MPC

This section presents a development history of industrial MPC technology. Since the advent of MPC, various model predictive controllers have evolved to address an array of control issues. Some early forms of these controllers use actual plant measurements to obtain the internal process model. The initial IDCOM and DMC algorithms represent the first generation of MPC technology. They had an enormous impact on industrial process control and served to define the industrial MPC paradigm.

Model Algorithmic Control (MAC) also known as Identification and Command (IDCOM) was developed by Richalet *et al.* (1978) and employ an impulse response model. They described their approach as model predictive heuristic control (MPHC). The distinguishing features of the IDCOM approach are:

1) impulse response model for the plant, linear in inputs or internal variables;
2) quadratic performance objective over a finite prediction horizon;
3) future plant output behavior specified by a reference trajectory;
4) input and output constraints included in the formulation;
5) optimal inputs computed using a heuristic iterative algorithm, interpreted as the dual of identification.

Richalet *et al*. (1978) chose an input–output representation of the process in which the process inputs influence the process outputs directly. Process inputs are divided into manipulated variables (MVs) which the controller adjusts, and disturbance variables (DVS) which are not available for control. Process outputs are referred to as controlled variables (CVs). They chose to describe the relationship between process inputs and outputs using a discrete-time finite impulse response (FIR) model. For the single input, single output (SISO) case the FIR model looks like:

$$y\sum_{i=1}^{N} h_i u_{k+j-i} \qquad (2.1)$$

This model predicts that the output at a given time depends on a linear combination of past input values, the summation weights $h_i$ are the impulse response coefficients. The sum is truncated at the point where past inputs no longer influence the output. This representation is therefore only possible for stable plants.

Next, Dynamic Matrix Control (DMC) was introduced by Cutler and Ramaker (1980). This controller uses a step response model. Key features of the DMC control algorithm include:

1) linear step response model for the plant;
2) quadratic performance objective over a finite prediction horizon;
3) future plant output behavior specified by trying to follow the set point as closely as possible;
4) optimal inputs computed as the solution to a least squares problem

The linear step response model used by the DMC algorithm relates changes in a process output to a weighted sum of past input changes, referred to as input moves. For the SISO case the step response model looks like:

$$y_{k+j} = \sum_{i=1}^{N-1} s_i \Delta u_{k+j-i} + s_N u_{k+j-N} \tag{2.2}$$

The move weights $s_i$ are the step response coefficients. Multiple outputs were handled by superposition. By using the step response model one can write predicted future output changes as a linear combination of future input moves. The matrix that ties the two together is the so-called Dynamic Matrix. Using this representation allows the optimal move vector to be computed analytically as the solution to a least-squares problem. Feed forward control is readily included in this formulation by modifying the predicted future outputs.

The objective of a DMC controller is to drive the output as close to the set point as possible in a least squares sense with a penalty term on the MV moves. This results in smaller computed input moves and a less aggressive output response. As with the IDCOM reference trajectory, this technique provides a degree of robustness to model error. Move suppression factors also provide an important numerical benefit in that they can be used to directly improve the conditioning of the numerical solution.

The original IDCOM and DMC algorithms provided excellent control of unconstrained multivariable processes. However, on-line constraint handling was still somewhat ad hoc. This matter was led some modifications to the first generations of MPC. This weakness was overcome by posing an extension of DMC that employs a robust quadratic performance objective with explicit incorporation of constraints. It

was proposed by Garcia and Morshedi (1986) and is known as Quadratic Dynamic Matrix Control (QDMC). Key features of the QDMC algorithm include:

1) linear step response model for the plant;
2) quadratic performance objective over a finite prediction horizon;
3) future plant output behavior specified by trying to follow the set point as closely as possible subject to a move suppression term;
4) optimal inputs computed as the solution to a quadratic program.

The QDMC algorithm can be regarded as representing a second generation of MPC technology, comprised of algorithms which provide a systematic way to implement input and output constraints. This was accomplished by posing the MPC problem as a QP, with the solution provided by standard QP codes.

As MPC technology gained wider acceptance and problems tackled by MPC technology grew larger and more complex, control engineers implementing second generation MPC technology ran into third generation MPC. A similar extension that replaces the iterative solution technique of IDCOM with a quadratic programming algorithm gave rise to IDCOM-M. A state space implementation of MPC was also proposed as the Shell Multivariable Optimizing Control (SMOC) algorithm. In the last 10 years, increased competition and the mergers of several MPC vendors have led to significant changes in the industrial MPC landscape. The Robust Model Predictive Control (RMPC) algorithm offered by Honeywell was merged with the Profimatics PCT controller to create their current offering called Robust Model Predictive Control Technology (RMPCT). In early 1996, Aspen Technology Incorporation purchased Setpoint Incorporation and DMC Corporation. The Set point Multivariable Control Architecture (SMCA) and DMC technologies were subsequently merge to create Aspen Technology's current DMC-plus product. DMC-plus and RMPCT are representative of the fourth generation MPC technology today.

Several commercial versions of MPC are now available for both implementation on a Distributed Computer System (DCS) module or implementation

on a separate computer networked to the DCS (Qin and Badgwell, 2003). Some of these include the Adersa's Identification and Command-Hierarchical Controller (IDCOM-HIECON) and Predictive Functional Control (PFC), AspenTech's Dynamic Control Plus (DMC-plus) package, Pavilion Technologies' Process Perfecter, Honeywell's Robust Model Predictive Control Technology (RMPCT), Treiber Controls' Optimum Predictive Control (OPC) and Control Soft's Modular Multivariable Control (MMC). Their differences lie in the specifics of the architecture, implementation strategy and application platform.

### 2.2.2 Dynamic Matrix Control

Dynamic Matrix Control is the most popular MPC algorithm used in the chemical process industry today due to it's major benefit in multivariable applications. It was introduced by Cutler and Ramaker (1980). Over the past decade, DMC has been implemented on a wide range of process. A major part of DMC 's appeal in industry stems from the use of a linear finite step response model of the process and a simple quadratic performance objective function. The objective function is minimized over a prediction horizon to compute the optimal controller output moves as a least-squares problem. When DMC is employed on nonlinear chemical processes, the application of this linear model-based controller is limited to relatively small operating regions. Hence, the capabilities of DMC will degrade as the operating level moves away from its original design level of operation. To maintain the performance of the controller over a wide range of operating levels, a multiple model adaptive control (MMAC) strategy for single loop DMC has been developed.

The method of approach is to construct a set of DMC process models that span the range of expected operation. By combining the process models to form a nonlinear approximation of the plant, the true plant behavior can be approached. The

more models that are combined, the more accurate the nonlinear approximation will be.

### 2.2.3 Algorithm of Dynamic Matrix Control

DMC uses a linear finite step response model of the process to predict the process variable profile, $\hat{y}(n+j)$ over j sampling instants ahead of the current time, n:

$$\hat{y}(n+j) = \underbrace{y_0 + \sum_{i=1}^{j} a_i \Delta u(n+j-i)}_{Effect\_of\_current\_and\_future\_moves} + \underbrace{\sum_{i=j+1}^{N-1} a_i \Delta u(n++j-i)}_{Effect\_of\_past\_moves} \quad (2.3)$$

In Eq. (2.3), $y_0$ is the initial condition of the process variable, $\Delta u_i = u_i - u_{i-1}$ is the change in the controller output at the *i*th sampling instant, $a_i$ is the *i*th unit step response coefficient of the process, and N is the model horizon and represents the number of sampling intervals of past controller output moves used by DMC to predict the future process variable profile. The current and future controller output moves have not been determined and cannot be used in the computation of the predicted process variable profile. Therefore, Eq. (2.3) reduces to

$$\hat{y}(n+j) = y_0 + \sum_{i=j+1}^{N-1} (a_i \Delta u(n+j-i)) + d(n+j) \quad (2.4)$$

where the term $d(n+j)$ combines the unmeasured disturbances and the inaccuracies due to plant-model mismatch. Since future values of the disturbances are not available, $d(n+j)$ over future sampling instants is assumed to be equal to the current value of the disturbance, or

$$d(n+j) = d(n) = y(n) - y_0 - \sum_{i=1}^{N-1} (a_i \Delta u(n-i)) \quad (2.5)$$

where $y$(n) is the current process variable measurement. The goal is to compute a series of controller output moves such that

$$y_{sp}(n+j) - \hat{y}(n+j) = 0 \qquad j = 1,2,\ldots\ldots,P, \tag{2.6}$$

where P is the prediction horizon and represents the number of sampling intervals into the future over which DMC predicts the future process variable. Substituting Eq.(2.4) in Eq.(2.6) gives

$$\underbrace{y_{sp}(n+j) - y_0 - \sum_{i=j+1}^{N-1} a_i \Delta u(n+j-1) - d(n)}_{predicted\_error\_based\_on\_pased\_moves} = \underbrace{\sum_{i=1}^{j} a_i \Delta u(n+j-i)}_{Effect\_of\_current\_and\_future\_moves\_to\_be\_determined} \tag{2.7}$$

Eq. (2.7) is a system of linear equations that can be represented as a matrix equation of the form

$$\begin{bmatrix} e(n+1) \\ e(n+2) \\ e(n+3) \\ : \\ e(n+N) \\ : \\ e(n+P) \end{bmatrix} = \begin{bmatrix} a_1 0 0 \ldots\ldots\ldots\ldots\ldots\ldots 0 \\ a_2 a_1 0 \ldots\ldots\ldots\ldots\ldots 0 \\ : \\ a_M a_{M-1} a_{M-2} \ldots\ldots\ldots a_1 \\ : \\ a_P a_{P-1} a_{P-2} \ldots . a_{P-M+1} \end{bmatrix} \text{X} \begin{bmatrix} \Delta u(n) \\ \Delta u(n+1) \\ \Delta u(n+2) \\ \\ \Delta u(n+M-1) \end{bmatrix} \tag{2.8}$$

P x 1 P x M M x 1

or in a compact matrix notation as

$$\bar{e} = A\Delta\bar{u}, \tag{2.9}$$

where ē is the vector of predicted errors over the next *P* sampling instants, ***A*** is the dynamic matrix, and $\Delta\bar{u}$ is the vector of controller output moves to be determined.

An exact solution to Eq. (2.8) is not possible since the number of equations exceeds the degrees off freedom (*P* > *M*). Hence, the control objective is posed as a least squares optimization problem with a quadratic performance objective function of the form

$$\underset{\Delta\bar{u}}{Min}\ J = [\bar{e} - A\Delta\bar{u}]^T [\bar{e} - A\Delta\bar{u}] \tag{2.10}$$

In the unconstrained case, this minimization problem has a closed form solution, which represents the DMC control law:

$$\Delta \bar{u} = (A^T A + \lambda I)^{-1} A^T \bar{e} \tag{2.11}$$

Implementation of DMC with the control law in Eq. (2.11) results in excessive control action, especially when the control horizon is greater than one. Therefore, a quadratic penalty on the size of controller output moves is introduced into the DMC performance objective function. The modified objective function has the form

$$\underset{\Delta \bar{u}}{Min}\ J = [\bar{e} - A\Delta \bar{u}]^T [\bar{e} - A\Delta \bar{u}] + [\Delta \bar{u}]^T \lambda [\Delta \bar{u}] \tag{2.12}$$

where λ is the move suppression coefficient. In the unconstrained case, the modified objective function has a closed form solution of (e.g., Marchetti, Mellichamp & Seborg, 1983; Ogunnaike, 1986)

$$\Delta \bar{u} = (A^T A + \lambda I)^{-1} A^T \bar{e} \tag{2.13}$$

Adding constraints to the classical formulation given in Eq. (2.13) produces the quadratic dynamic matrix control (QDMC) (Morshedi *et al.*, 1985; Garcia & Morshedi, 1986) algorithm.

### 2.2.3 Tuning Of DMC

The foundation of this strategy lies with the formal tuning rules for non-adaptive DMC based on fitting the controller output to measured process variable dynamics at one level of operation with a FOPDT model approximation (Shridhar & Cooper, 1998). A FOPDT model has the form

$$\tau_p \frac{dy(t)}{dt} + y(t) = Kpu(t-\theta p) \quad \text{or} \quad \frac{y(s)}{u(s)} = \frac{Kpe^{-\theta ps}}{\tau_p s + 1} \tag{2.14}$$

where $K_p$ is the process gain, $\tau_p$ is the overall time constant and $\theta_p$ is the effective dead time. Specifically, $K_p$ indicates the size and direction of the process variable response to a control move, $\tau_p$ describes the speed of the response, and $\theta_p$ tells the delay prior to when the response begins. The tuning parameters for single-loop DMC include:

1. The sample time, *T*;
2. Finite prediction horizon, *P*;
3. Model horizon (process settling time in samples), *N*;
4. Control horizon (number of controller output moves that are computed), *M*; and
5. Move suppression coefficient (controller output weight),$\lambda$.

The sample time, *T*, is computed as:

$$T=Max\ (\ 0.1\tau_p,\ 0.5\theta_p) \tag{2.15}$$

This value of sample time balances the desire for a low computation load (a large *T*) with the need to properly track the evolving dynamic behavior (a small *T*). Many control computers restrict the choice of *T*, the remaining tuning rules permit values of *T* other than that computed by Eq. (2.15) to be used. The sample time and the effective dead time are used to compute the discrete dead time in integer samples as

$$k = Int\left(\frac{\theta_p}{T}\right) + 1 \tag{2.16}$$

The prediction horizon, *P*, and the model horizon, *N*, are computed as the process settling time in samples as

$$P = N = Int\left(\frac{5\tau_p}{T}\right) + k \tag{2.17}$$

Note that both *N* and *P* cannot be selected independent of the sample time, *T*. A larger *P* improves the nominal stability of the closed loop. For this reason, *P* is selected such that it includes the steady-state effect of all past controller output moves. The value of *P* calculated as the open loop settling time of the FOPDT model approximation.

In addition, it is important that *N* be equal to the open loop settling time of the process to avoid truncation error in the predicted process variable profile. Eq. (2.17) computes *N* as the settling time of the FOPDT model approximation. This value is long enough to avoid the instabilities. Then, the control horizon, *M*, must be long enough such that the results of the control actions are clearly evident in the response of the measured process variable. The tuning rule thus chooses *M* as one dead time plus one time constant, or;

$$M = Int\left(\frac{\tau_p}{T}\right) + k \tag{2.18}$$

Eq.(2.18) calculates *M* such that *M* x *T* is larger than the time required for the FOPDT model approximation to reach 60% of the steady state. The final step is the calculation of the move suppression coefficient, $\lambda$. Its primary role in DMC is to suppress aggressive controller actions. Shridhar (1997) and Cooper (1998) derived the move suppression coefficient based on a FOPDT model fit as

$$\lambda = \frac{M}{10}\left(\frac{3.5\tau_p}{T} + 2 - \frac{(M-1)}{2}\right)K_p{}^2 \tag{2.19}$$

Eq. (2.19) is valid for a control horizon greater than 1 ($M > 1$). When the control horizon is 1 ($M = 1$), no move suppression coefficient should be used ($\lambda = 0$).

## 2.3 Concluding Remark

This chapter firstly discussed about vinyl acetate and its usage. This is followed by explanation about MPC and its development. The MPC uses the models

and current plant measurements to calculate future moves in the independent variables that will result in operation that honors all independent and dependent variable constraints. From this chapter it is concluded that MPC is good controller due to some of its advantages. The advantages are the process model captures the dynamic and static interactions between input, output and disturbance variables, constraints on inputs and outputs are considered in a systematic manner and an accurate model prediction can provide early warnings of potential problems.

# CHAPTER 3

# PLANTWIDE SIMULATION AND CONTROL ON VINYL ACETATE MONOMER PROCESS

## 3.1 Process Description of Vinyl Acetate Monomer Process

In the Vinyl Acetate (VAC) process, there are 10 basic unit operations, which include a vaporizer, a catalytic plug flow reactor, a feed-effluent heat exchanger, a separator, a gas compressor, an absorber, a carbon dioxide ($CO_2$) removal system, a gas removal system, a tank for the liquid recycle stream, and an azeotropic distillation column with a decanter. Figure 3.1 shows the process flow sheet with locations of the manipulated variables. The numbers on the streams are the same as those given by Luyben *et al*. (1997). There are seven chemical components in the VAC process. Ethylene ($C_2H_4$), pure oxygen ($O_2$), and acetic acid (HAC) are converted into the vinyl acetate (VAC) product, and water ($H_2O$) and carbon dioxide ($CO_2$) are by-products. An inert, ethane ($C_2H_6$), enters with the fresh $C_2H_4$ feed stream.

The following reactions take place:

$$C_2H_4 + CH_3COOH + 1/2O_2 \rightarrow CH_2{=}CHOCOCH_3 + H_2O \quad (3.1)$$

$$C_2H_4 + 3O_2 \rightarrow 2CO_2 + 2H_2O \quad (3.2)$$

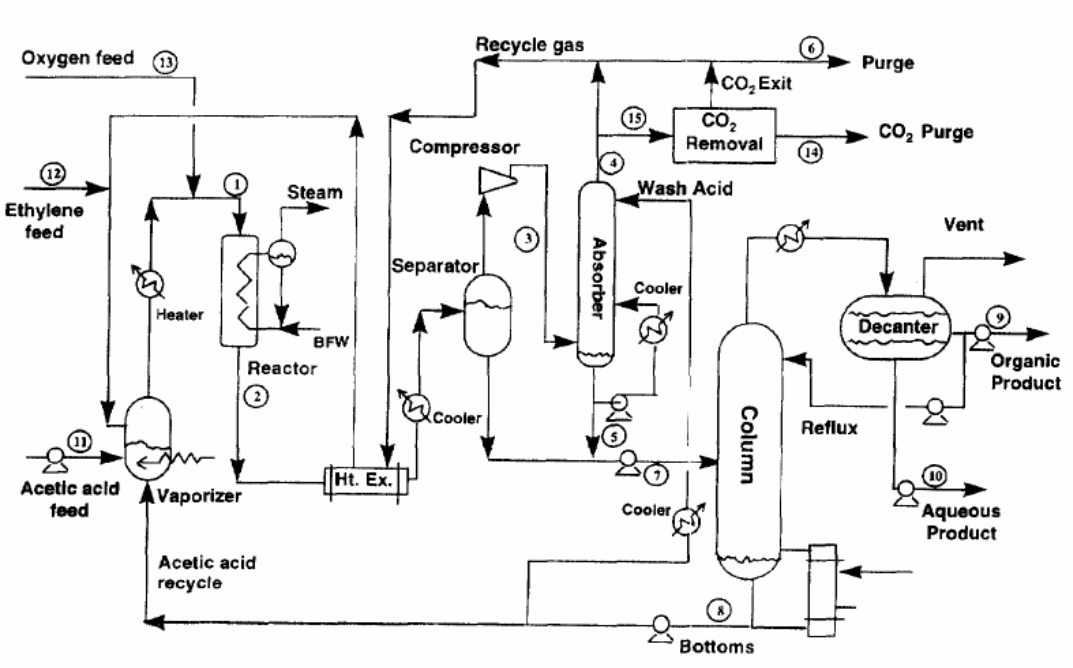


**Figure 3.1** Vinyl Acetate Monomer Process Flowsheet (Luyben *et al.*, 1997)

The exothermic reactions occur in a reactor containing tubes packed with a precious metal catalyst on silica support. Heat is removed from the reactor by generating steam on the shell side of the tubes. Water flows to the reactor from a steam drum, to which make-up water is supplied. The steam leaves the drum as saturated vapor. The reactions are irreversible and the reaction rates have an Arrhenius-type dependence on temperature.

The reactor effluent flows through a process-to-process heat exchanger, where the cold stream is the gas recycle. The rector effluent is then cooled with cooling water and the vapor (oxygen, ethylene, carbon dioxide and ethane) and liquid (vinyl acetate, water and acetic acid) are separated. The vapor stream from the separator goes to the compressor and the liquid stream from the separator becomes a part of the feed to the azeotropic distillation column. The gas from the compressor enters the bottom of the absorber, where the remaining vinyl acetate is recovered. A liquid stream from the base is recirculated through a cooler and fed to the middle of the absorber. Liquid acetic acid that has been cooled is fed into the top of the absorber to provide the final scrubbing. The liquid bottoms product from the absorber

combines with the liquid from the separator as the feed stream to the distillation column.

Part of the overhead gas exiting the absorber enters the carbon removal system. This could be one of the several standard industrial $CO_2$ removal processes. Here we simplify this system by treating is as a component separator with a certain efficiency that is a function of rate and composition. The gas stream minus carbon dioxide is split, with part going to the purge for removal of the inert ethane from the process.

The rest combines with the large recycle gas stream and goes to the feed-effluent heat exchanger. The fresh ethylene feed stream is added. The gas recycle stream, the fresh acetic acid feed, and the recycle liquid acetic acid stream enter the vaporizer, where the steam is used to vaporize the liquid. The gas stream from the vaporizer is further heated to the desired reactor inlet temperature in a trim heater using steam. Fresh oxygen is added to the gas stream from the vaporizer just prior to the reactor to keep the oxygen composition in the gas recycle loop outside the explosively region.

The azeotropic distillation column separates the vinyl acetate and water from the unconverted acetic acid. The overhead product is condensed with cooling water and liquid goes to decanter, where the vinyl acetate and water phases separate. The organic and aqueous products are sending for further refining to another distillation section. Here the additional separation steps required to produce vinyl acetate of sufficient purity is ignore because there is no recycle from the refining train back to the reaction loop. The bottom product from the distillation column contains acetic acid, which recycle back to the vaporizer along with fresh make-up acetic acid. Part of this bottoms stream is the wash acid used in the absorber after being cooled.

## 3.2 Data Collection

The collection of plant data will give a general understanding of the process behavior as well as its dynamics. The data collection help in identifying the variables, and its relationship to other variables, approximate correlations and dynamic characteristics such as dead time and time delays. DCS data is collected from the previous study by Mc Avoy *et al*. (1998).

## 3.3 Mathematical Modeling of Vinyl Acetate Monomer Process

This section discusses design assumptions, equipment data, and modeling formulations for each unit operation. In this section, the simulation model used for each major unit is discussed in detail after a brief discussion of the thermodynamic and physical property data. For each unit, the state and manipulated variables are identified.

1) Physical Properties of the Pure Component

In the MATLAB model, the vapor liquid equilibrium (VLE) calculations are performed assuming an ideal vapor phase and a standard Wilson liquid activity coefficient model. The Wilson parameters and molar volumes are listed in Table 3.1, and they are obtained directly from the TMODS model. The pure component physical property data is from Luyben *et al*. (1997) and it is listed in Table 3.2. The component vapor pressures are calculated using the Antoine equation and Antoine coefficients are get from Luyben *et al*. (1997) and it is listed in Table 3.3.

**Table 3.1:** Wilson parameters $a_{ij}$ and molar volumes $V_i$

| $a_{ij}$ | VAC | $H_2O$ | HAC | $V_i$ (ml/mol) |
|---|---|---|---|---|
| VAC | 0 | 1384.6 | -136.1 | 93.1 |
| $H_2O$ | 2266.4 | 0 | 670.7 | 18.07 |
| HAC | 726.7 | 230.6 | 0 | 57.54 |

The heat capacity expressions use have the following temperature dependence:

$$c_p = a + bt \tag{3.3}$$

where $c_p$ is in cal/g$^o$C and $t$ is the temperature in $^o$C.

**Table 3.2:** Pure component physical properties

| Component | Molecular weight | Specific gravity | Latent heat (cal/mol) | $c_p$ Liquid (a-b) cal/g$^o$C | $c_p$ Vapor (a-b) cal/g$^o$C |
|---|---|---|---|---|---|
| $O_2$ | 32 | 0.5 | 2300 | 0.3-0 | 0.218-0.0001 |
| $CO_2$ | 44.01 | 1.18 | 2429 | 0.6-0 | 0.23-0 |
| $C_2H_4$ | 28.05 | 0.57 | 1260 | 0.6-0 | 0.37-0.0007 |
| $C_2H_6$ | 30.05 | 0.57 | 1260 | 0.6-0 | 0.37-0.0007 |
| VAC | 86.09 | 0.85 | 8600 | 0.44-0.0011 | 0.29-0.0006 |
| $H_2O$ | 18.02 | 1 | 10684 | 0.99-0.0002 | 0.56-0.0016 |
| HAC | 60.05 | 0.98 | 5486 | 0.46-0.0012 | 0.52-0.0007 |

Component vapor pressure $P^s$ in psia are calculated using Antoine equation, using Antoine coefficients listed in Table 3.3.

$$\ln P^s = A + B/(t + C) \tag{3.4}$$

where $t$ is the temperature in $^o$C

**Table 3.3:** Component vapor pressure Antoine coefficient

| Component | A | B | C |
|---|---|---|---|
| $O_2$ | 9.2 | 0 | 273 |
| $CO_2$ | 7.937 | 0 | 273 |
| $C_2H_4$ | 9.497 | -313 | 273 |
| $C_2H_6$ | 9.497 | -313 | 273 |
| VAC | 12.6564 | -2984.45 | 226.66 |
| $H_2O$ | 14.6394 | -3984.92 | 233.426 |
| HAC | 14.5236 | -4457.83 | 258.45 |

2) The Vaporizer

The vaporizer is implemented as a well-mixed system with seven components. It has a gas input stream ($F_1$), which is a mixture of the $C_2H_4$ feed stream and the absorber vapor effluent stream. It also has a liquid input stream ($F_2$), which comes from the HAC tank. There are 8 state variables in the vaporizer, including the liquid level, the mole fractions of $O_2$, $CO_2$, $C_2H_4$, VAC, $H_2O$, and HAC components in the liquid, and the liquid temperature. The liquid level is defined by the ratio of the liquid holdup volume over the total working volume. Since the dynamics of the vapor phase are ignored, total mass, component and an energy balance are used to calculate the dynamics in the liquid as:

$$\rho_L^{VAP}\dot{V}_L^{VAP} = F_1^{VAP}MW_1^{VAP} + F_2^{VAP}MW_2^{VAP} - F_V^{VAP}MW_V^{VAP} \tag{3.5}$$

$$M_L^{VAP}\dot{x}_{L,i}^{VAP} = F_1^{VAP}(X_{1,i}^{VAP} - x_{L,i}^{VAP}) + F_2^{VAP}(x_{2,i}^{VAP} - x_{L,i}^{VAP}) - F_V^{VAP}(y_{V,i}^{VAP} - x_{L,i}^{VAP}) \tag{3.6}$$

$$Cp_L^{VAP}M_L^{VAP}\dot{T}_L^{VAP} = F_1^{VAP}(h_1^{VAP} - h_L^{VAP}) + F_2^{VAP}(h_2^{VAP} - h_L^{VAP}) - F_V^{VAP}(H_V^{VAP} - h_L^{VAP}) + Q^{VAP} \tag{3.7}$$

Vapor liquid equilibrium (VLE) is assumed in the vaporizer, and as a result, the vaporizer pressure and the vapor compositions are determined by a bubble point calculation. Two manipulated variables ($Q^{VAP}$ and $F_V^{VAP}$) are available in the vaporizer. In the base operation, the liquid holdup, $V_L^{VAP}$, is 2.8 $m^3$, which is 70% of

the working level volume. The vaporizer is followed by a heater, and the heater duty is a manipulated variable. In the base operation, the heater exit temperature is specified to be 150 $^{o}$C.

3) Catalytic Plug Flow Reactor

The reactor is implemented as a distributed system with ten sections in the axial direction. Two irreversible exothermic reactions, given by Eq.3.1 and 3.2, take place. In the MATLAB model, the following assumptions are made for the purpose of model simplification:

- Plug flow is assumed so that there are no radial gradients in velocity, concentration, or temperature. Diffusion occurring in the axial direction is considered negligible compared to the bulk flow. Potential and kinetic energy and work are considered negligible in the energy balance calculation.

- It is assumed that the mass and heat transfer between the fluid and catalyst are very fast and therefore the concentrations and temperatures in the two phases are always equal.

- Pressure drop is assumed linear along the length of a tube, and it is time-independent. Eqn.3.8 is used to calculate the pressure drop in each section:

$$\Delta P / \Delta Z = f * \rho_1^{RCT} * (\upsilon_1^{RCT})^2 \tag{3.8}$$

where $\Delta P / \Delta Z$ is the pressure drop per unit length (psia/m), $f$ is a constant friction factor, $\rho_1^{RCT}$ is the mass density of the feed stream (kg/m$^3$), $\upsilon_1^{RCT}$ is the volumetric flow rate of the feed stream (m$^3$/min).

- As stated earlier, the shell temperature is assumed uniform, and it is used as a manipulated variable in the MATLAB model. Thus, the steam drum dynamics are not modeled.

Material and energy balances on the reactor, which are based on a tubular reactor dynamic model developed by Reyes and Luyben (2001), are given by Eq.3.9 and 3.10:

$$\varepsilon \frac{\delta C_{i,j}}{\delta t} = -\frac{\delta (C_{i,j} v_i)}{\delta z} + \phi_i \rho_b (\theta_{1,j} r_{1,i} + \theta_{2,j} r_{2,i}) \tag{3.9}$$

$$(\varepsilon \sum_{k=1}^{7} C_{i,k} Cp_{i,k} + \rho_b Cp_b) \frac{\delta T_i}{\delta t} = -\frac{\delta (v_i \sum_{k=1}^{7} (C_{i,k} Cp_{i,k}) T_i)}{\delta z} - \phi_i \rho_b (r_{1,i} E_1 + r_{2,i} E_2) - Q_i^{RCT} \tag{3.10}$$

where index *i* represents the section number and index *j* represents component *j*, $\phi_i$ is the catalyst activity in section *i*. $\theta_{1,j}, \theta_{2,j}$ are the stoichiometric coefficients for component *j* in the two reactions, $r_{1,i}, r_{2,i}$ are the reaction rates in section *i*, $E_1, E_2$ are the heats of reactions. $Q_i^{RCT}$ is the external heat flux per unit volume in section *i*, and it is calculated by $Q_i^{RCT} = UA(T_i - T_S)$, where $T_S$ is the shell temperature.

In the MATLAB model, the molar concentrations of components $O_2$, $CO_2$, $C_2H_4$, VAC, $H_2O$ and HAC and the tube temperature in each section of the reactor are state variables. Therefore totally 70 state variables are present in the reactor. The molar concentration of component $C_2H_6$ can be calculated based on the ideal gas law. Only one manipulated variable $T_S$ is available in the reactor. In the base operation, the reactor exit temperature is equal to 159.17 $^oC$.

4) Feed Effluent Heat Exchanger (FEHE)

For the purpose of plant wide control studies, it is not necessary to rigorously model the dynamics of a process-to-process heat exchanger if it doesn't dominate the process response. The inverse of the total thermal resistance, *UA*, is calculated by Eq.3.11, which shows that the effective *UA* is a function of the mass flow rates of the two streams:

$$UA = UA_0 * [(F_1^{FEHE} / F_{C\_REF})^{0.8} + (F_2^{FEHE} / F_{H\_REF})^{0.8}] / 2 \quad (3.11)$$

where $F_1^{FEHE}$ is the mass flow rate of the cold stream and $F_2^{FEHE}$ is the mass flow rate of the hot stream. There is one manipulated variable, the bypass ratio, and no state variable in the FEHE. In the base operation, the FEHE hot effluent temperature is equal to 134 $^{o}$C.

5) Separator

There are 16 state variables in the separator, including the liquid level, vapor phase pressure, mole fractions of components $O_2$, $CO_2$, $C_2H_4$, VAC, $H_2O$, and HAC, and temperatures in both phases. The ideal gas law is applied to the vapor phase. In the separator, three manipulated variables are available, the liquid exit stream flow rate, the vapor exit stream flow rate, and the cooling jacket temperature. In the base operation, the liquid holdup is 4 m$^3$, which is 50% of the working level volume. The separator pressure is 84.25 psia, and the separator liquid phase temperature is 40 $^{o}$C.

6) Compressor

In the MATLAB model, the pressure increase across the compressor is calculated by Eq.3.12 and 3.13;

$$P_{OUT}^{COM} = P_{IN}^{COM} + \Delta P \quad (3.12)$$

$$\Delta P = \gamma \rho^{COM} \quad (3.13)$$

where $\gamma$ is the compressor coefficient, $\rho^{COM}$ is the compressor inlet stream density. The exit temperature is calculated by assuming an isentropic compression. The compressor is followed by a cooler, and the cooler duty is a manipulated variable. In the base operation, the cooler exit temperature is 80 $^{o}$C.

7) Absorber

The mass transferred from the vapor phase to the liquid phase is given by Eq.3.14:

$$N_i = \min[N_{MT} * (y_i - y_{INT,i}), 0.5 * F_{v,i} * y_i] \tag{3.14}$$

where $N_i$ is the molar flow rate of component *i* (kmol/min), $N_{MT}$ is a constant mass transfer coefficient, $y_i$ is the mole fraction of component *i* in the vapor inlet stream, $y_{INT,i}$ is the mole fraction of component *i* at the gas-liquid interface, which is obtained from an equilibrium calculation using the liquid phase compositions and temperature. $F_{V,i}$ is the mole flow rate of component *i* in the inlet vapor stream. To avoid a large mass-transfer rate between the two phases, it is assumed that the largest amount of component *i* transferred between two phases is the half of the amount of component *i* in the inlet vapor stream.

The heat transferred from the vapor phase to the liquid phase is given by:

$$Q_j = Q_{MT,j} * (T_{V,j} - T_{L,j}) \tag{3.15}$$

where $Q_j$ is the heat transferred between the two phases on stage j (kcal/min), $Q_{MT,j}$ is a constant heat transfer coefficient, $T_{V,j}$ is the temperature of the vapor inlet stream, $T_{L,j}$ is the temperature of the liquid phase. During stage-to-stage calculations, total mass, component and an energy balance around the vapor phase are used to calculate the vapor exit stream flow rate, composition, and temperature. A total mass, component and an energy balance around the liquid phase, which are similar to Eq.3.5 to 3.7, are used to model the absorber dynamics. In the energy balance, the enthalpy of the material transferred between the two phases is calculated as a vapor phase enthalpy at the stage liquid temperature.

There are totally 72 state variables in the absorber, which are the liquid holdup, mole fractions of components $O_2$, $CO_2$, $C_2H_4$, VAC, $H_2O$, and HAC in the

liquid phase and liquid temperature on each stage. There are three manipulated variables, the liquid exit stream flow rate, the scrub stream flow rate, and the circulation stream flow rate. In the base operation, the liquid holdup is 0.25 $m^3$, which is 50% of the working level volume. There are two coolers, which are installed on the scrub stream and the circulation stream respectively, and the cooler duties are manipulated variables. In the base operation, the exit stream temperatures of the two coolers are 25 $^oC$.

8) $CO_2$ Removal System

There is one manipulated variable, which is the inlet stream to the $CO_2$ removal system. In the base operation, the $CO_2$ mole fraction in the gas recycle stream is 0.73%. The system efficiency is given by Eq.3.16;

$$Eff = 0.995 - 3.14\times10^{-6} * (F_{CO_2} - 6.4136) - 32.5 * (x_{co_2} - 0.01342) \quad (3.16)$$

where $F_{CO_2}$ is the inlet stream flow rate (kmol/min), and the $x_{CO_2}$ is the mole fraction of $CO_2$ in the inlet stream.

9) Gas Removal System

The gas removal system is designed to remove all the light components in the column feed stream before they enter the column. The system has two liquid inlet streams that come from the bottoms of the separator and the absorber. An ideal component separator, which can completely separate the seven components into two streams, is implemented here. The gas stream ($O_2$, $CO_2$, $C_2H_4$, $C_2H_6$) is sent back and combined with the vapor produced from the separator to form the vapor feed to the compressor. The liquid stream (VAC, $H_2O$, HAC) is the feed to the column.

10) Azeotropic Distillation Tower

The column is assumed homogeneous, and only one liquid phase is present. To reduce the system stiffness, the pressure profile in the column is assumed known. A bubble-point calculation is used to determine temperature and compositions on each stage, and then the energy balance is used to solve for the vapor flow rate from stage to stage. Since the Wilson model can't be used in the decanter due to the liquid-liquid equilibrium, the equilibrium partition coefficients, β, used in the decanter are assumed constant and independent of temperature. It is also assumed that the temperatures of the two liquid phases in the decanter are always same. There are totally 69 state variables in the distillation column. There are six manipulated variables, reflux flow rate, reboiler duty, condenser duty, organic product flow rate, aqueous product flow rate and bottom flow rate. In the base operation, the bottom liquid holdup is 2.33 $m^3$, which is 50% of the working level volume. The organic liquid holdup and the aqueous liquid holdup are 0.85 $m^3$, which are 50% of their working level volumes. In the base operation, the decanter temperature is 45.85 $^oC$.

11) HAC Tank

The HAC tank is only used to mix the liquid recycle stream and the fresh HAC feed stream. There are totally 4 state variables in the tank, which are the liquid holdup, mole fractions of VAC, and HAC in the liquid, and the liquid temperature. The flow rates of all the streams connected to the tank are manipulated variables.

### 3.3.1 Steady State Simulation

The steady state data for VAC process are obtained after a control structure similar to that developed by Luyben *et al*. (1997) is implemented. The control system used is shown in Figure 3.1. The steady-state values of manipulated variables, the control structure and controller parameters and steady-state values for measurements

are listed in Appendix A. The data involve the whole process such as separator, reactor, distillation column, scrubber and others.

Once the required equipment design parameters and thermodynamic-related properties have been set, the simulation can proceed by fixing initial conditions as can be seen in the detailed programming in Appendix B. Once initial conditions have been specified, simulation is performed until all the values in calculated streams match with those in the assumed stream. In this simulation, a sample time used is 350 minutes and it was involved the whole process of VAC.

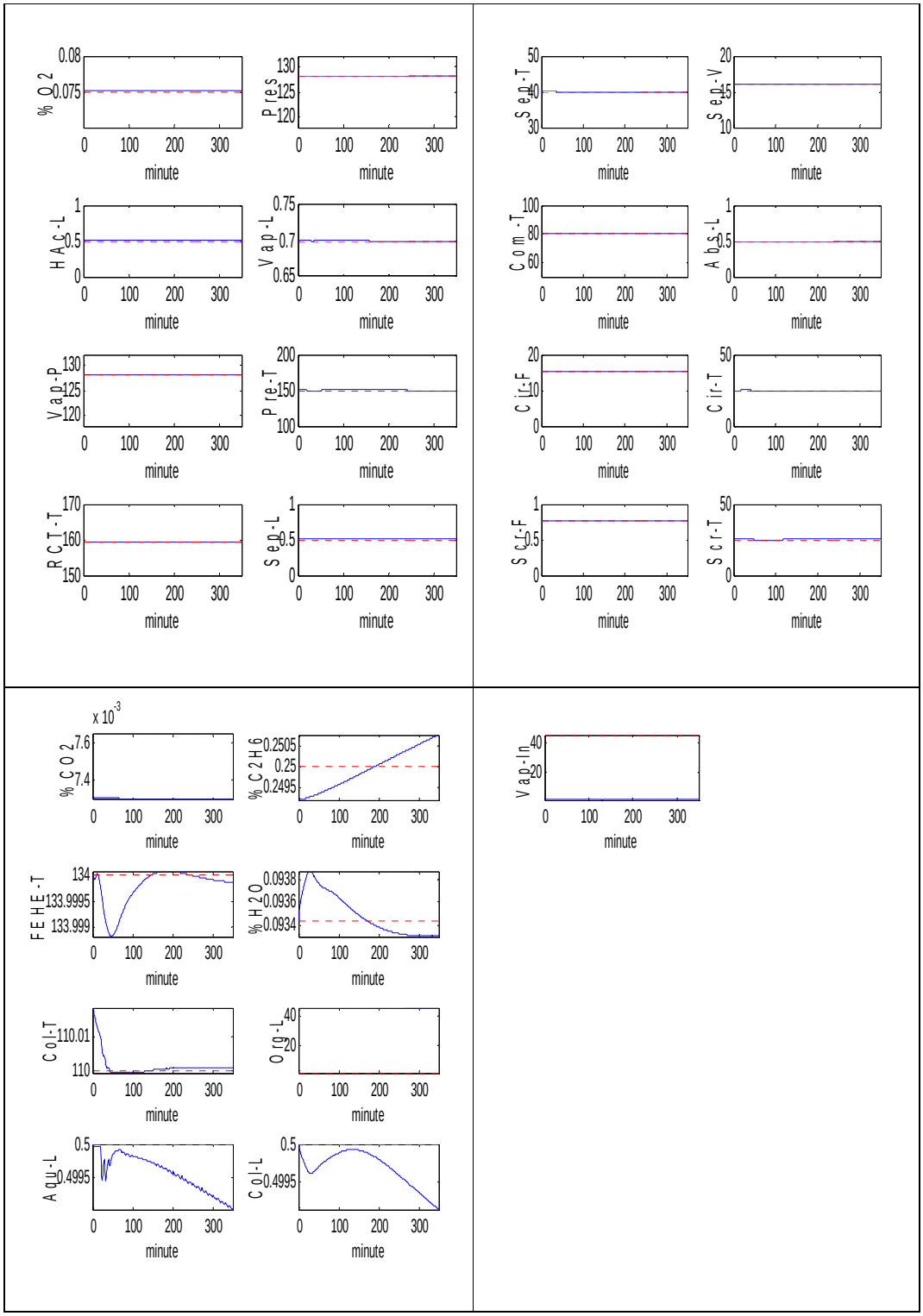


**Figure 3.2** Data generation in steady state condition

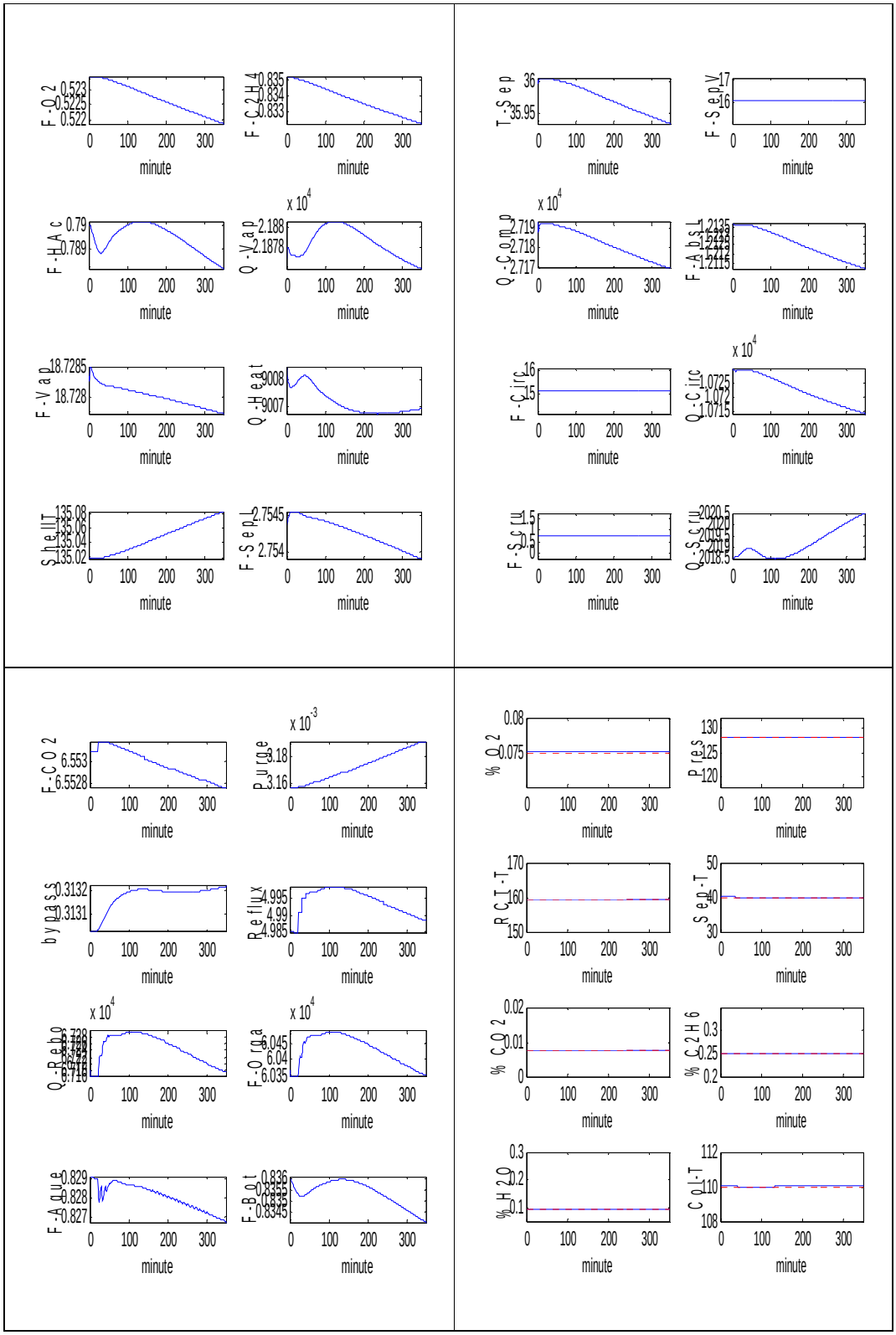


**Figure 3.3** Data generation in steady state condition

Figure 3.2 and 3.3 show the data that have been generate from the VAC process at steady state condition. The results in Figure 3.2 and 3.3 represent all controlled variables and manipulated variables involved in the vinyl acetate monomer process. In these figures, the setpoint is representing by red line while the response is represent by blue color line. The results show that the controller tracked the reference trajectory closely for all variables except for % $CO_2$ and % $C_2H_6$ in the gas recycle, % $H_2O$ in the column bottom, decanter organic and aqueous level and column bottom level.

### 3.3.2 Simulation Results

The results for steady state simulation were compared with the actual plant data to verify the reliability of the simulation. It is shown that the behavior of the actual plant could be predicted quite wee using the simulation model and the results is presented in Table 3.4.

**Table 3.4:** Comparison between actual plant data and simulation

| | Reactor Out | | Absorber Vapor Out | | Organic Product | | Aqueous Product | |
|---|---|---|---|---|---|---|---|---|
| MOLE FRACT | Plant | Simulation | Plant | Simulation | Plant | Simulation | Plant | Simulation |
| $O_2$ | 0.049 | 0.049 | 0.058 | 0.058 | | | | |
| $CO_2$ | 0.011 | 0.011 | 0.014 | 0.014 | | | | |
| $C_2H_4$ | 0.551 | 0.551 | 0.658 | 0.658 | | | | |
| $C_2H_6$ | 0.221 | 0.221 | 0.263 | 0.263 | | | | |
| VAc | 0.043 | 0.043 | 0.002 | 0.002 | 0.95 | 0.95 | 0.002 | 0.002 |
| $H_2O$ | 0.055 | 0.055 | 0.001 | 0.001 | 0.05 | 0.05 | 0.998 | 0.998 |
| HAc | 0.07 | 0.07 | 0.004 | 0.004 | 370[a] | 370[a] | 370[a] | 370[a] |
| | | | | | | | | |
| Reactor Feed Temperature($^oC$) | | | 148.5 | | | 148.5 | | |
| Absorber Feed Temperature($^oC$) | | | 80 | | | 80 | | |
| Reactor Feed Pressure(psia) | | | 128 | | | 128 | | |

[a]moles/million

In general, the values of actual plant data are same as simulation results. It is therefore concluded that the simulation model is able to represent the steady-state condition of the process reasonably.

## 3.4 Analysis of Dynamic Response

The sensitivity and dynamic response analysis of vinyl acetate monomer process were carried out to identify the effect of set point tracking and disturbance rejection. Selected process inputs were changed and the corresponding process outputs were monitored.

The model developed by Mc Avoy *et al.* (1998) in MATLAB has been used to generate a data for this research. In that research, it provides nine different conditions for the Vinyl Acetate process, that is:

A. No disturbance
B. Setpoint of the reactor outlet temperature decreases 8$^{o}$C (from 159$^{o}$C to 151$^{o}$C)
C. Setpoint of the reactor outlet temperature increases 6$^{o}$C (from 159$^{o}$C to 165$^{o}$C)
D. Setpoint of the $H_2O$ composition in the column bottom increases 9% (from 9% to 18%)
E. The vaporizer liqiud inlet flowrate increases 0.44 kmol/min (from 2.2 kmol/min to 2.64 kmol/min)
F. HAc fresh feed stream lost for 5 minutes
G. $O_2$ fresh feed stream lost for 5 minutes
H. $C_2H_6$ composition changes from 0.001 to 0.003 in the $C_2H_4$ fresh feed stream
I. column feed lost for 5 minutes

The analysis was focused on separator unit. The parameters involved are level and temperature. The set point for the separator level and temperature has been changes to study the effect of set point changes. The different conditions for the process provided by previous research give the advantages to study the disturbance rejection.

### 3.4.1 Effect of Set Point Changes

As mention before, this study was focus on separator unit in vinyl acetate monomer process. Table 3.5 below show controller parameters for separator. Because of separator vapor flowrate is fixed, it was exclude and this study involved the separator level and temperature only.

**Table 3.5:** Controller Parameter for Separator

| Loop | Controlled Variable | Manipulated Variable | C.V. Value | Type |
|---|---|---|---|---|
| 1 | Separator Level | Separator Liquid Exit Valve | 50% (0-100) | PI |
| 2 | Separator Temperature | Separator Coolant Valve | 40$^{o}$C (0-80) | PI |
| 3 | Separator Vapor Flowrate | Separator Vapor Exit Valve | | Fixed |

Closed loop sensitivity analysis on the system was carried out by changing the setpoint value of separator level and temperature by $\pm$10% and $\pm$20%. This closed loop system behavior for set point changes referred as the servomechanism. It was assumed that there is no disturbance occurs. Table 3.6 show the set point value of separator level and temperature after changes.

**Table 3.6:** Value of separator level and temperature after changes

| No | Parameter | New Setpoint |
|---|---|---|
| 1 | Separator Level (%) | 45(-10%)<br>40(-20%)<br>55(+10%)<br>60(+20%) |
| 2 | Separator Temperature($^{o}$C) | 36(-10%)<br>32(-20%)<br>44(+10%)<br>48(+20%) |

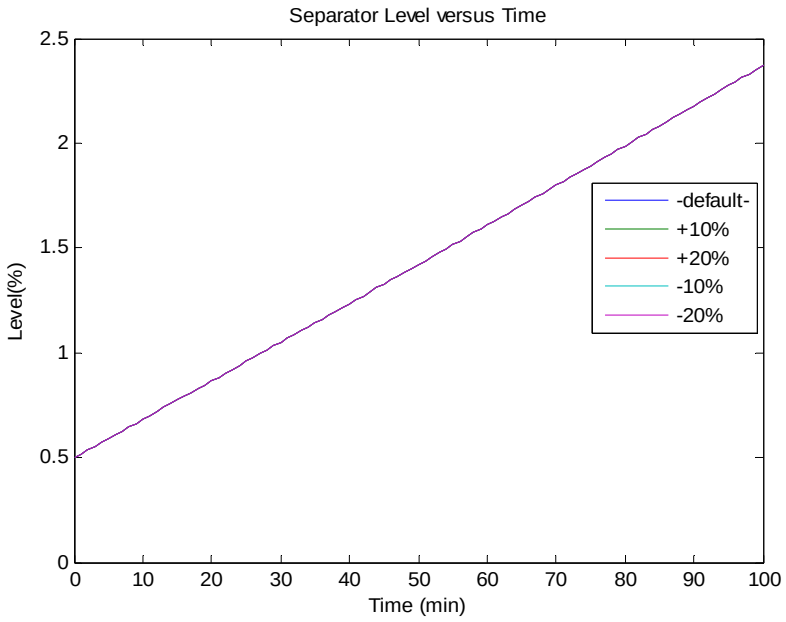


**Figure 3.4** Effect on ±10% and ±20% of separator level

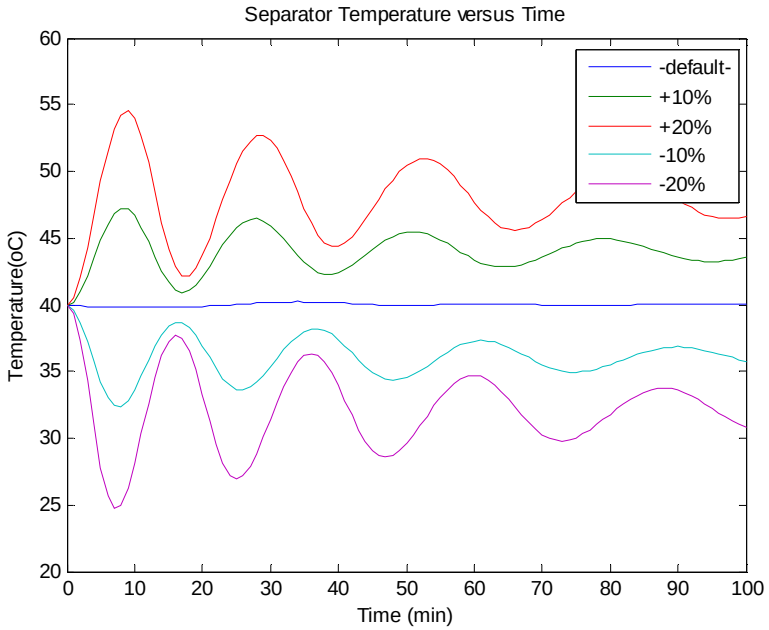


**Figure 3.5** Effect on ±10% and ±20% of separator temperature

Figure 3.4 represent the effect of ±10% and ±20% fluctuation of the separator level. The graph shows there is one line only. It is because all of the lines are overlap. The results show that the separator level is proportional with time. For Figure 3.5, the

graphs are oscillating for some time and the oscillation is smaller before reach to the setpoint. The graph showing that the controller tends to response on the changes but it the response is very slow.

### 3.4.2 Controller Performance for Disturbance Rejection

Disturbance rejection is the capability of a control system to bring the process back to normal condition from process upsets. The ability of a control system to reject a disturbance depends on the gain of each disturbance. This research is considering only conditions B till D (refer section 3.4). The detailed programming is shown in Appendix B.

In order to test the performance of the controller for disturbance rejection, a change in the reactor outlet temperature and % $H_2O$ composition in column bottom as condition B, C and D was made. The same sample time, manipulated variable and set-point tracking were used in this simulation. The results are shown in Figure 3.6 until 3.8.

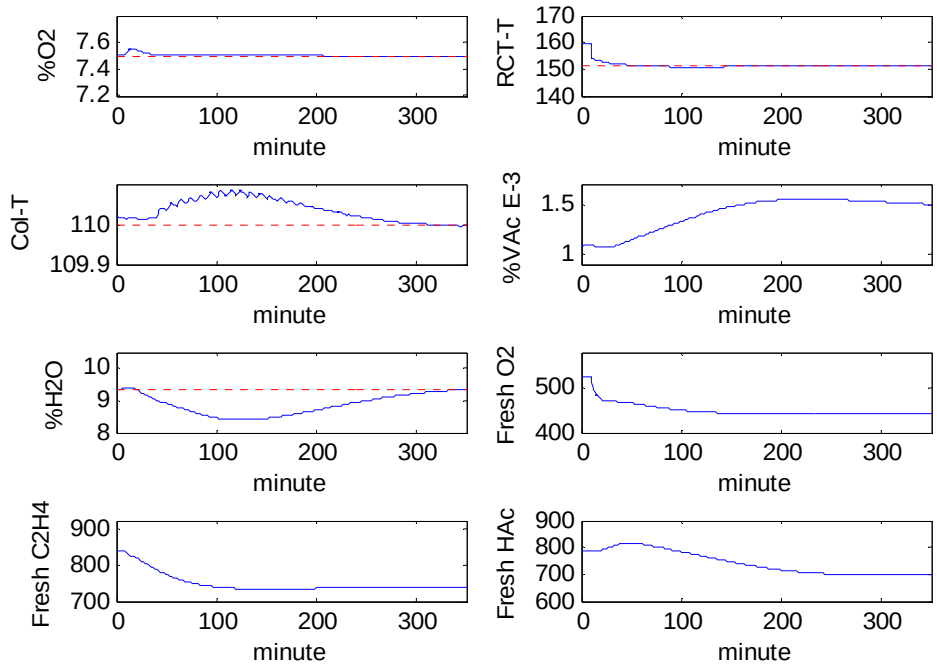


**Figure 3.6** Dynamic Response in Condition B

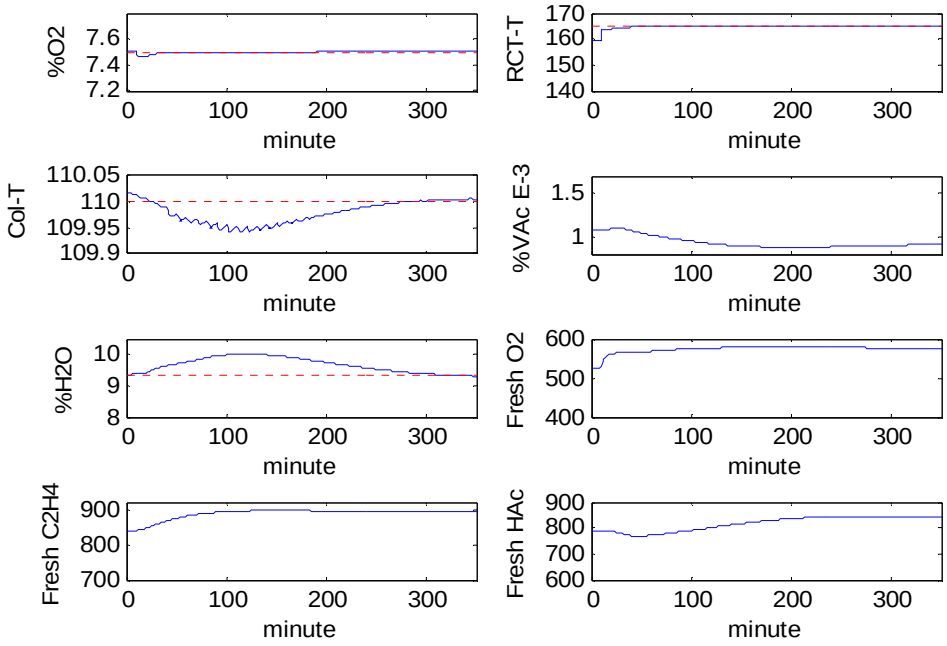


**Figure 3.7** Dynamic Response in Condition C

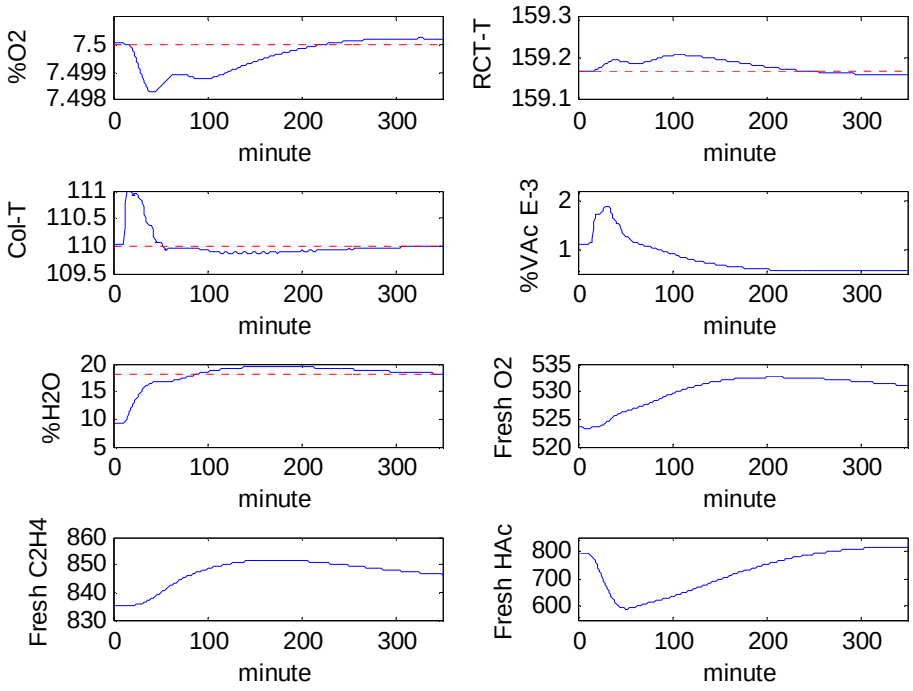


**Figure 3.8** Dynamic Response in Condition D

Figure 3.6 until 3.8 shows the response of the variable in condition B, C and D where disturbance present. A setpoint change in reactor outlet temperature, from 159$^{o}$C to 151$^{o}$C is shown in Figure 3.6. As can be seen the reactor temperature response is very rapid and the new steady state is quickly achieved. Tight control of the mole % $H_2O$ is achieved here. The oscillations in the tray 5 temperature are caused by the fact that the reflux is manipulated by a signal from the analyzer that measures the % $H_2O$ in the bottoms. Other variables affected are % VAC, fresh $O_2$, fresh HAC and fresh $C_2H_4$. Because of the interacting process, when the outlet temperature is decrease, the reactor temperature also decreases as well as fresh $O_2$, HAC and $C_2H_4$ so that the % VAC product also decreases.

A setpoint change in reactor outlet temperature, from 159$^{o}$C to 165$^{o}$C is shown on Figure 3.7. Again the response of the reactor temperature is very fast and the new steady state is achieved very quickly. The variables affected are same as variables affected in condition B. But for this condition, when outlet temperature

increase, the reactor temperature also increase as well as fresh $C_2H_4$ and $O_2$ and so that the % VAC and $H_2O$ produced also increase.

Figure 3.8 shows a setpoint change of the $H_2O$ composition in the column bottom, from 9% to 18%. As can be seen, this setpoint change produces a very slow transient, which takes over 5 hours to die out. From the graph, the variables that mostly affected are $\%O_2$ and % VAC. These variables relate each other. When % $H_2O$ in the column bottom is increasing, % VAC is decreasing. For % $O_2$, it is decreasing for 100 minutes earlier and then it is increasing and takes about 5 hours to stable.

## 3.5 Concluding Remark

The result presented here has pointed out some important conclusion. Dynamic modeling and simulation of vinyl acetate monomer process has been successfully accomplished using MATLAB 7.1. The sensitivity and dynamic response analysis of vinyl acetate monomer process has been carried out and effect of set point tracking and disturbance rejection was identified. The interacting process also analyzed. The results on the analyses carried out on the system have been displayed. Based on the results, it is concluded that the VAC model discussed here essentially captures the dynamics. It is also concluded that in the interacting process, the variables and units are related each other, even small changes in a units can affect the other units.

# CHAPTER 4

# IMPLEMENTATION OF MPC ON SEPARATOR

## 4.1 Introduction

Transfer function is an algebraic expression for the dynamic relation between selected input and output of the process model. It is defined so as to be independent of the initial conditions and of the particular choice of forcing function. From the simulation, the data generated need to convert to simple form in order to use in Model Predictive Control. Transfer function is one of the simple forms. It was used in developing Model Predictive Control model. MPC was implemented in this research using MATLAB 7.1/ Simulink.

## 4.2 Transfer Function Development

The development of transfer function for the process is specific to the separator level and temperature. There are several ways to develop the transfer function. In this study, the transfer function is developed using step test method. The method to develop the transfer function is referring to Seborg *et al*. (2004). The Vinyl Acetate monomer process is complete and potentially non-linear. Hence, it is desirable to model the dynamics of these processes near the current operating point by simpler models that is the first order plus dead time (FOPDT) which as shown in equation 4.1.

$$G(s) = \frac{Ke^{-\theta s}}{\tau s + 1} \tag{4.1}$$

where , $\mathrm{K} = \mathrm{Gain} = \frac{\Delta y}{\Delta x} = \frac{\text{Output Changes}}{\text{Input Changes}}$

$\theta = \text{Dead time} = 1.3t_1 - 0.29t_2$

$\tau$ = Reset = Value of t which the response is 63.2% complete

In developing transfer function, the Feed Effluent Heat Exchanger (FEHE) hot temperature is considered as input for both separator level and temperature. The initial value of FEHE hot temperature which is 134°C is increased by 10% and become 147.4°C. Figure 4.1 shows the step input for the process while Figure 4.2 and 4.3 show the step output of the separator level and separator temperature.

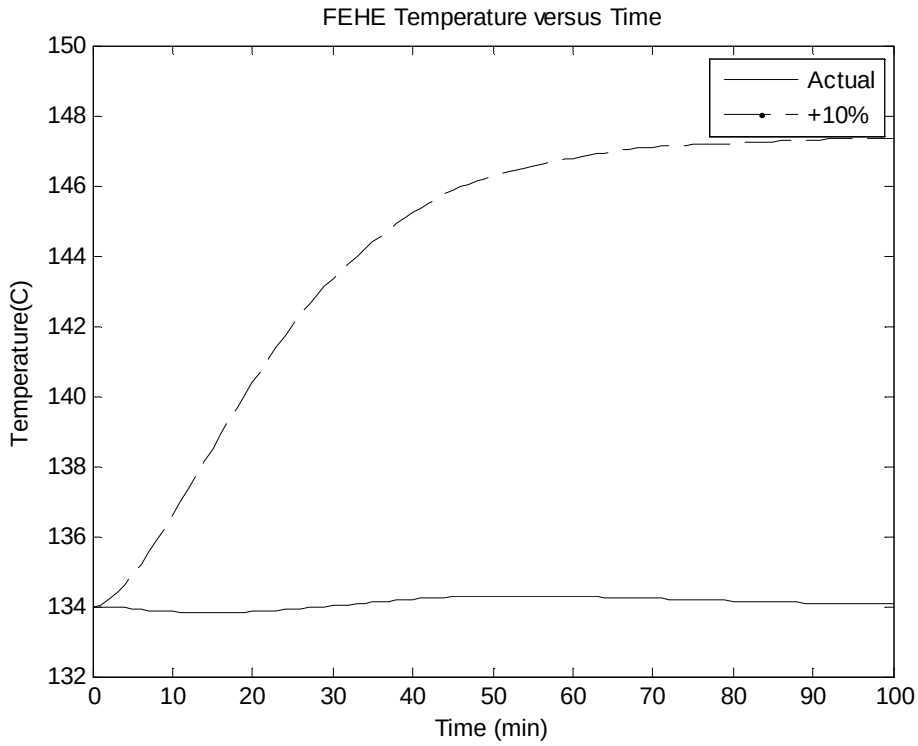


**Figure 4.1** Input changes of FEHE hot temperature

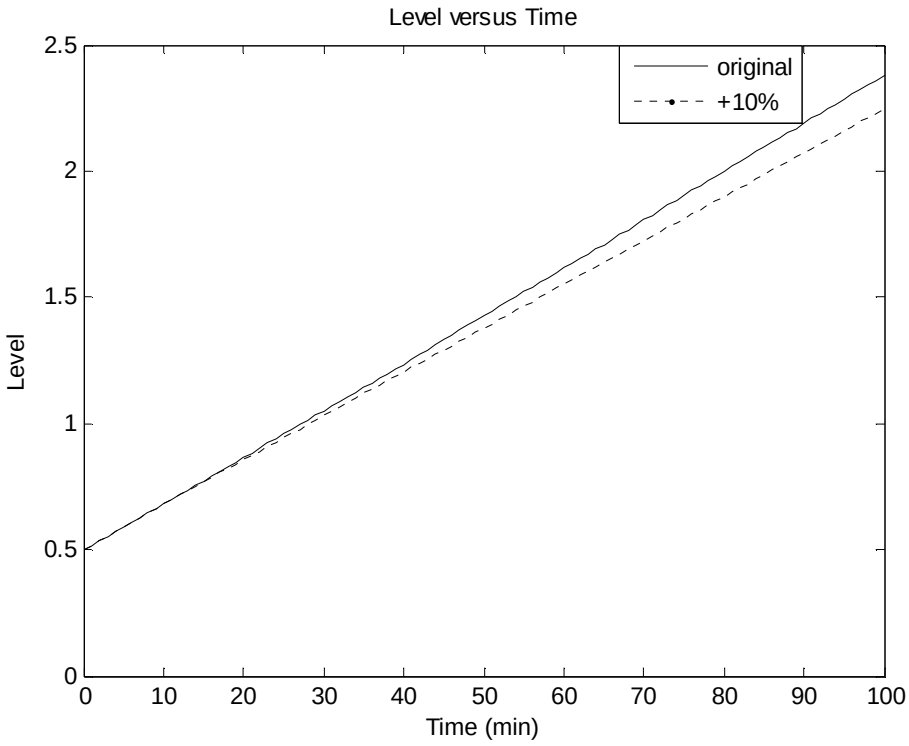


**Figure 4.2** Output changes for separator level

For the separator level,

$$K = \frac{\text{Output Changes}}{\text{Input Changes}} = \frac{2.36 - 0.5}{147.4 - 134} = 0.1388$$

$\tau = 60 \text{ min}$

$\theta = 10 \text{ min}$

Then the transfer function is

$$G(s) = \frac{0.1388e^{10s}}{60s + 1}$$

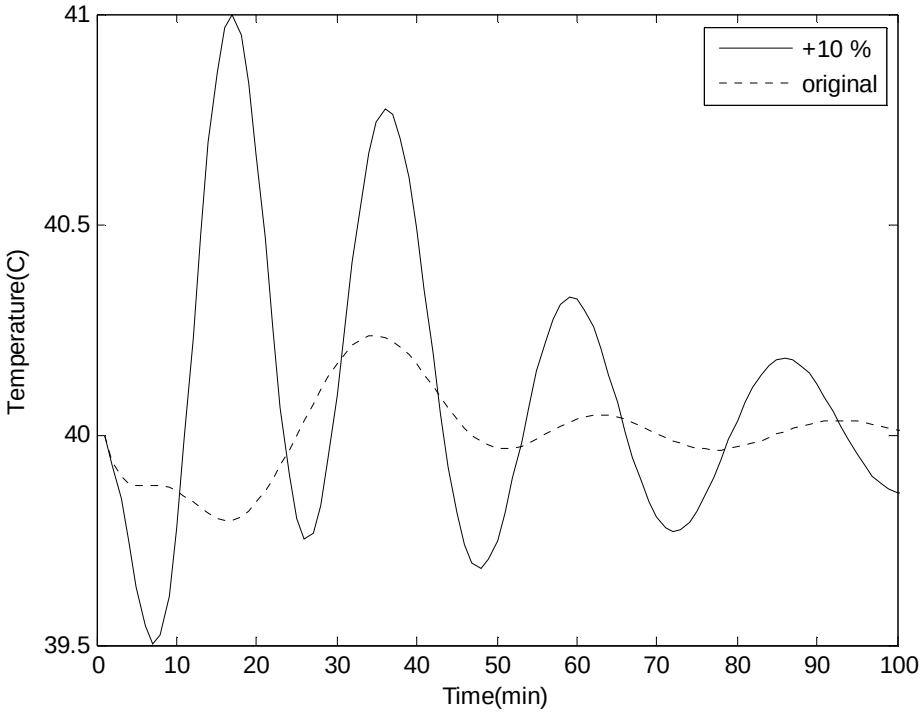


**Figure 4.3** Output changes for separator temperature

For the separator temperature,

$$K = \frac{\text{Output Changes}}{\text{Input Changes}} = \frac{41 - 39.8}{147.4 - 134} = 0.08955$$

$\tau = 30$ min

$\theta = 10$ min

Then the transfer function is

$$G(s) = \frac{0.08955e^{10s}}{30s + 1}$$

## 4.3 Implementation of MPC

The transfer function that was developed before is used in MPC design. In implementing the MPC using Matlab 7.1/Simulink, 3 models were developed:

1) The separator level closed loop.
2) The separator temperature closed loop.
3) The interacting closed loop for both transfer function

The design of MPC involved development of three parts:

1) Open loop for the system as shown in Figure 4.4
2) Closed loop model as shown in Figure 4.5
3) The m.files that combine both of the models that would be attached in Appendix C

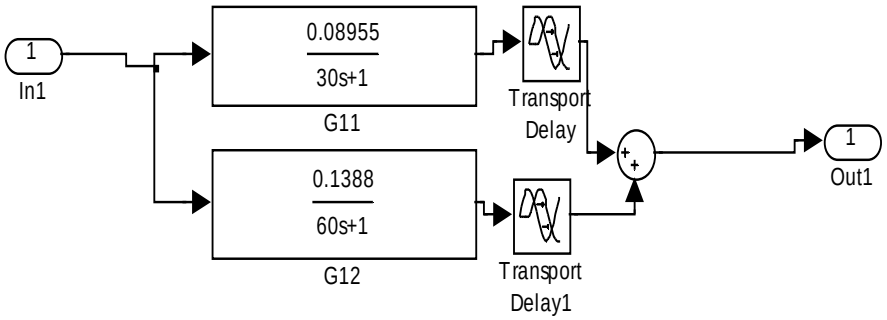


**Figure 4.4** Open loop model in MATLAB 7.1/ Simulink

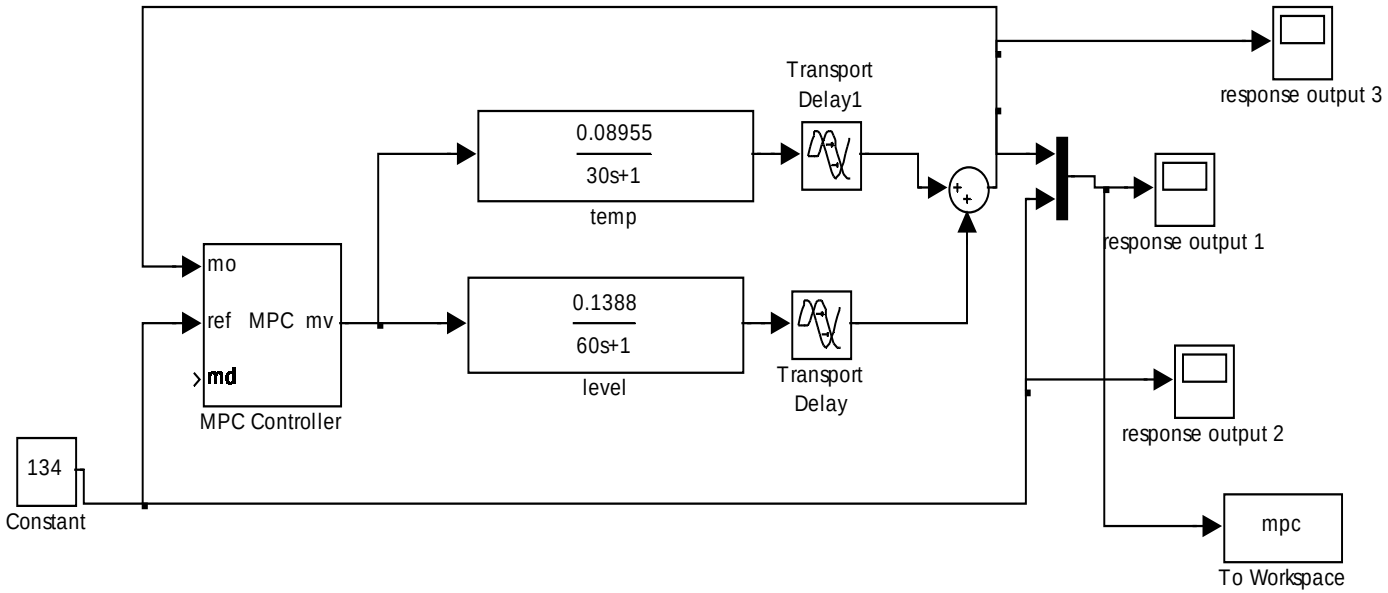


**Figure 4.5** MPC model in MATLAB 7.1/Simulink

## 4.4 Tuning The MPC

In the tuning process, the process has been tested with three types of sources that is constant, random number and band limited white noise. This is done to see the performance of the MPC whether it can adapt or not to the disturbance presence. The difference between band limited noise number and the random number is it produces output at a specific sample rate, which is related to the correlation time of the noise. These can provide the real situation to the system.

Basically, in order to determine the best control loop, the behavior of the graph is evaluated in term of stability, elimination of offset and ability to capture the set point changes and disturbance rejection. The process that stable fastest, can minimize offset to the smallest value, able to reach to the initial set point and can adapt to the disturbance presence is considered as the best process.

Theoretically, the non-adaptive DMC tuning strategy by Dougherty and Cooper (2002) have been implemented in this research. This tuning process is involved tuning the value of the prediction horizon, P and control horizon, M. The process usually trying to optimize over P sampling period. Generally, the value of P is larger than M. It is because when the value of P is bigger, the system can take corrective action immediately. After the tuning process, the optimum value of P and M is determined.

### 4.4.1 Tuning For Interacting Control Loop

The interacting control loop is combination of both level and temperature transfer function. Tuning for interacting control loop, multi input single output (MISO) is made using some values of P and M. There are also some disturbance supplied in this process which is random number and band limited white noise. The values of P and M and the response performance is summarizing in the Table 4.1 below:

**Table 4.1:** The tuning values of P and M for interacting process

| TRIAL | INPUT | TUNING VALUE | | PERFORMANCE |
|---|---|---|---|---|
| | | P | M | |
| 1 | CONSTANT | 200 | 1 | Unstable |
| 2 | | **100** | **1** | **Most Stable** |
| 3 | | 50 | 1 | Stable |
| 4 | | 30 | 1 | Less stable |
| 5 | | 10 | 1 | Take long time to get the desired output |
| 1 | RANDOM NUMBER | 50 | 1 | Stable |
| 2 | | 20 | 1 | Less stable |
| 3 | | 10 | 1 | Take long time to get the desired output |
| 4 | | **100** | **1** | **Most Stable** |
| 1 | BAND LIMITED WHITE NOISE | 50 | 1 | Less stable |
| 2 | | **100** | **1** | **Most Stable** |
| 3 | | 70 | 1 | Stable |
| 4 | | 10 | 1 | Take long time to get the desired output |
| 5 | | 150 | 1 | Unstable |

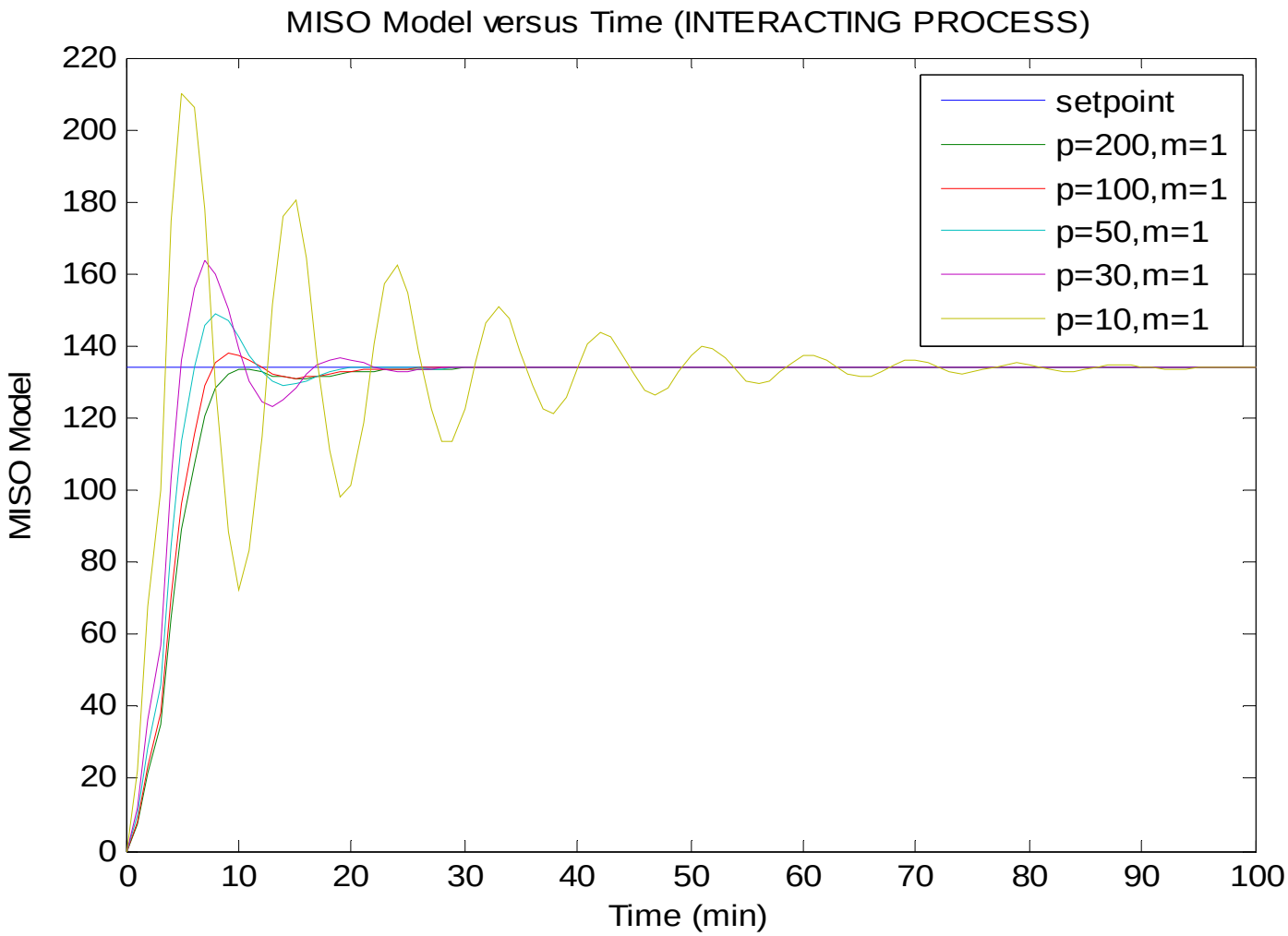


**Figure 4.6** The tuning graph for the MISO model using constant

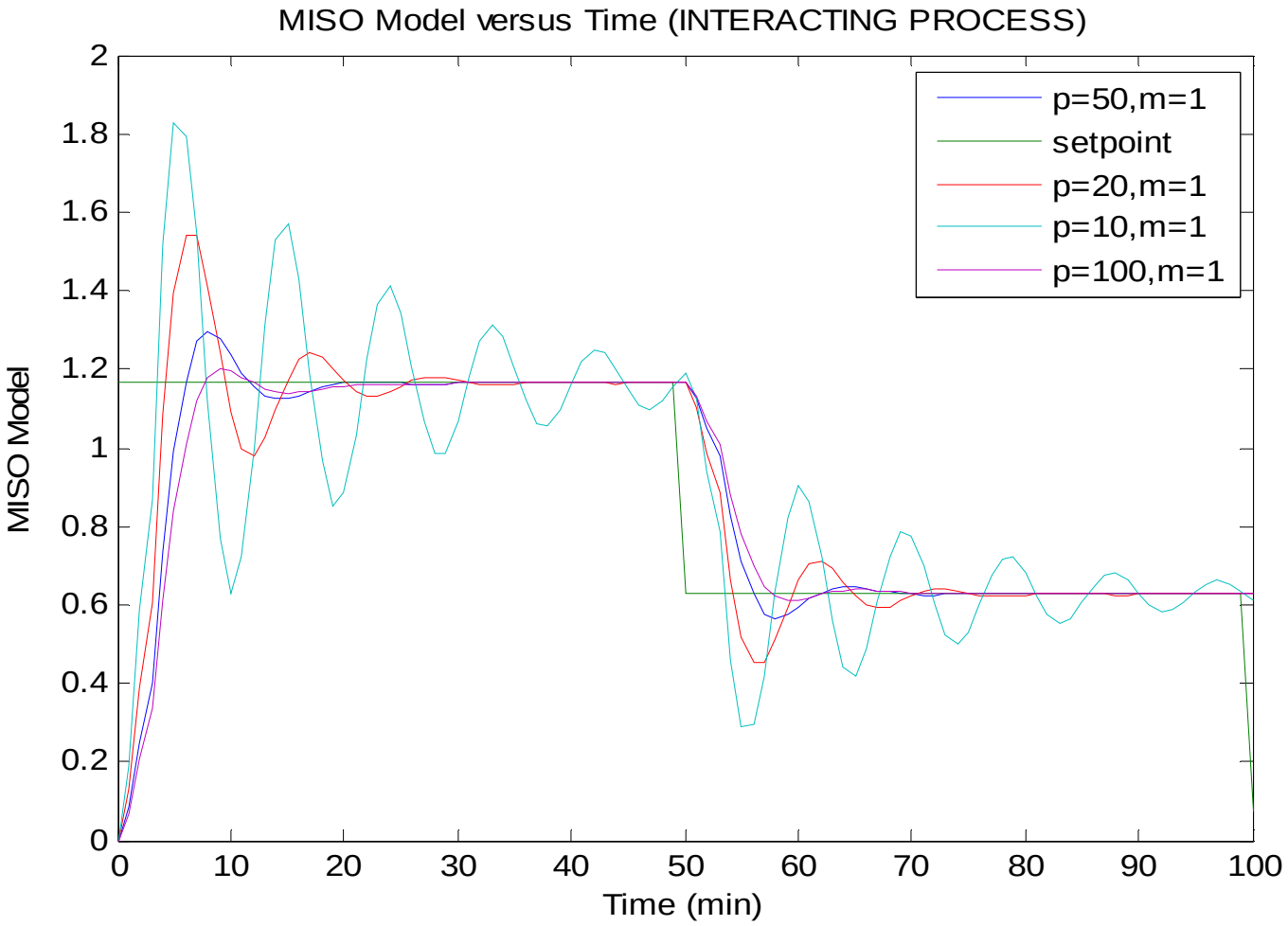


**Figure 4.7** The tuning graph for MISO model using random number

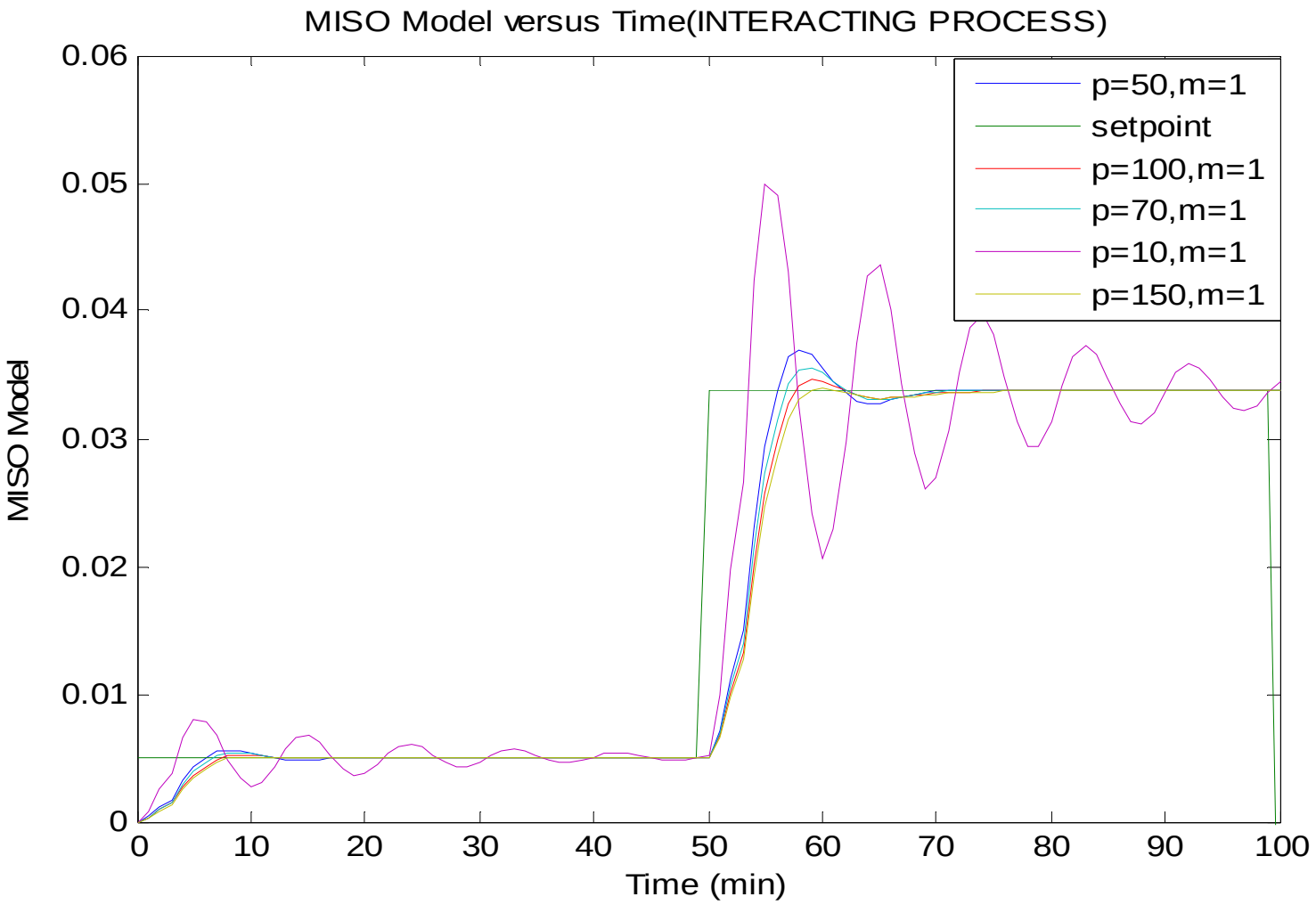


**Figure 4.8** The tuning graph for the MISO model using band limited white noise

Figure 4.6 until 4.8 shows the response of the interacting process for separator in three conditions which are without disturbance (constant), and with disturbances random number and band limited white noise. As the result, the optimize value for P and M is 100 and 1. This would give the best result for the MPC to this control loop as shown in Figure 4.9.

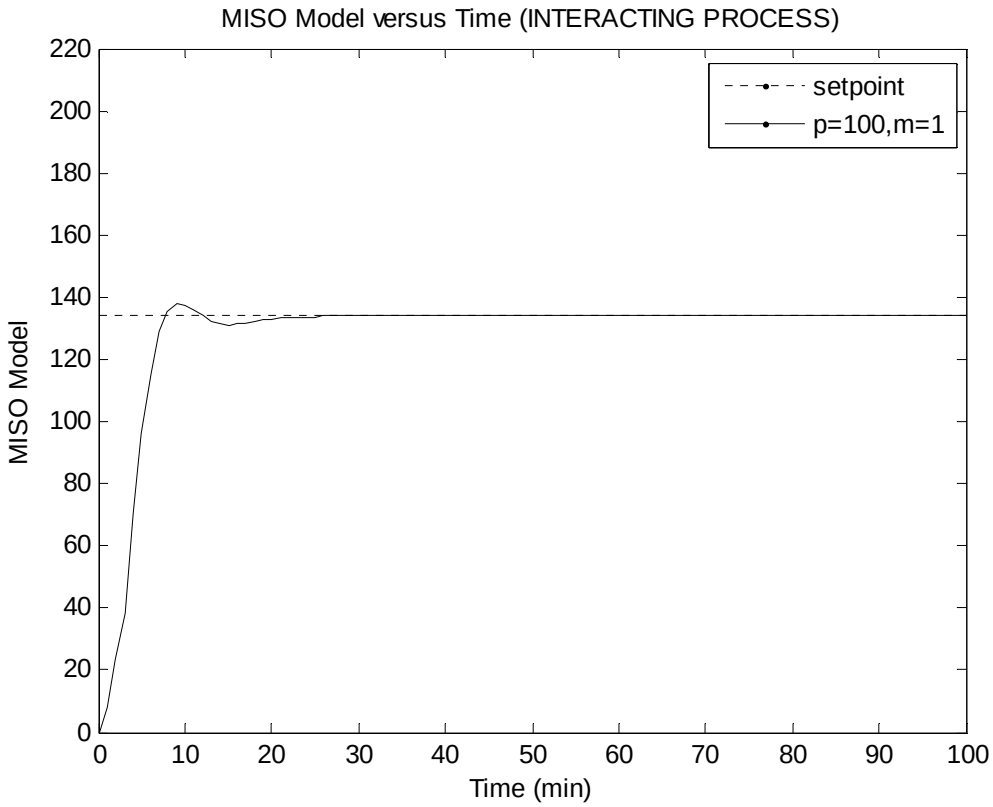


**Figure 4.9** The optimum condition for interacting (MISO) control loop

### 4.4.2 Tuning On The Separator Level Control Loop

Separator level control loop is use single transfer function that is transfer function for level. Separator level control loop was tuned for some value of P and M. The tuning value of P and M and the response performance is summarizing in the Table 4.2.

**Table 4.2:** Tuning value for separator level control loop

| TRIAL | SOURCES | TUNING VALUE | | PERFORMANCE |
|---|---|---|---|---|
| | | P | M | |
| 1 | CONSTANT | 10 | 2 | Less stable |
| 2 | | 1 | 1 | Take long time to get the desired output |
| 3 | | 2 | 2 | Take long time to get the desired output |
| 4 | | 3 | 1 | Take long time to get the desired output |
| 5 | | 20 | 2 | Unstable |
| 6 | | **15** | **2** | **Most Stable** |
| 1 | RANDOM NUMBER | 10 | 2 | Less stable |
| 2 | | **13** | **1** | **Most stable** |
| 3 | | 10 | 1 | Stable |
| 4 | | 50 | 1 | Unstable |
| 5 | | 1 | 1 | Take long time to get the desired output |
| 6 | | 3 | 4 | Take long time to get the desired output |
| 1 | BAND LIMITED WHITE NOISE | 10 | 2 | Stable |
| 2 | | 5 | 1 | Less stable |
| 3 | | **15** | **2** | **Most stable** |
| 4 | | 50 | 2 | Unstable |
| 5 | | 1 | 20 | Take long time to get the desired output |

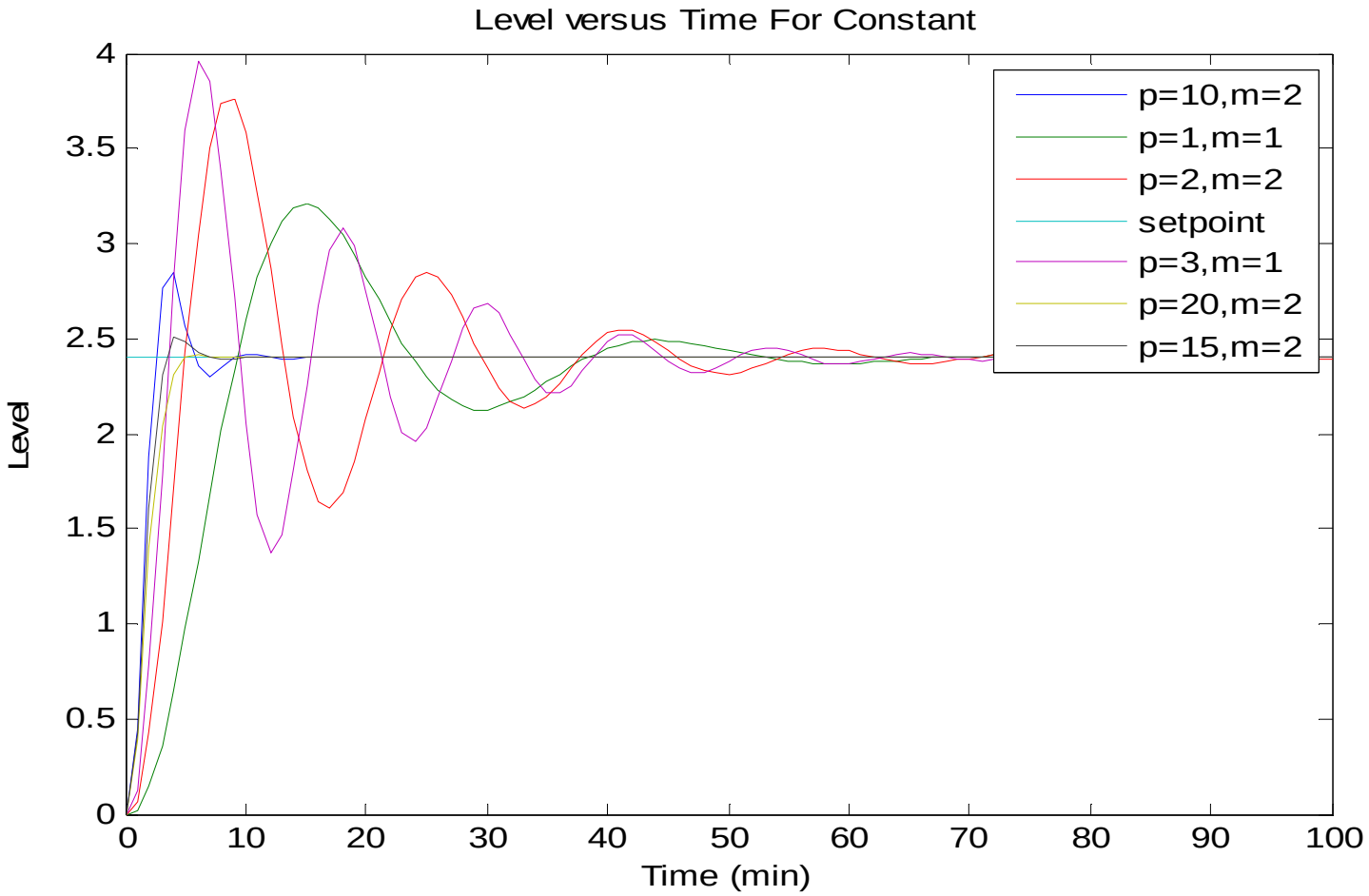


**Figure 4.10** The tuning graph for the separator level using constant input

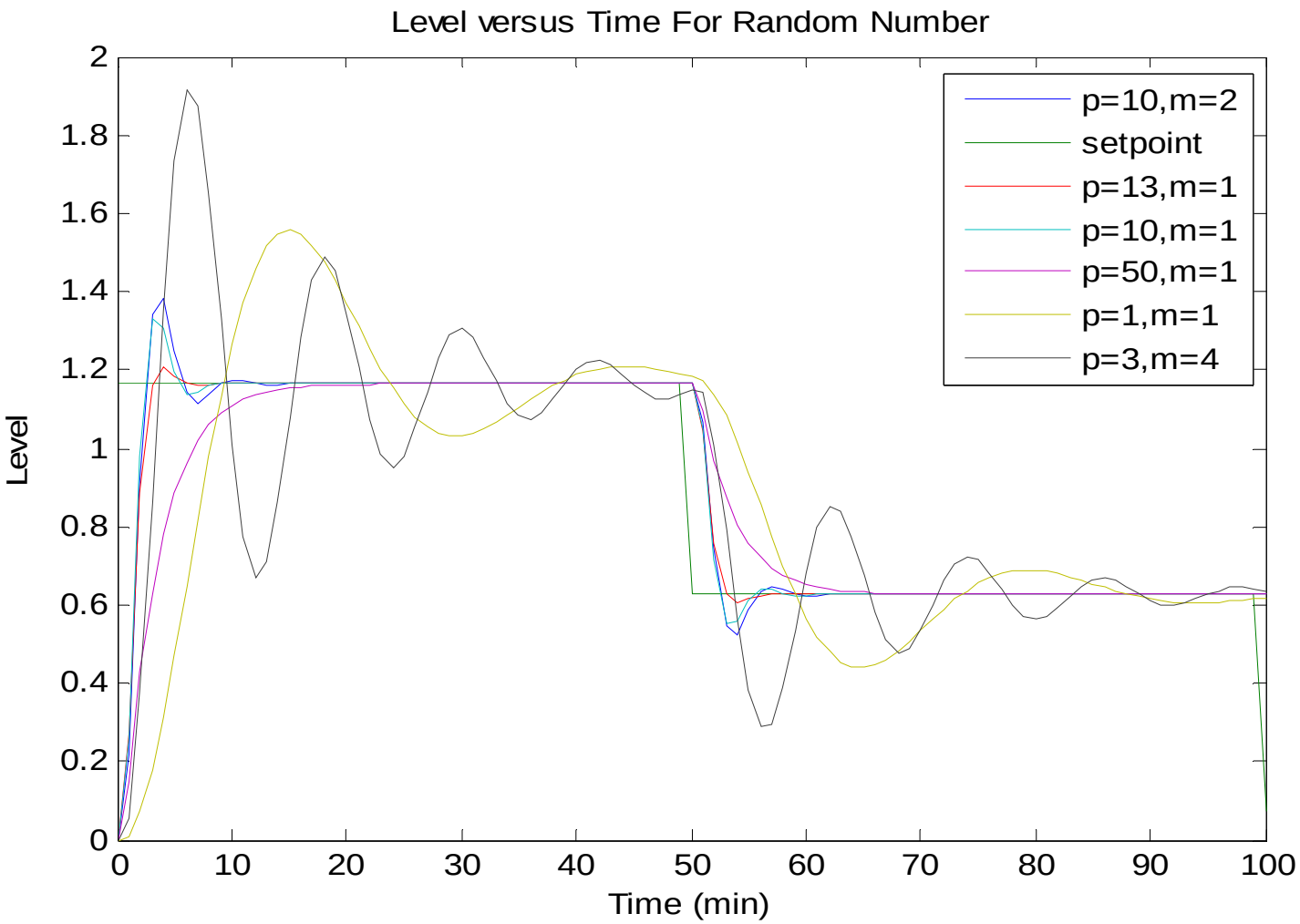


**Figure 4.11** The tuning graph for the separator level using random number input

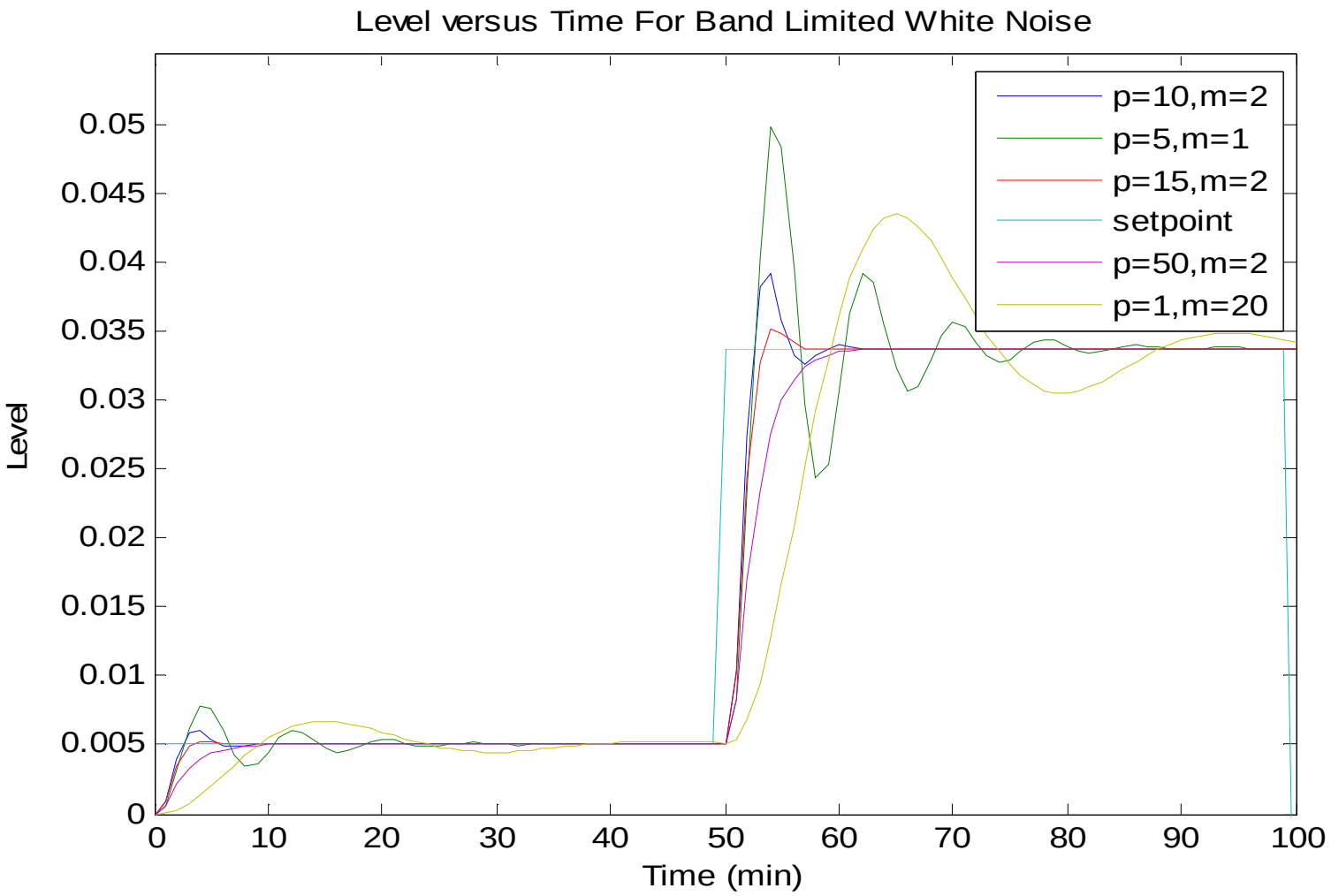


**Figure 4.12** The tuning graph for the separator level band limited white noise inputs

Figure 4.10 until 4.12 shows the response of the process for separator level in three conditions which are without disturbance (constant), and with disturbances random number and band limited white noise. As the result, the optimize value for P and M is 15 and 2. This would give the best result for the MPC to this control loop as shown in Figure 4.13.

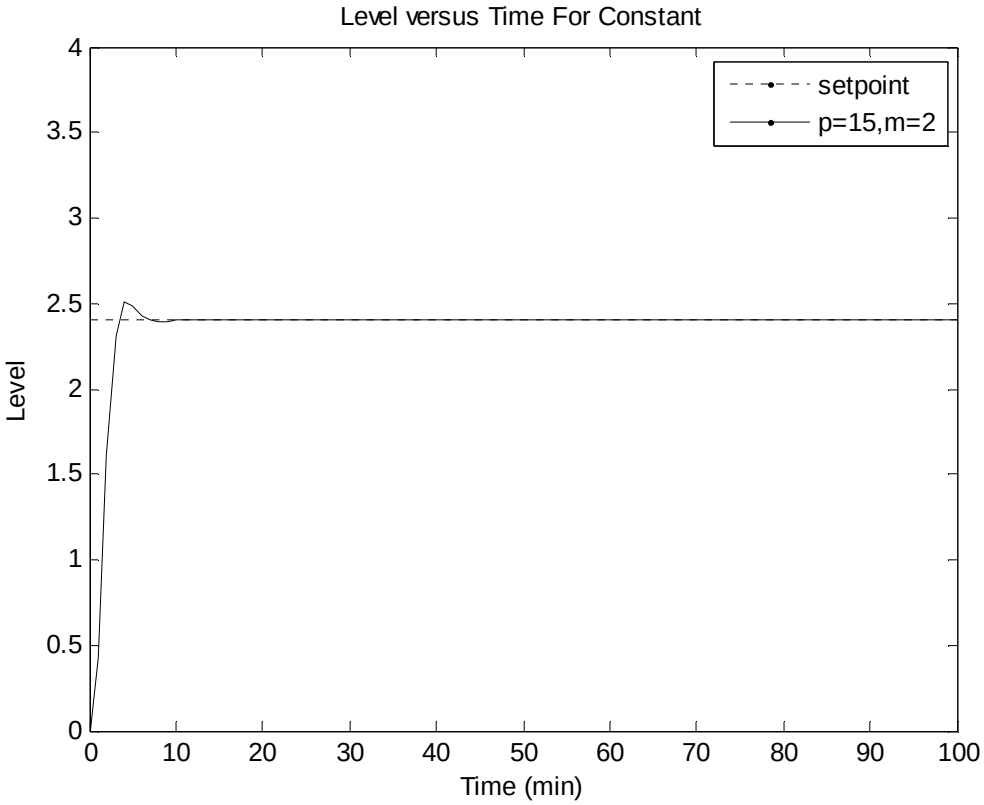


**Figure 4.13** The optimum condition for separator level control loop

### 4.4.3 Tuning For Separator Temperature Control Loop

The method for tuning process is repeated for separator temperature control loop. This loop used single transfer function which is transfer function for temperature. The control loop is also tuned with some value of P and M. The tuning value, P and M for separator temperature and the response performance is summarizing in the Table 4.3.

**Table 4.3:** The tuning value for separator temperature control loop

| TRIAL | SOURCES | TUNING VALUE | | PERFORMANCE |
|---|---|---|---|---|
| | | P | M | |
| 1 | CONSTANT | **500** | **1** | **Most stable** |
| 2 | | 200 | 1 | Less stable |
| 3 | | 50 | 1 | Take long time to get the desired output |
| 4 | | 100 | 1 | Less stable |
| 1 | RANDOM NUMBER | 50 | 1 | Less stable |
| 2 | | 100 | 1 | Less stable |
| 3 | | **500** | **1** | **Most stable** |
| 4 | | 50 | 2 | Take long time to get the desired output |
| 1 | BAND LIMITED WHITE NOISE | 40 | 1 | Take long time to get the desired output |
| 2 | | 100 | 1 | Les stable |
| 3 | | **500** | **1** | **Most stable** |
| 4 | | 1 | 1 | Take long time to get the desired output |

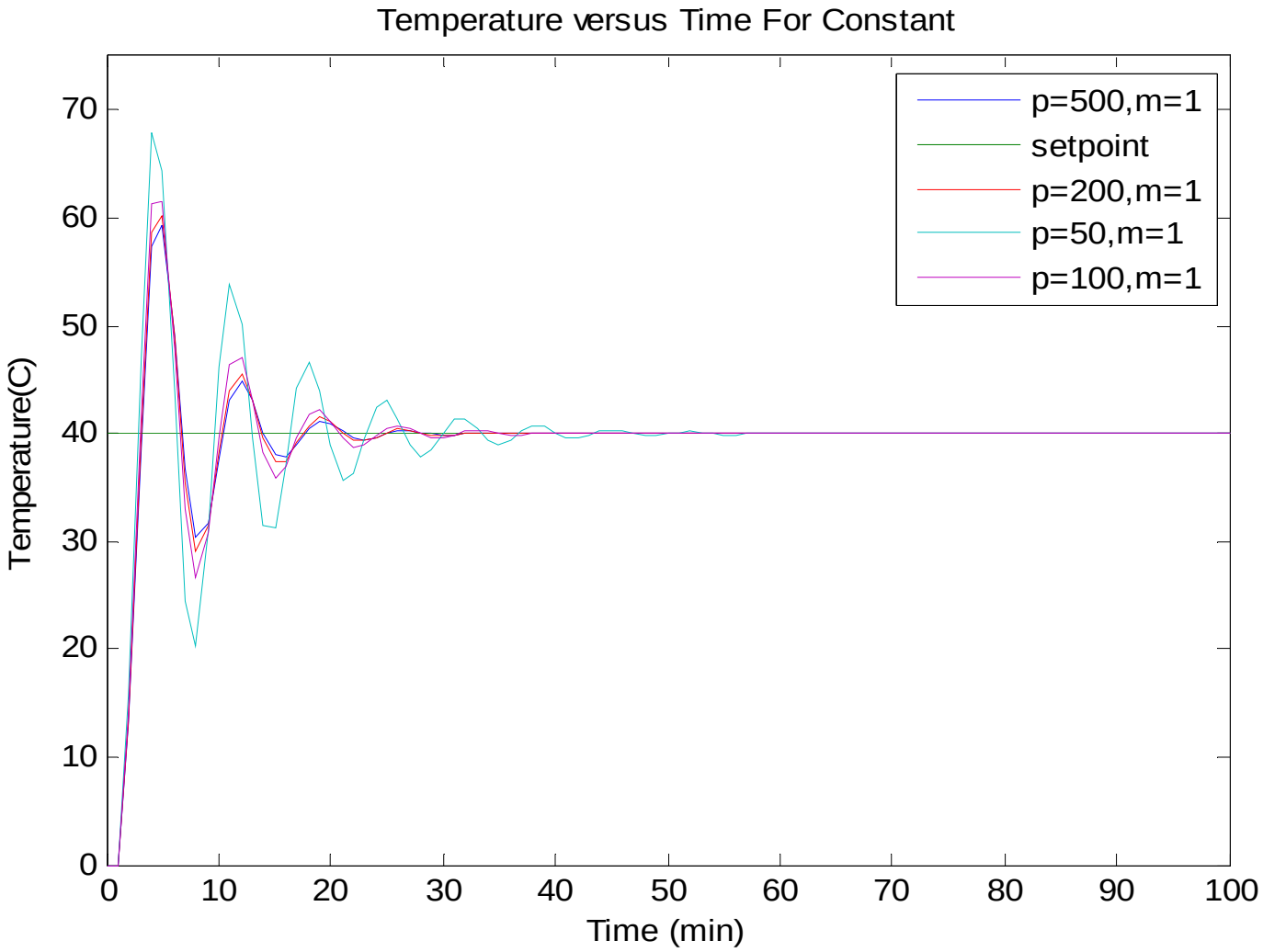


**Figure 4.14** Tuning process on the separator temperature using constant input

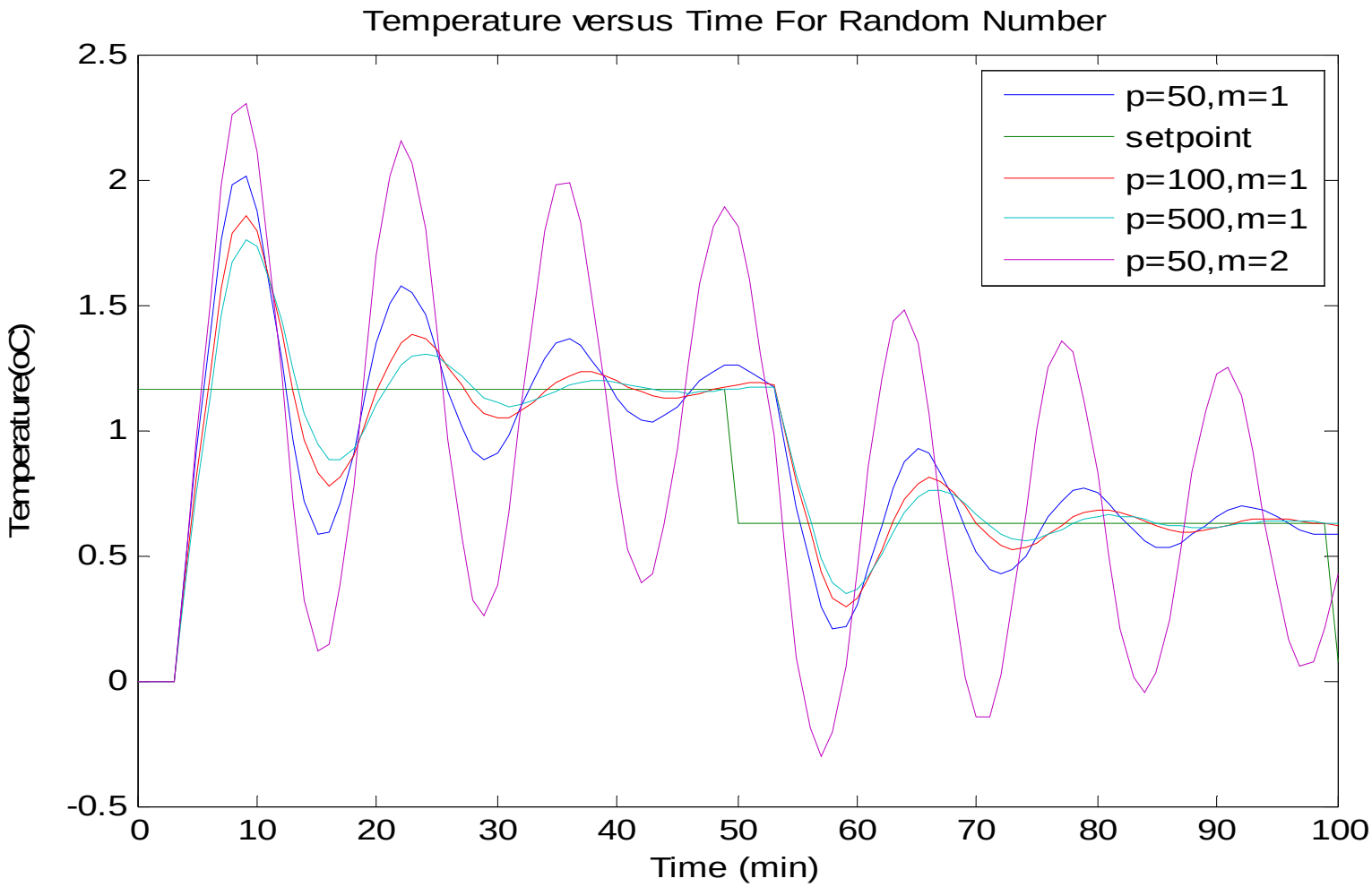


**Figure 4.15** Tuning process on separator temperature using random number input

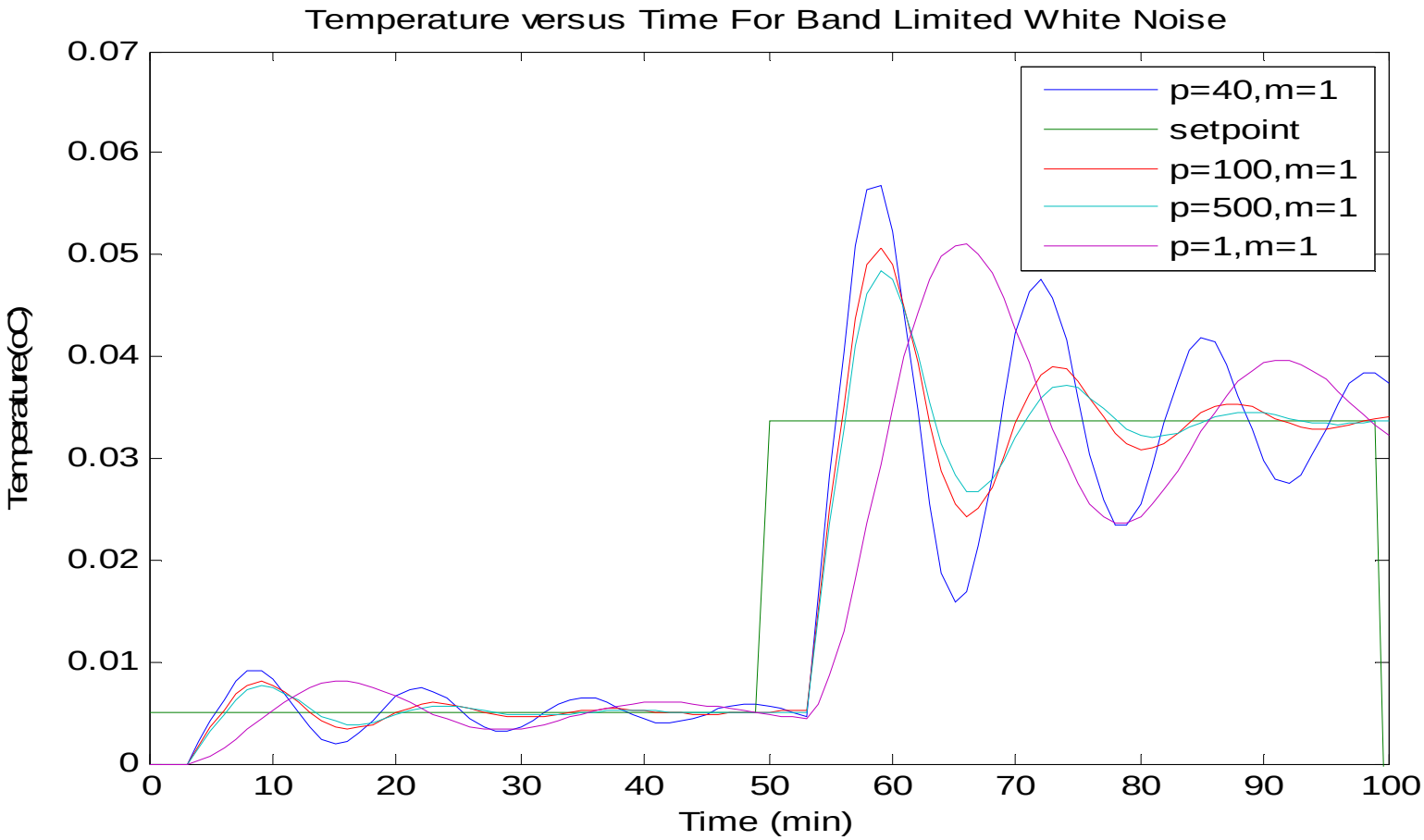


**Figure 4.16** Tuning process on the separator temperature using band limited white noise input

Figure 4.14 until 4.16 shows the response of the process for separator temperature in three conditions which are without disturbance (constant), and with disturbance random number and band limited white noise. As the result, the optimize value for P and M is 500 and 1. This would give the best result for the MPC to this control loop as shown in Figure 4.17.

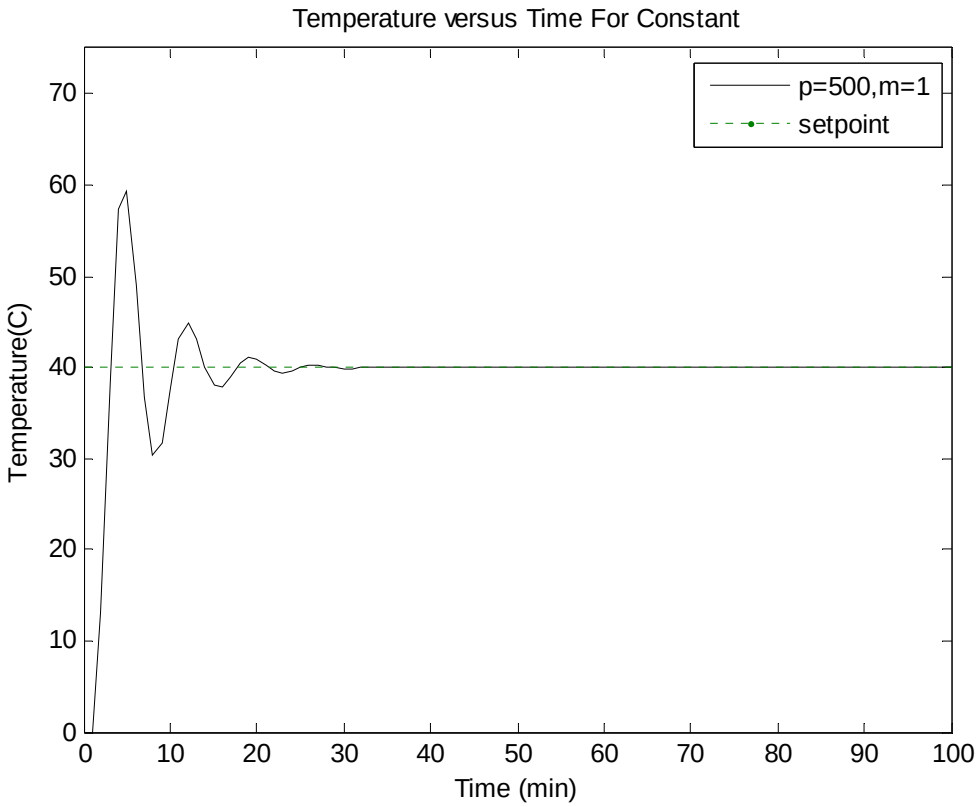


**Figure 4.17** The optimum condition for separator temperature control loop

## Concluding Remark

In this chapter, transfer function for VAC process has been presented. The transfer functions was developed using step test method by simplified the models as first order plus dead time (FOPDT). The method to develop the transfer function is referring

to Seborg *et al.* (2004). Two transfer functions were developed and they are used in developing MPC. Tuning process for MPC was made and results on sensitivity analysis carried out on the system have been displayed. The optimum condition for interacting closed loop, separator level closed loop and separator temperature control loop was determined and presented. Based on the results, it is concluded that the MPC is a very good controller. It is due to its ability to reject the disturbance presence and minimize the offset.

# CHAPTER 5

# PRODUCT OPTIMIZATION ON VINYL ACETATE MONOMER PROCESS USING MPC

## 5.1 Introduction

In a large scale plant, the output variables may be influenced by many input variables in a way which is not easy to predict. Chemical industries are deal with more complexes, nonlinear and highly interacting process which is hard to control with traditional controller. So, the study of MPC performance is important in order to overcome these problems.

In this study, the process has been tested with three inputs that is providing in MATLAB 7.1/ Simulink. These inputs are considering the mechanistic of the real chemical industry. They are constant value which is without disturbance presence, random number and band limited white noise.

The random number and band limited white noise are disturbances for the process. They are providing opportunity for sensitivity analysis of the MPC controller performance for disturbance rejection. The difference between band limited noise number and the random number is it produces output at a specific sample rate, which is related to the correlation time of the noise. From the tuning process in Section 4.4, the best performance of MPC and the optimum value for P and M has been determined.

Study was carried out for Multi Input Single Output (MISO) process. In MISO process, the input which is manipulated variables is more than one and only one output, which is controlled variable, is use. In this study, the controlled variable used is FEHE hot exit temperature while manipulated variables used are separator level and temperature.

## 5.2 MPC Performance For MISO Process

As mention earlier at Section 4.4.1, the interacting control loop is combination of both level and temperature transfer function. The controller has been tested with three inputs which are constant, random number and band limited white noise. The graphs are shown in Figure 5.1 until 5.3.

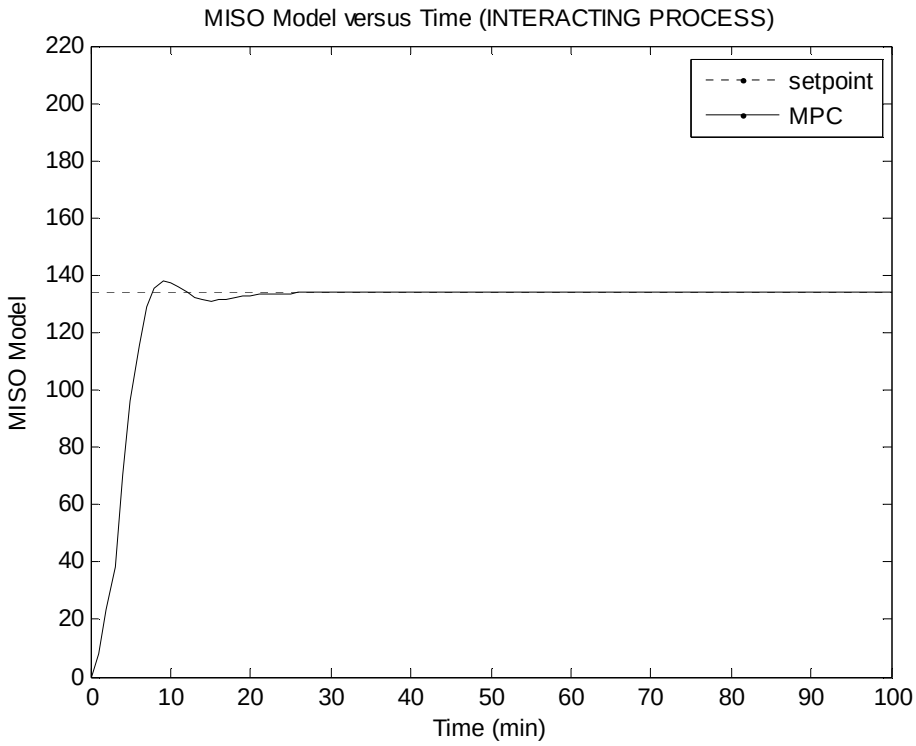


**Figure 5.1** MPC performance for MISO model using constant input

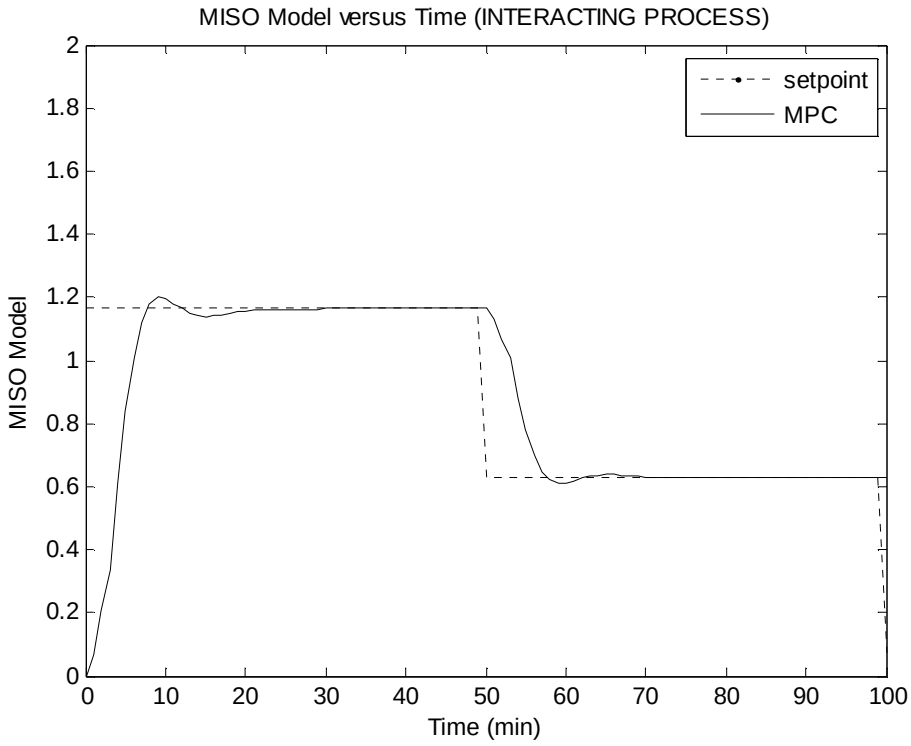


**Figure 5.2** MPC performance for MISO model using random number input

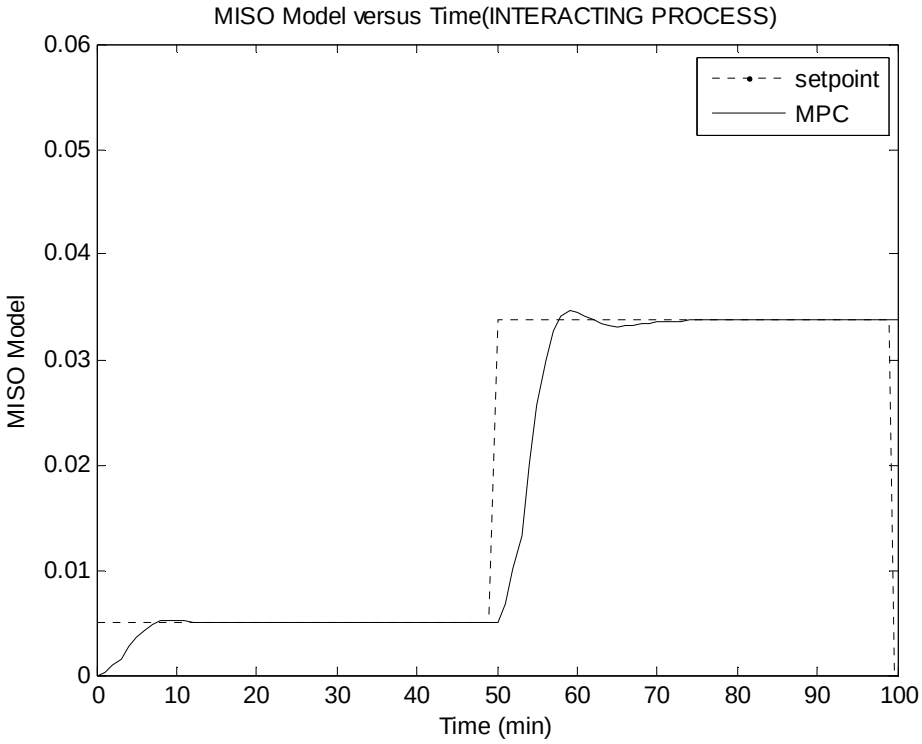


**Figure 5.3** MPC performance for MISO model using band limited white noise input

Figure 5.1 shows the MPC performance for MISO model using constant input. From the graph as can be seen, the response takes about 30 minutes to stable and the offset is very small. Figure 5.2 and 5.3 shows the performance for MISO model using random number and band limited white noise input. From Figure 5.2, the response is stable at 30 minutes while the response is stable at 10 minutes for Figure 5.3. For both graphs, the offset is very small. These results show that MPC is a good controller in terms of ability to reject disturbance and minimize offset.

## 5.3 Product Optimization Using MPC

Typically, processes with variables that interact with each other or that contain internal feedback of material and energy will exhibit so-called interacting behavior. In interacting process, the units and variables are relating each other. When there is a change in a unit, the other units will be affected.

Vinyl acetate monomer process is highly interacting process. In this research, a change was made in separator unit. The variables involved are level and temperature. When the change was made in separator level and temperature, one of the units that affected is distillation column where the vinyl acetate product taken. From the research that was carried out, it was found that MPC can optimize the production of vinyl acetate. The result is representing by Figure 5.4.

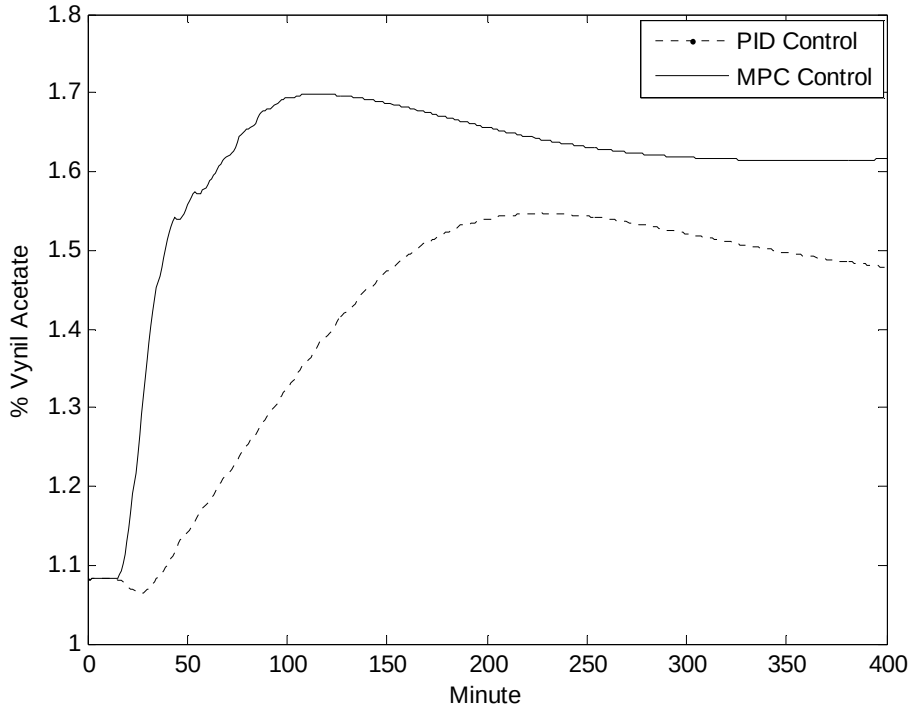


**Figure 5.4** Comparison between PI and MPC controller

From Figure 5.4, it shows that the MPC controller is stable faster than PI controller. At 325 minute, the MPC is achieved the stability while PI controller in not stable yet. Besides that, MPC also able optimize the production of vinyl acetate up to 1.7 mole % where maximum value that achieved by PI controller is only about 1.5 mole % of vinyl acetate. It is mean that MPC controller is better than PI controller.

So, from this study, the development of MPC leads to new finding, that is MPC ability to optimize the production of vinyl acetate. This is mean that MPC not only can be a controller to the process but it is also can optimize the desired product.

## 5.4 Concluding Remark

In this chapter, MISO process was introduced. In this process, two manipulated variable which are separator level and temperature were used. Sensitivity analysis of MPC has been carried out using two disturbances that are random number and band limited white noise. The MPC performance for MISO process has been studied and the results have been displayed. Based on the results, it is concluded that MPC is a very good controller in terms of ability to reject disturbances and minimize the offset. From the research, it is also found that the MPC has an ability to optimize the production of vinyl acetate. The result has been displayed. Based on the result, it is concluded that MPC is better than PI controller and it is not only can act as a controller to the process, but it is also can optimize the desired product.

# CHAPTER 6

# CONCLUSION AND RECOMMENDATION

## 6.1 Introduction

During the past decade, the importance of plantwide control has been recognised and considerable efforts have been directed to systematic procedures to satisfy the various requirements for effective process control.

The aim of this study is to design a Model Predictive Control for a separator in vinyl acetate monomer process so that the closed loop dynamic behavior and controllability features are adequate to satisfy the increasingly difficult product requirement.

In order to study the controller performance, a vinyl acetate monomer process, which consists of 10 unit operations, had been chosen as a test-bed. To facilitate the study, a simulation of this vinyl acetate monomer process was carried out using MATLAB 7.1. The simulated model was validated using steady state process data. This is explained in Chapter 3. Dynamic modeling and simulation of vinyl acetate monomer process also has been successfully accomplished in Chapter 3. Analysis of dynamic response was carried out and effect of set point tracking and disturbance rejection was identified.

In Chapter 4, transfer function for VAC process was developed using step test method by simplified the models as first order plus dead time (FOPDT). Two transfer

functions were developed. Then, MPC was developed using these transfer function and after that MPC was implemented. Tuning process for MPC was made and sensitivity analysis of MPC was carried out using two disturbances that are random number and band limited white noise.

In Chapter 5, MPC performance on MISO process was introduced. Sensitivity analysis of MPC on MISO process also was carried out using disturbances random number and band limited white noise. The MPC performance for MISO process has been studied.

## 6.2 Conclusion

Based on above discussion, the main conclusions of this research are as follows:

1) By using MATLAB 7.1, a dynamic model of vinyl acetate monomer process has been studied.
2) Simulation on vinyl acetate monomer process has been carried out for nominal condition and PI control. Dynamic response of the process and effect of set point tracking and disturbance rejection to the process has been analyzed.
3) The transfer functions for separator level and temperature has been developed from the simulation data.
4) Model Predictive Control has successfully developed and implemented. Tuned also made to see the MPC performance.
5) The MPC model was tested with some disturbances such as random number and band limited white noise. The test shows that the MPC is a good controller in term of ability to reject the disturbances and minimize offset. This study also leads to new finding that is MPC ability to optimize the production of vinyl acetate.

## 6.3 Recommendation for Future Work

The chemical industry it's more integrated with involving the nonlinear process which the parameter is affected each other. This research it's only concentrates to control on separator unit which is involved parameter level and temperature. It is better if control can be done on all unit operation to make a real plantwide control and considering all the effect and interactions between units so that whether MPC can perform or not can be determine for this situation. Besides that, since the product concentration was optimized in this research, an estimation process is needed in the future in order to measure the concentration.

## REFERENCES


Cutler, C. R., and Ramaker, B. L. (1979). Dynamic matrix control- a computer control algorithm. *AICHE national meeting*, Houston, TX.

Cutler, C. R., & Ramaker, B. L. (1980). Dynamic matrix control- a computer control algorithm. *In Proceedings of the joint automatic control conference*.

Dougherty, D. and Cooper, D. (2003). A Practical Multiple Model Adaptive Strategy for Single-loop MPC. *Control Engineering Practice* 11, 141–159.

Downs J.J and E. F. Vogel (1993). A plant-wide Industrial Process Control Problem. *Computers Chemical Engineering*, 17(2): 245-255.

Downs, J. and Vogel, E. (1993). A Plant-Wide Industrial Process Control Problem. *Computers & Chemical Engineering*. 17, 245.

Ergun, S. (2001). Fluid Fow Through Packed Columns. *Chem. Engineering Prog*ramme. 1952, 48, 89.

Garcia, C. E., & Morshedi, A. M. (1986). Quadratic programming solution of dynamic matrix control (QDMC). *Chemical Engineering Communications*, 46, 73-87.

Garcia, C. E., Prett, D. M., & Morari, M. (1989). Model predictive control: Theory and practice- a survey. *Automatica*, 25(3), 335–348.

Lee, J. H., Morari, M., & Garcia, C. E. (1994). State Space Interpretation of Model Predictive Control. *Automatica*, 30 (4), 707–717.

Luyben, M.L. and Tyreus, B.D. (1998). An Industrial Design/Control Study for the Vinyl Acetate Monomer Process. *Computers & Chemical Engineering*. 22, 867.

Marchetti, J. L., Mellichamp, D. A., & Seborg, D. E. (1983). Predictive control based on discrete convolution models. *Industrial & Engineering Chemistry, Processing Design and Development*, 22, 488–495.

Mc Avoy, T., Chen,R., Dave,K. (1998). A Nonlinear Dynamic Model of a Vinyl Acetate Process. *Department of Chemical Engineering/ Institute of System Research*. University of Maryland.

Morari, M. and Lee, J.H. (1999). *Model Predictive Control: Past, Present and Future*. *Computers and Chemical Engineering*, 23, 667–682.

Morshedi, A. M., Cutler, C. R., & Skrovanek, T. A. (1985). Optimal solution ofdynamic matrix control with linear programming techniques (LDMC). *Proceedings of the American Control Conference*, New Jersey: IEEE Publications, pp. 199–208.

Ogunnaike, B. A. (1986). Dynamic matrix control: A nonstochastic, industrial process control technique with parallels in applied statistics. *Industrial & Engineering Chemical Fundamentals*, 25, 712–718.

Qin, S. J., & Badgwell, T. A. (1996). An overview of industrial model predictive control technology. *In Chemical process control-CPC V*, CACHE.

Qin, S. J., & Badgwell, T. A. (1997). An overview of industrial model predictive control technology. In J. C. Kantor, C. E. Garcia, & B. Carnahan (Eds.), *Chemical process control- V, Fifth international conference on chemical process control* CACHE and AICHE, (pp.232–256).

Qin, S. J., & Badgwell, T. A. (2000). An overview of nonlinear model predictive control applications. In F. Allgower, & A. Zheng (Eds.), *Nonlinear model predictive control* (pp. 369–392). Switzerland: Birkhauser.

Qin, S.J and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice* 11, 733–764.

Reyes, F. and Luyben, W. (2001). Extensions of the Simultaneous Design of Gas-Phase Adiabatic Tubular Reactor Systems with Gas Recycle. *Ind. Eng. Chem. Res*. 40, 635.

Richalet, J. (1993). Industrial applications of model-based control. *Automatica*, 29, 1251–1274.

Richalet, J., Rault, A., Testud, J. L., & Papon, J. (1978). Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14, 413–428.

Ricker, N. L. and Lee, J. H.(1995b). Nonlinear modeling and state estimation for the Tennessee Eastman Challenge Process. *Computers and Chem. Engineering.* 19, 983-1005.

Seborg, D.E., Edgar, T.F., Mellichamp, D.A. (2004). *Process Dynamic and Control*. Second Edition. USA: John Wiley & Sons, Inc.

Shridhar, R., & Cooper, D. J. (1997). A tuning strategy for unconstrained SISO model predictive control. *Industrial & Engineering Chemical Research*, 36, 729–746.

Shridhar, R., & Cooper, D. J. (1998). A tuning strategy for unconstrained multivariable model predictive control. *Industrial & Engineering Chemical Research*, 37, 4003–4016.

Soni, A. (2000). A *Multi-Scale Approach To Fed-Batch Bioreactor Control*. University of Mumbai: Master Thesis.

Leong Yau, L. (2004). Plantwide Control of A Fatty Acid Fractionation Process. Universiti Teknologi Malaysia: Master Thesis.

# APPENDIX A

## THE VALUE OF MANIPULATED, CONTROLLED AND MEASUREMENT VARIABLES AT STEADY STATE

Table A.1: Steady-State Values of Manipulated Variables

| MV | Description | Steady State | Range | Unit |
|---|---|---|---|---|
| 1 | Fresh $O_2$ Feed | 0.52343 | 0-2.268 | Kmol/min |
| 2 | Fresh $C_2H_4$ Feed | 0.83522 | 0-7.56 | Kmol/min |
| 3 | Fresh HAC Feed | 0.79003 | 0-4.536 | Kmol/min |
| 4 | Vaporizer Steam Duty | 21877 | 0-1433400 | Kmol/min |
| 5 | Vaporizer Vapor Exit | 18.728 | 0-50 | Kmol/min |
| 6 | Vaporiver heater Duty | 9008.54 | 0-15000 | Kmol/min |
| 7 | Reactor Shell Temp. | 135.02 | 110-150 | $^oC$ |
| 8 | Separator Liquid Exit | 2.7544 | 0-4.536 | Kmol/min |
| 9 | Separator Jacket Temp. | 36.001 | 0-80 | $^oC$ |
| 10 | Separator Vapor Exit | 16.1026 | 0-30 | Kmol/min |
| 11 | Compressor Heater Duty | 27192 | 0-50000 | Kmol/min |
| 12 | Absorber Liquid Exit | 1.2137 | 0-4.536 | Kmol/min |
| 13 | Absorber Circulation Flow | 15.1198 | 0-50 | Kmol/min |
| 14 | Circulation Cooler Duty | 10730 | 0-30000 | Kmol/min |
| 15 | Absorber Scrub Flow | 0.756 | 0-7.56 | Kmol/min |
| 16 | Scrub Cooler Duty | 2018.43 | 0-5000 | Kmol/min |
| 17 | $CO_2$ Removal Inlet | 6.5531 | 0-22.68 | Kmol/min |
| 18 | Purge | 0.003157 | 0-0.02268 | Kmol/min |
| 19 | FEHE Bypass Ratio | 0.31303 | 0-1 | |
| 20 | Column Reflux | 4.9849 | 0-7.56 | Kmol/min |
| 21 | Column Reboiler Duty | 67179 | 0-100000 | Kmol/min |
| 22 | Column Condenser Duty | 60367 | 0-150000 | Kmol/min |
| 23 | Column Organic Exit | 0.829 | 0-2.4 | Kmol/min |
| 24 | Column Aqueous Exit | 0.8361 | 0-2.4 | Kmol/min |
| 25 | Column Bottom Exit | 2.1584 | 0-4.536 | Kmol/min |
| 26 | Vaporizer Liquid Inlet | 2.1924 | 0-4.536 | Kmol/min |

Table A.2: Control Structure and Controller Parameters.

| LOOP | Controlled Variable | Manipulated Variable | C.V. Value | Type | Kc | $T_R$(min) |
|---|---|---|---|---|---|---|
| 1 | % $O_2$ in the Reactor Inlet | $O_2$ Fresh Feed sp | 7.5% (0-20) | PI | 10 | 10 |
| 2 | Gas Recycle Stream Pressure | $C_2H_4$ Fresh Feed Valve | 128 psia (0-200) | PI | 0.3 | 20 |
| 3 | HAC Tank Level | HAC Fresh Feed Valve | 50% (0-100) | PI | 2 | |
| 4 | Vaporizer Level | Vaporizer heater Valve | 70% (0-100) | PI | 0.1 | 30 |
| 5 | Vaporizer Pressure | Vaporizer Vapor Exit Flowrate | 128 psia (0-200) | PI | 5 | 10 |
| 6 | Heater Exit Temp. | Reactor Preheater Valve | 150°C (120-170) | PI | 1 | 5 |
| 7 | Rector Exit Temp. | Steam Drum Pressure sp | 159.17°C (0-200) | PI | 3 | 10 |
| 8 | Separator Level | Separator Liquid Exit Valve | 50% (0-100) | P | 5 | |
| 9 | Separator Temp. | Separator Coolant Valve | 40°C (0-80) | PI | 5 | 20 |
| 10 | Separator Vapor Flowrate | Separator vapor Exit Valve | | FIXED | | |
| 11 | Compressor Exit Temp. | Compressor Heater Valve | 80°C (70-90) | PI | 1 | 5 |
| 12 | Absorber Level | Absorber Liquid Exit Valve | 50% (0-100) | P | 5 | |
| 13 | Absorber Scrub Flowrate | HAC Tank Exit Valve 2 | | FIXED | | |
| 14 | Circulation Stream Temp. | Absorber Scrub Heater Valve | 25°C (10-40) | PI | 1 | 5 |
| 15 | Absorber Circulation Flowrate | Absorber Circulation Valve | | FIXED | | |
| 16 | Scrub Stream Temp. | Circulation Cooler Valve | 25°C (10-40) | PI | 1 | 5 |
| 17 | % $CO_2$ in Gas Recycle | $CO_2$ Purge Flowrate sp | 0.764% (0-50%) | P | 1 | |
| 18 | % $C_2H_6$ in Gas Recycle | Purge Flowrate sp | 25% (0-100%) | P | 1 | |
| 19 | FEHE Hot Exit Temp. | Bypass Valve | 134°C (0-200) | PI | 5 | 10 |
| 20 | % $H_2O$ in Column Bottom | Column Reflux Flowrate sp | 9.344% (0-20) | PI | 0.5 | 60 |
| 21 | $5^{th}$ Tray Temp. | Reboiler Steam Valve | 110°C (0-120) | PI | 20 | 30 |
| 22 | Decanter Temp. | Column Condenser Duty | 45.845°C (40-50) | PI | 1 | 5 |
| 23 | Decanter Organic Level | Organic Product Flowrate | 50% (0-100) | P | 1 | |
| 24 | Decanter Aqueous Level | Aqueous Product Flowrate | 50% (0-100) | P | 1 | |
| 25 | Column Bottom Level | Column Bottom Flowrate | 50% (0-100) | P | 1 | |
| 26 | Liquid recycle Flowrate | HAC Tank Exit Valve 1 | | FIXED | | |

Table A.3: Measurements at Steady-state

| Measurement | Description | Value | Unit |
|---|---|---|---|
| 1 | Vaporizer Pressure | 128 | Psia |
| 2 | Vaporizer Level | 0.7 | |
| 3 | Vaporizer Temperature | 119.145 | $^{o}$C |
| 4 | Heater Exit Temp. | 150 | $^{o}$C |
| 5 | Rector Exit Temp. | 159.17 | $^{o}$C |
| 6 | Reactor Exit Flowrate | 18.857 | Kmol/min |
| 7 | FEHE Cold Exit Temp. | 97.1 | $^{o}$C |
| 8 | FEHE Hot Exit Temp. | 134 | $^{o}$C |
| 9 | Separator Level | 0.5 | |
| 10 | Separator Temp. | 40 | $^{o}$C |
| 11 | Compressor Exit Temp. | 80 | $^{o}$C |
| 12 | Absorber Pressure | 128 | Psia |
| 13 | Absorver Level | 0.5 | |
| 14 | Circulation Cooler Exit Temp. | 25 | $^{o}$C |
| 15 | Scrub Cooler Exit Temp | 25 | $^{o}$C |
| 16 | Gas Recycle Flowrate | 16.5359 | Kmol/min |
| 17 | Organic Product Flowrate | 0.829 | Kmol/min |
| 18 | Decanter Level (Organic) | 0.5 | |
| 19 | Decanter Level (Aqueous) | 0.5 | |
| 20 | Decanter Temp. | 45.845 | $^{o}$C |
| 21 | Column Bottom Level | 0.5 | |
| 22 | $5^{th}$ Tray Temp. | 110 | $^{o}$C |
| 23 | HAC Tank Level | 0.5 | |
| 24 | Organic Product Composition (VAC, $H_2O$, HAC) | 0.949786 | %mol |
| 25 | | 0.049862 | %mol |
| 26 | | 0.000352 | %mol |
| 27 | Column Bottom Composition (VAC, $H_2O$, HAC) | 0.00001 | %mol |
| 28 | | 0.09344 | %mol |
| 29 | | 0.90655 | %mol |
| 30 | Gas Recycle Composition ($O_2$, $CO_2$, $C_2H_4$, VAC, $H_2O$, HAC) | 0.055664 | %mol |
| 31 | | 0.007304 | %mol |
| 32 | | 0.681208 | %mol |
| 33 | | 0.249191 | %mol |
| 34 | | 0.001597 | %mol |
| 35 | | 0.000894 | %mol |
| 36 | | 0.004142 | %mol |
| 37 | Reactor Feed Composition ($O_2$,$CO_2$, $C_2H_4$, VAC, $H_2O$, HAC) | 0.075 | %mol |
| 38 | | 0.006273 | %mol |
| 39 | | 0.58511 | %mol |
| 40 | | 0.214038 | %mol |
| 41 | | 0.001373 | %mol |
| 42 | | 0.008558 | %mol |
| 43 | | 0.109648 | %mol |

# APPENDIX B

## PROGRAMMING DATA FOR DATA GENERATION

```
%test_VAcPlant(t, ID) is an example, which shows how to use the VAModel in MATLAB
%In this example, Luyben's multi-loop control structure is implemented and tested
%Eight process disturbances are available and transients are generated at the end of simulation
%On a PIII-1GHz PC, the VA plant model runs approaximately 80 times faster than the real time
%To use this example:
%       t sets the simulation time (in munite)
%       ID is an integer selected between 0 and 8
%           0: no disturbance
%           1: setpoint of the reactor outlet temperature decreases 8 degC (from 159 to 151)
%           2: setpoint of the reactor outlet temperature increases 6 degC (from 159 to 165)
%           3: setpoint of the H2O composition in the column bottom increases 9% (from 9% to 18%)
%           4: the vaporizer liqiud inlet flowrate increases 0.44 kmol/min (from 2.2 to 2.64)
%           5: HAc fresh feed stream lost for 5 minutes
%           6: O2 fresh feed stream lost for 5 minutes
%           7: C2H6 composition changes from 0.001 to 0.003 in the C2H4 fresh feed stream
%           8: column feed lost for 5 minutes
%       Note: in this example, all the disturbances occur at 10 minute after the simulation starts
%-------------------------------------------------------------------------
%Copyright: Rong Chen and Kedar David, June 2002
%-------------------------------------------------------------------------
function test_VAcPlant(minute,selected_ID)

%-------------------------------------------------------------------------
close all
format short g
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
%The recommended simulation sampling time is 1/3 second, which corresponds to a frequency of 180
samples in one minute
model_sampling_frequency=180;
%The recommended historical data sampling time is 1 minute, which corresponds to a frequency of 1
sample in one minute
storage_sampling_frequency=1;
%The recommended display sampling time is 1 minute, which corresponds to a frequency of 1 sample in one
minute
display_sampling_frequency=1;
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Inintialze process states and MV's
% 246 state variables
states=zeros(246,1);
% 26 manipulated variables
MVs=zeros(26,1);
% set the current process time to 0 (in minute)
time=0;
% set initialization flag 1
is_initial=1;
% set disturbance_ID 0
disturbance_ID=0;
```

```
% get the base operation
[dstatedt,states,MVs,y_ss]=VAModel(states,MVs,time,is_initial,disturbance_ID);
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Implement Luyben and Tyreus's control structure for the VA plant
% Note that: in this example,
%   1. for each transmitter, a 3 second time lag is used
%   2. for two analyzer transmitters (on the gas recycle and column bottom), a 10 minute deadtime is
used
%   3. for other transmitters, no deadtime is used
%   4. for three controllers using analyzers on the gas recycle and column bottom, a 10 minute sampling
time is used
%   5. for other controllers, a 1 second sampling time is used
%   6. for each transmitter, a 1 second sampling time is used
%-------------------------------------------------------------------------
transmitter_lag=0.05; %in minute
transmitter_deadtime=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 10 0 10 0 0 0 0 0 0]; %in minute
transmitter_sampling_frequency=[60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
60 60];
controller_sampling_frequency=[60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 0.1 0.1 60 0.1 60 60 60
60 60 60];
%-------------------------------------------------------------------------
%transmitter and controller initialization
%SP: setpoint (not scaled)
%K: controller gain which is dimentionless
%Ti: reset time (in minute)
%act: 1 if positive process gain, -1 if negative process gain
%mode: 1 for automatic, 2 for manual
%Ponly: 1 for proportional only control, 0 for PI control
%hc: controller sampling time (in minute), should be 1/"controller_sampling_frequency"
%ht: transmitter sampling time (in minute), should be 1/"transmitter_sampling_frequency"
%uLO: low limit of MV
%uHI: high limit of MV
%ded: should be "transmitter_deadtime" defined above
%tau: should be "transmitter_lag" defined above
%yLO: low limit of measurement
%yHI: high limit of measurement
%Nded: measurement deadtime storage, which should be
1+transmitter_deadtime*transmitter_sampling_frequency
%Ntau: measurement time constant storage, which should be 1 all the time
%-------------------------------------------------------------------------
%MV1: F_O2 - O2 composition
SP_O2=0.075;
K_O2=5;
Ti_O2=10;
act_O2=1;
mode_O2=1;
Ponly_O2=0;
ht_O2=1/transmitter_sampling_frequency(1);
hc_O2=1/controller_sampling_frequency(1);
uLO_O2=0;
uHI_O2=2.268;
ded_O2=transmitter_deadtime(1);
tau_O2=transmitter_lag;
yLO_O2=0;
yHI_O2=0.2;
Nded_O2=1+transmitter_deadtime(1)*transmitter_sampling_frequency(1);
Ntau_O2=1;
xxx_O2(1)= (y_ss(37) -yLO_O2) / (yHI_O2 -yLO_O2);
for i=2:Nded_O2
    xxx_O2(i)=xxx_O2(1);
end
for i=1:Ntau_O2
    yyy_O2(i)= xxx_O2(Nded_O2);
end
xi_O2= (MVs(1) - uLO_O2) /(uHI_O2 -uLO_O2)-K_O2*act_O2*((SP_O2-yLO_O2)/(yHI_O2-yLO_O2)-
yyy_O2(Ntau_O2));
%-------------------------------------------------------------------------
%controller initialization
%MV2: F_C2H4 - Recycle Pressure
```

```
SP_C2H4=128;
K_C2H4=0.3;
Ti_C2H4=20;
act_C2H4=1;
mode_C2H4=1;
Ponly_C2H4=0;
ht_C2H4=1/transmitter_sampling_frequency(2);
hc_C2H4=1/controller_sampling_frequency(2);
uLO_C2H4=0;
uHI_C2H4=7.56;
ded_C2H4=transmitter_deadtime(2);
tau_C2H4=transmitter_lag;
yLO_C2H4=0;
yHI_C2H4=200;
Nded_C2H4=1+transmitter_deadtime(2)*transmitter_sampling_frequency(2);
Ntau_C2H4=1;
xxx_C2H4(1)= (y_ss(12) -yLO_C2H4) / (yHI_C2H4 -yLO_C2H4);
for i=2:Nded_C2H4
    xxx_C2H4(i)= xxx_C2H4(1);
end
for i=1:Ntau_C2H4
    yyy_C2H4(i)= xxx_C2H4(Nded_C2H4);
end
xi_C2H4= (MVs(2) - uLO_C2H4) /(uHI_C2H4 -uLO_C2H4)-K_C2H4*act_C2H4*((SP_C2H4-yLO_C2H4)/(yHI_C2H4-
yLO_C2H4)-yyy_C2H4(Ntau_C2H4));
%-------------------------------------------------------------------------------
%controller initialization
%MV3: HAc feed stream - HAc tank level
SP_HAc=0.5;
K_HAc=2;
Ti_HAc=100;
act_HAc=1;
mode_HAc=1;
Ponly_HAc=1;
ht_HAc=1/transmitter_sampling_frequency(3);
hc_HAc=1/controller_sampling_frequency(3);
uLO_HAc=0;
uHI_HAc=4.536;
ded_HAc=transmitter_deadtime(3);
tau_HAc=transmitter_lag;
yLO_HAc=0;
yHI_HAc=1;
Nded_HAc=1+transmitter_deadtime(3)*transmitter_sampling_frequency(3);
Ntau_HAc=1;
xxx_HAc(1)= (y_ss(23) -yLO_HAc) / (yHI_HAc -yLO_HAc);
for i=2:Nded_HAc
    xxx_HAc(i)= xxx_HAc(1);
end
for i=1:Ntau_HAc
    yyy_HAc(i)= xxx_HAc(Nded_HAc);
end
xi_HAc= (MVs(3) - uLO_HAc) /(uHI_HAc -uLO_HAc)-K_HAc*act_HAc*((SP_HAc-yLO_HAc)/(yHI_HAc-yLO_HAc)-
yyy_HAc(Ntau_HAc));
%-------------------------------------------------------------------------------
%controller initialization
%MV4: Qv - Level
SP_LevelVap=0.7;
K_LevelVap=0.1;
Ti_LevelVap=30;
act_LevelVap=-1;
mode_LevelVap=1;
Ponly_LevelVap=0;
ht_LevelVap=1/transmitter_sampling_frequency(4);
hc_LevelVap=1/controller_sampling_frequency(4);
uLO_LevelVap=0;
uHI_LevelVap=1433400;
ded_LevelVap=transmitter_deadtime(4);
tau_LevelVap=transmitter_lag;
yLO_LevelVap=0;
yHI_LevelVap=1;
Nded_LevelVap=1+transmitter_deadtime(4)*transmitter_sampling_frequency(4);
```

```
Ntau_LevelVap=1;
xxx_LevelVap(1)= (y_ss(2) -yLO_LevelVap) / (yHI_LevelVap -yLO_LevelVap);
for i=2:Nded_LevelVap
    xxx_LevelVap(i)= xxx_LevelVap(1);
end
for i=1:Ntau_LevelVap
    yyy_LevelVap(i)= xxx_LevelVap(Nded_LevelVap);
end
xi_LevelVap= (MVs(4) - uLO_LevelVap) /(uHI_LevelVap -uLO_LevelVap)-
K_LevelVap*act_LevelVap*((SP_LevelVap-yLO_LevelVap)/(yHI_LevelVap-yLO_LevelVap)-
yyy_LevelVap(Ntau_LevelVap));
%-------------------------------------------------------------------------
%controller initialization
%MV5: F_Vaporizer - Vaporizer Pressure
SP_PresVap=128;
K_PresVap=5;
Ti_PresVap=10;
act_PresVap=-1;
mode_PresVap=1;
Ponly_PresVap=0;
ht_PresVap=1/transmitter_sampling_frequency(5);
hc_PresVap=1/controller_sampling_frequency(5);
uLO_PresVap=0;
uHI_PresVap=50;
ded_PresVap=transmitter_deadtime(5);
tau_PresVap=transmitter_lag;
yLO_PresVap=0;
yHI_PresVap=200;
Nded_PresVap=1+transmitter_deadtime(5)*transmitter_sampling_frequency(5);
Ntau_PresVap=1;
xxx_PresVap(1)= (y_ss(1) -yLO_PresVap) / (yHI_PresVap -yLO_PresVap);
for i=2:Nded_PresVap
    xxx_PresVap(i)= xxx_PresVap(1);
end
for i=1:Ntau_PresVap
    yyy_PresVap(i)= xxx_PresVap(Nded_PresVap);
end
xi_PresVap= (MVs(5) - uLO_PresVap) /(uHI_PresVap -uLO_PresVap)-K_PresVap*act_PresVap*((SP_PresVap-
yLO_PresVap)/(yHI_PresVap-yLO_PresVap)-yyy_PresVap(Ntau_PresVap));
%-------------------------------------------------------------------------
%controller initialization
%MV6: Q_heater - Vaporizer Outlet Temperature
SP_ReactorInletTemp=150;
K_ReactorInletTemp=1;
Ti_ReactorInletTemp=5;
act_ReactorInletTemp=1;
mode_ReactorInletTemp=1;
Ponly_ReactorInletTemp=0;
ht_ReactorInletTemp=1/transmitter_sampling_frequency(6);
hc_ReactorInletTemp=1/controller_sampling_frequency(6);
uLO_ReactorInletTemp=0;
uHI_ReactorInletTemp=15000;
ded_ReactorInletTemp=transmitter_deadtime(6);
tau_ReactorInletTemp=transmitter_lag;
yLO_ReactorInletTemp=120;
yHI_ReactorInletTemp=170;
Nded_ReactorInletTemp=1+transmitter_deadtime(6)*transmitter_sampling_frequency(6);
Ntau_ReactorInletTemp=1;
xxx_ReactorInletTemp(1)= (y_ss(4) -yLO_ReactorInletTemp) / (yHI_ReactorInletTemp -
yLO_ReactorInletTemp);
for i=2:Nded_ReactorInletTemp
    xxx_ReactorInletTemp(i)= xxx_ReactorInletTemp(1);
end
for i=1:Ntau_ReactorInletTemp
    yyy_ReactorInletTemp(i)= xxx_ReactorInletTemp(Nded_ReactorInletTemp);
end
xi_ReactorInletTemp= (MVs(6) - uLO_ReactorInletTemp) /(uHI_ReactorInletTemp -uLO_ReactorInletTemp)-
K_ReactorInletTemp*act_ReactorInletTemp*((SP_ReactorInletTemp-
yLO_ReactorInletTemp)/(yHI_ReactorInletTemp-yLO_ReactorInletTemp)-
yyy_ReactorInletTemp(Ntau_ReactorInletTemp));
%-------------------------------------------------------------------------
```

```
%controller initialization
%MV7: T_Shell - T_RCTOUT
SP_TRCT=159.17;
K_TRCT=3;
Ti_TRCT=10;
act_TRCT=1;
mode_TRCT=1;
Ponly_TRCT=0;
ht_TRCT=1/transmitter_sampling_frequency(7);
hc_TRCT=1/controller_sampling_frequency(7);
uLO_TRCT=110;
uHI_TRCT=150;
ded_TRCT=transmitter_deadtime(7);
tau_TRCT=transmitter_lag;
yLO_TRCT=0;
yHI_TRCT=200;
Nded_TRCT=1+transmitter_deadtime(7)*transmitter_sampling_frequency(7);
Ntau_TRCT=1;
xxx_TRCT(1)= (y_ss(5)-yLO_TRCT) / (yHI_TRCT -yLO_TRCT);
for i=2:Nded_TRCT
    xxx_TRCT(i)= xxx_TRCT(1);
end
for i=1:Ntau_TRCT
    yyy_TRCT(i)= xxx_TRCT(Nded_TRCT);
end
xi_TRCT= (MVs(7) - uLO_TRCT) /(uHI_TRCT -uLO_TRCT)-K_TRCT*act_TRCT*((SP_TRCT-yLO_TRCT)/(yHI_TRCT-
yLO_TRCT)-yyy_TRCT(Ntau_TRCT));
%-------------------------------------------------------------------------------
%controller initialization
%MV8: Separator Liquid Exit - Separator Level
SP_LevelSep=0.5;
K_LevelSep=0;
Ti_LevelSep=1;
act_LevelSep=-1;
mode_LevelSep=1;
Ponly_LevelSep=1;
ht_LevelSep=1/transmitter_sampling_frequency(8);
hc_LevelSep=1/controller_sampling_frequency(8);
uLO_LevelSep=0;
uHI_LevelSep=4.536;
ded_LevelSep=transmitter_deadtime(8);
tau_LevelSep=transmitter_lag;
yLO_LevelSep=0;
yHI_LevelSep=1;
Nded_LevelSep=1+transmitter_deadtime(8)*transmitter_sampling_frequency(8);
Ntau_LevelSep=1;
xxx_LevelSep(1)= (y_ss(9)-yLO_LevelSep) / (yHI_LevelSep -yLO_LevelSep);
for i=2:Nded_LevelSep
    xxx_LevelSep(i)= xxx_LevelSep(1);
end
for i=1:Ntau_LevelSep
    yyy_LevelSep(i)= xxx_LevelSep(Nded_LevelSep);
end
xi_LevelSep= (MVs(8) - uLO_LevelSep) /(uHI_LevelSep -uLO_LevelSep)-
K_LevelSep*act_LevelSep*((SP_LevelSep-yLO_LevelSep)/(yHI_LevelSep-yLO_LevelSep)-
yyy_LevelSep(Ntau_LevelSep));
%-------------------------------------------------------------------------------
%controller initialization
%MV9: T_Shell_Separator - Separator Liquid Temperature
SP_TempSep=40;
K_TempSep=0;
Ti_TempSep=1;
act_TempSep=-1;
mode_TempSep=1;
Ponly_TempSep=0;
ht_TempSep=1/transmitter_sampling_frequency(9);
hc_TempSep=1/controller_sampling_frequency(9);
uLO_TempSep=0;
uHI_TempSep=80;
ded_TempSep=transmitter_deadtime(9);
tau_TempSep=transmitter_lag;
```

```
yLO_TempSep=0;
yHI_TempSep=80;
Nded_TempSep=1+transmitter_deadtime(9)*transmitter_sampling_frequency(9);
Ntau_TempSep=1;
xxx_TempSep(1)= (y_ss(10)-yLO_TempSep) / (yHI_TempSep -yLO_TempSep);
for i=2:Nded_TempSep
    xxx_TempSep(i)= xxx_TempSep(1);
end
for i=1:Ntau_TempSep
    yyy_TempSep(i)= xxx_TempSep(Nded_TempSep);
end
xi_TempSep= (MVs(9) - uLO_TempSep) /(uHI_TempSep -uLO_TempSep)-K_TempSep*act_TempSep*((SP_TempSep-
yLO_TempSep)/(yHI_TempSep-yLO_TempSep)-yyy_TempSep(Ntau_TempSep));
%------------------------------------------------------------------------
%controller initialization
%MV10: Separator Vapor Exit is fixed at 16.1026 kmol/min
%------------------------------------------------------------------------
%controller initialization
%MV11: Q_Compressor_Cooler - Compressor Outlet Temperature (80degC)
SP_CompOutletTemp=80;
K_CompOutletTemp=1;
Ti_CompOutletTemp=5;
act_CompOutletTemp=-1;
mode_CompOutletTemp=1;
Ponly_CompOutletTemp=0;
ht_CompOutletTemp=1/transmitter_sampling_frequency(11);
hc_CompOutletTemp=1/controller_sampling_frequency(11);
uLO_CompOutletTemp=0;
uHI_CompOutletTemp=50000;
ded_CompOutletTemp=transmitter_deadtime(11);
tau_CompOutletTemp=transmitter_lag;
yLO_CompOutletTemp=70;
yHI_CompOutletTemp=90;
Nded_CompOutletTemp=1+transmitter_deadtime(11)*transmitter_sampling_frequency(11);
Ntau_CompOutletTemp=1;
xxx_CompOutletTemp(1)= (y_ss(11) -yLO_CompOutletTemp) / (yHI_CompOutletTemp -yLO_CompOutletTemp);
for i=2:Nded_CompOutletTemp
    xxx_CompOutletTemp(i)= xxx_CompOutletTemp(1);
end
for i=1:Ntau_CompOutletTemp
    yyy_CompOutletTemp(i)= xxx_CompOutletTemp(Nded_CompOutletTemp);
end
xi_CompOutletTemp= (MVs(11) - uLO_CompOutletTemp) /(uHI_CompOutletTemp -uLO_CompOutletTemp)-
K_CompOutletTemp*act_CompOutletTemp*((SP_CompOutletTemp-yLO_CompOutletTemp)/(yHI_CompOutletTemp-
yLO_CompOutletTemp)-yyy_CompOutletTemp(Ntau_CompOutletTemp));
%------------------------------------------------------------------------
%controller initialization
%MV12: Absorber Liquid Exit - Absorber Level
SP_LevelAbs=0.5;
K_LevelAbs=5;
Ti_LevelAbs=100;
act_LevelAbs=-1;
mode_LevelAbs=1;
Ponly_LevelAbs=1;
ht_LevelAbs=1/transmitter_sampling_frequency(12);
hc_LevelAbs=1/controller_sampling_frequency(12);
uLO_LevelAbs=0;
uHI_LevelAbs=4.536;
ded_LevelAbs=transmitter_deadtime(12);
tau_LevelAbs=transmitter_lag;
yLO_LevelAbs=0;
yHI_LevelAbs=1;
Nded_LevelAbs=1+transmitter_deadtime(12)*transmitter_sampling_frequency(12);
Ntau_LevelAbs=1;
xxx_LevelAbs(1)= (y_ss(13)-yLO_LevelAbs) / (yHI_LevelAbs -yLO_LevelAbs);
for i=2:Nded_LevelAbs
    xxx_LevelAbs(i)= xxx_LevelAbs(1);
end
for i=1:Ntau_LevelAbs
    yyy_LevelAbs(i)= xxx_LevelAbs(Nded_LevelAbs);
end
```

```
xi_LevelAbs= (MVs(12) - uLO_LevelAbs) /(uHI_LevelAbs -uLO_LevelAbs)-
K_LevelAbs*act_LevelAbs*((SP_LevelAbs-yLO_LevelAbs)/(yHI_LevelAbs-yLO_LevelAbs)-
yyy_LevelAbs(Ntau_LevelAbs));
%-------------------------------------------------------------------------
%controller initialization
%MV13: Circulation Flowrate is fixed to 15.1198 kmol/min
%-------------------------------------------------------------------------
%controller initialization
%MV14: Q_Cooler_Circulation - Circulation Temperature
SP_CircOutletTemp=25;
K_CircOutletTemp=1;
Ti_CircOutletTemp=5;
act_CircOutletTemp=-1;
mode_CircOutletTemp=1;
Ponly_CircOutletTemp=0;
ht_CircOutletTemp=1/transmitter_sampling_frequency(14);
hc_CircOutletTemp=1/controller_sampling_frequency(14);
uLO_CircOutletTemp=0;
uHI_CircOutletTemp=30000;
ded_CircOutletTemp=transmitter_deadtime(14);
tau_CircOutletTemp=transmitter_lag;
yLO_CircOutletTemp=10;
yHI_CircOutletTemp=40;
Nded_CircOutletTemp=1+transmitter_deadtime(14)*transmitter_sampling_frequency(14);
Ntau_CircOutletTemp=1;
xxx_CircOutletTemp(1)= (y_ss(14) -yLO_CircOutletTemp) / (yHI_CircOutletTemp -yLO_CircOutletTemp);
for i=2:Nded_CircOutletTemp
    xxx_CircOutletTemp(i)= xxx_CircOutletTemp(1);
end
for i=1:Ntau_CircOutletTemp
    yyy_CircOutletTemp(i)= xxx_CircOutletTemp(Nded_CircOutletTemp);
end
xi_CircOutletTemp= (MVs(14) - uLO_CircOutletTemp) /(uHI_CircOutletTemp -uLO_CircOutletTemp)-
K_CircOutletTemp*act_CircOutletTemp*((SP_CircOutletTemp-yLO_CircOutletTemp)/(yHI_CircOutletTemp-
yLO_CircOutletTemp)-yyy_CircOutletTemp(Ntau_CircOutletTemp));
%-------------------------------------------------------------------------
%controller initialization
%MV15: Scrub Flowrate is fixed to 0.756 kmol/min
%-------------------------------------------------------------------------
%controller initialization
%MV16: Q_Cooler_Scrub - Scrub Temperature
SP_ScrubOutletTemp=25;
K_ScrubOutletTemp=1;
Ti_ScrubOutletTemp=5;
act_ScrubOutletTemp=-1;
mode_ScrubOutletTemp=1;
Ponly_ScrubOutletTemp=0;
ht_ScrubOutletTemp=1/transmitter_sampling_frequency(16);
hc_ScrubOutletTemp=1/controller_sampling_frequency(16);
uLO_ScrubOutletTemp=0;
uHI_ScrubOutletTemp=5000;
ded_ScrubOutletTemp=transmitter_deadtime(16);
tau_ScrubOutletTemp=transmitter_lag;
yLO_ScrubOutletTemp=10;
yHI_ScrubOutletTemp=40;
Nded_ScrubOutletTemp=1+transmitter_deadtime(16)*transmitter_sampling_frequency(16);
Ntau_ScrubOutletTemp=1;
xxx_ScrubOutletTemp(1)= (y_ss(15) -yLO_ScrubOutletTemp) / (yHI_ScrubOutletTemp -yLO_ScrubOutletTemp);
for i=2:Nded_ScrubOutletTemp
    xxx_ScrubOutletTemp(i)= xxx_ScrubOutletTemp(1);
end
for i=1:Ntau_ScrubOutletTemp
    yyy_ScrubOutletTemp(i)= xxx_ScrubOutletTemp(Nded_ScrubOutletTemp);
end
xi_ScrubOutletTemp= (MVs(16) - uLO_ScrubOutletTemp) /(uHI_ScrubOutletTemp -uLO_ScrubOutletTemp)-
K_ScrubOutletTemp*act_ScrubOutletTemp*((SP_ScrubOutletTemp-yLO_ScrubOutletTemp)/(yHI_ScrubOutletTemp-
yLO_ScrubOutletTemp)-yyy_ScrubOutletTemp(Ntau_ScrubOutletTemp));
%-------------------------------------------------------------------------
%controller initialization
%MV17: CO2 - CO2 composition
SP_CO2=0.0076393;
```

```
K_CO2=1;
Ti_CO2=100;
act_CO2=-1;
mode_CO2=1;
Ponly_CO2=1;
ht_CO2=1/transmitter_sampling_frequency(17);
hc_CO2=1/controller_sampling_frequency(17);
uLO_CO2=0;
uHI_CO2=22.68;
ded_CO2=transmitter_deadtime(17);
tau_CO2=transmitter_lag;
yLO_CO2=0;
yHI_CO2=0.5;
Nded_CO2=1+transmitter_deadtime(17)*transmitter_sampling_frequency(17);
Ntau_CO2=1;
xxx_CO2(1)= (y_ss(31) -yLO_CO2) / (yHI_CO2 -yLO_CO2);
for i=2:Nded_CO2
    xxx_CO2(i)= xxx_CO2(1);
end
for i=1:Ntau_CO2
    yyy_CO2(i)= xxx_CO2(Nded_CO2);
end
xi_CO2= (MVs(17) - uLO_CO2) /(uHI_CO2 -uLO_CO2)-K_CO2*act_CO2*((SP_CO2-yLO_CO2)/(yHI_CO2-yLO_CO2)-
yyy_CO2(Ntau_CO2));
%-------------------------------------------------------------------------------
%controller initialization
%MV18: Purge - C2H6 composition
SP_C2H6=0.25;
K_C2H6=1;
Ti_C2H6=100;
act_C2H6=-1;
mode_C2H6=1;
Ponly_C2H6=1;
ht_C2H6=1/transmitter_sampling_frequency(18);
hc_C2H6=1/controller_sampling_frequency(18);
uLO_C2H6=0;
uHI_C2H6=0.02268;
ded_C2H6=transmitter_deadtime(18);
tau_C2H6=transmitter_lag;
yLO_C2H6=0;
yHI_C2H6=1;
Nded_C2H6=1+transmitter_deadtime(18)*transmitter_sampling_frequency(18);
Ntau_C2H6=1;
xxx_C2H6(1)= (y_ss(33) -yLO_C2H6) / (yHI_C2H6 -yLO_C2H6);
for i=2:Nded_C2H6
    xxx_C2H6(i)= xxx_C2H6(1);
end
for i=1:Ntau_C2H6
    yyy_C2H6(i)= xxx_C2H6(Nded_C2H6);
end
xi_C2H6= (MVs(18) - uLO_C2H6) /(uHI_C2H6 -uLO_C2H6)-K_C2H6*act_C2H6*((SP_C2H6-yLO_C2H6)/(yHI_C2H6-
yLO_C2H6)-yyy_C2H6(Ntau_C2H6));
%-------------------------------------------------------------------------------
%controller initialization
%MV19: bypass - FEHE Cold Exit Temp
SP_FEHE=134;
K_FEHE=5;
Ti_FEHE=10;
act_FEHE=1;
mode_FEHE=1;
Ponly_FEHE=0;
ht_FEHE=1/transmitter_sampling_frequency(19);
hc_FEHE=1/controller_sampling_frequency(19);
uLO_FEHE=0;
uHI_FEHE=1;
ded_FEHE=transmitter_deadtime(19);
tau_FEHE=transmitter_lag;
yLO_FEHE=0;
yHI_FEHE=200;
Nded_FEHE=1+transmitter_deadtime(19)*transmitter_sampling_frequency(19);
Ntau_FEHE=1;
```

```
xxx_FEHE(1)= (y_ss(8) -yLO_FEHE) / (yHI_FEHE -yLO_FEHE);
for i=2:Nded_FEHE
    xxx_FEHE(i)= xxx_FEHE(1);
end
for i=1:Ntau_FEHE
    yyy_FEHE(i)= xxx_FEHE(Nded_FEHE);
end
xi_FEHE= (MVs(19) - uLO_FEHE) /(uHI_FEHE -uLO_FEHE)-K_FEHE*act_FEHE*((SP_FEHE-yLO_FEHE)/(yHI_FEHE-
yLO_FEHE)-yyy_FEHE(Ntau_FEHE));
%-------------------------------------------------------------------------
%controller initialization
%MV20: LR - %H2O in bottom
SP_H2OCol=0.09344;
K_H2OCol=0.5;
Ti_H2OCol=60;
act_H2OCol=-1;
mode_H2OCol=1;
Ponly_H2OCol=0;
ht_H2OCol=1/transmitter_sampling_frequency(20);
hc_H2OCol=1/controller_sampling_frequency(20);
uLO_H2OCol=0;
uHI_H2OCol=7.56;
ded_H2OCol=transmitter_deadtime(20);
tau_H2OCol=transmitter_lag;
yLO_H2OCol=0;
yHI_H2OCol=0.2;
Nded_H2OCol=1+transmitter_deadtime(20)*transmitter_sampling_frequency(20);
Ntau_H2OCol=1;
xxx_H2OCol(1)= (y_ss(28) -yLO_H2OCol) / (yHI_H2OCol -yLO_H2OCol);
for i=2:Nded_H2OCol
    xxx_H2OCol(i)= xxx_H2OCol(1);
end
for i=1:Ntau_H2OCol
    yyy_H2OCol(i)= xxx_H2OCol(Nded_H2OCol);
end
xi_H2OCol= (MVs(20) - uLO_H2OCol) /(uHI_H2OCol -uLO_H2OCol)-K_H2OCol*act_H2OCol*((SP_H2OCol-
yLO_H2OCol)/(yHI_H2OCol-yLO_H2OCol)-yyy_H2OCol(Ntau_H2OCol));
%-------------------------------------------------------------------------
%controller initialization
%MV21: Qr - Temp
SP_TempCol=110;
K_TempCol=20;
Ti_TempCol=30;
act_TempCol=1;
mode_TempCol=1;
Ponly_TempCol=0;
ht_TempCol=1/transmitter_sampling_frequency(21);
hc_TempCol=1/controller_sampling_frequency(21);
uLO_TempCol=0;
uHI_TempCol=100000;
ded_TempCol=transmitter_deadtime(21);
tau_TempCol=transmitter_lag;
yLO_TempCol=0;
yHI_TempCol=120;
Nded_TempCol=1+transmitter_deadtime(21)*transmitter_sampling_frequency(21);
Ntau_TempCol=1;
xxx_TempCol(1)= (y_ss(22) -yLO_TempCol) / (yHI_TempCol -yLO_TempCol);
for i=2:Nded_TempCol
    xxx_TempCol(i)= xxx_TempCol(1);
end
for i=1:Ntau_TempCol
    yyy_TempCol(i)= xxx_TempCol(Nded_TempCol);
end
xi_TempCol= (MVs(21) - uLO_TempCol) /(uHI_TempCol -uLO_TempCol)-K_TempCol*act_TempCol*((SP_TempCol-
yLO_TempCol)/(yHI_TempCol-yLO_TempCol)-yyy_TempCol(Ntau_TempCol));
%-------------------------------------------------------------------------
%controller initialization
%MV22: Q_Condenser - Decanter Temperature (45.845 degC) is perfetly controlled in the code
SP_DecanterTemp=45.845;
K_DecanterTemp=1;
Ti_DecanterTemp=5;
```

```
act_DecanterTemp=-1;
mode_DecanterTemp=1;
Ponly_DecanterTemp=0;
ht_DecanterTemp=1/transmitter_sampling_frequency(22);
hc_DecanterTemp=1/controller_sampling_frequency(22);
uLO_DecanterTemp=0;
uHI_DecanterTemp=150000;
ded_DecanterTemp=transmitter_deadtime(22);
tau_DecanterTemp=transmitter_lag;
yLO_DecanterTemp=40;
yHI_DecanterTemp=50;
Nded_DecanterTemp=1+transmitter_deadtime(22)*transmitter_sampling_frequency(25);
Ntau_DecanterTemp=1;
xxx_DecanterTemp(1)= (y_ss(20) -yLO_DecanterTemp) / (yHI_DecanterTemp -yLO_DecanterTemp);
for i=2:Nded_DecanterTemp
    xxx_DecanterTemp(i)= xxx_DecanterTemp(1);
end
for i=1:Ntau_DecanterTemp
    yyy_DecanterTemp(i)= xxx_DecanterTemp(Nded_DecanterTemp);
end
xi_DecanterTemp= (MVs(22) - uLO_DecanterTemp) /(uHI_DecanterTemp -uLO_DecanterTemp)-
K_DecanterTemp*act_DecanterTemp*((SP_DecanterTemp-yLO_DecanterTemp)/(yHI_DecanterTemp-
yLO_DecanterTemp)-yyy_DecanterTemp(Ntau_DecanterTemp));
%-------------------------------------------------------------------------------
%controller initialization
%MV22: Organic Level is controlled by Organic Product Flow at 0.5
SP_Organic=0.5;
K_Organic=1;
Ti_Organic=100;
act_Organic=-1;
mode_Organic=1;
Ponly_Organic=1;
ht_Organic=1/transmitter_sampling_frequency(23);
hc_Organic=1/controller_sampling_frequency(23);
uLO_Organic=0;
uHI_Organic=2.4;
ded_Organic=transmitter_deadtime(23);
tau_Organic=transmitter_lag;
yLO_Organic=0;
yHI_Organic=1;
Nded_Organic=1+transmitter_deadtime(23)*transmitter_sampling_frequency(23);
Ntau_Organic=1;
xxx_Organic(1)= (y_ss(18)-yLO_Organic) / (yHI_Organic -yLO_Organic);
for i=2:Nded_Organic
    xxx_Organic(i)= xxx_Organic(1);
end
for i=1:Ntau_Organic
    yyy_Organic(i)= xxx_Organic(Nded_Organic);
end
xi_Organic= (MVs(23) - uLO_Organic) /(uHI_Organic -uLO_Organic)-K_Organic*act_Organic*((SP_Organic-
yLO_Organic)/(yHI_Organic-yLO_Organic)-yyy_Organic(Ntau_Organic));
%-------------------------------------------------------------------------------
%controller initialization
%MV23: Aqueous Level is controlled by Aqueous Product Flow at 0.5
SP_Aqueous=0.5;
K_Aqueous=1;
Ti_Aqueous=100;
act_Aqueous=-1;
mode_Aqueous=1;
Ponly_Aqueous=1;
ht_Aqueous=1/transmitter_sampling_frequency(24);
hc_Aqueous=1/controller_sampling_frequency(24);
uLO_Aqueous=0;
uHI_Aqueous=2.4;
ded_Aqueous=transmitter_deadtime(24);
tau_Aqueous=transmitter_lag;
yLO_Aqueous=0;
yHI_Aqueous=1;
Nded_Aqueous=1+transmitter_deadtime(24)*transmitter_sampling_frequency(24);
Ntau_Aqueous=1;
xxx_Aqueous(1)= (y_ss(19)-yLO_Aqueous) / (yHI_Aqueous -yLO_Aqueous);
```

```
for i=2:Nded_Aqueous
    xxx_Aqueous(i)= xxx_Aqueous(1);
end
for i=1:Ntau_Aqueous
    yyy_Aqueous(i)= xxx_Aqueous(Nded_Aqueous);
end
xi_Aqueous= (MVs(24) - uLO_Aqueous) /(uHI_Aqueous -uLO_Aqueous)-K_Aqueous*act_Aqueous*((SP_Aqueous-
yLO_Aqueous)/(yHI_Aqueous-yLO_Aqueous)-yyy_Aqueous(Ntau_Aqueous));
%-------------------------------------------------------------------------
%MV24: Column Bottom Exit is used to control the Column Bottom Level
SP_ColButtom=0.5;
K_ColButtom=1;
Ti_ColButtom=100;
act_ColButtom=-1;
mode_ColButtom=1;
Ponly_ColButtom=1;
ht_ColButtom=1/transmitter_sampling_frequency(25);
hc_ColButtom=1/controller_sampling_frequency(25);
uLO_ColButtom=0;
uHI_ColButtom=4.536;
ded_ColButtom=transmitter_deadtime(25);
tau_ColButtom=transmitter_lag;
yLO_ColButtom=0;
yHI_ColButtom=1;
Nded_ColButtom=1+transmitter_deadtime(25)*transmitter_sampling_frequency(25);
Ntau_ColButtom=1;
xxx_ColButtom(1)= (y_ss(21)-yLO_ColButtom) / (yHI_ColButtom -yLO_ColButtom);
for i=2:Nded_ColButtom
    xxx_ColButtom(i)= xxx_ColButtom(1);
end
for i=1:Ntau_ColButtom
    yyy_ColButtom(i)= xxx_ColButtom(Nded_ColButtom);
end
xi_ColButtom= (MVs(25) - uLO_ColButtom) /(uHI_ColButtom -uLO_ColButtom)-
K_ColButtom*act_ColButtom*((SP_ColButtom-yLO_ColButtom)/(yHI_ColButtom-yLO_ColButtom)-
yyy_ColButtom(Ntau_ColButtom));
%-------------------------------------------------------------------------
%controller initialization
%MV26: Vaporizer Liquid Inlet is fixed at 2.1924 kmol/min
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
%initialize historical data vectors
x_history=[];
y_history=[];
u_history=[];
dx_history=[];
%initialize simulation parameters
tic
timer=0;
is_initial=0;
disturbance_ID=0;
%start simulation
for k=0:(model_sampling_frequency*minute)
    time=k/model_sampling_frequency; %in minute
    %initialize disturbances
    if (k/model_sampling_frequency)>=10
        switch selected_ID
        case 0
            ;
        case 1
            SP_TRCT=151;
        case 2
            SP_TRCT=165;
        case 3
            SP_H2OCol=0.18;
        case 4
            MVs(26)=2.64;
        case 5
            if (k/model_sampling_frequency)<15
                MVs(3)=0;
```

```
            end
        case 6
            if (k/model_sampling_frequency)<15
                MVs(1)=0;
            end
        case 7
             disturbance_ID=1;
        case 8
            if (k/model_sampling_frequency)<15
                disturbance_ID=2;
            else
                disturbance_ID=0;
            end
        end
    end
    %calculate state derivatives and measurements
    [dstatedt,states,MVs,y_ss]=VAModel(states,MVs,time,is_initial,disturbance_ID);
    %in this example, we assume the setpoint of the vaporizer pressure control is given by the current
gas recycle pressure
    SP_PresVap=y_ss(12);
    %command window display refreshing rate
    if rem(k,display_sampling_frequency*model_sampling_frequency)==0
        [error, indx]=max(abs(dstatedt./states));
%        info=['minute: ' num2str(time) '    max_dxdt_error: ' num2str(error) '    location: '
num2str(indx)];
 %       disp(info);
    end
    %save information
    if rem(k,storage_sampling_frequency*model_sampling_frequency)==0
        x_history=[x_history;states'];

y_history=[y_history;[y_ss(37);y_ss(12);y_ss(23);y_ss(2);y_ss(1);y_ss(4);y_ss(5);y_ss(9);y_ss(10);MVs(1
0);y_ss(11);y_ss(13);MVs(13);y_ss(14);MVs(15);y_ss(15);y_ss(31);y_ss(33);y_ss(8);y_ss(28);y_ss(22);y_ss
(20);y_ss(18);y_ss(19);y_ss(21);MVs(26);y_ss(27)]'];
        u_history=[u_history;MVs'];
        dx_history=[dx_history;dstatedt'];
    end
    %update states
    states=states+(1/model_sampling_frequency)*dstatedt;
    %controller action
    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(1))==0 & k~=0
        [xxx_O2, yyy_O2]=Transmit(y_ss(37), xxx_O2, yyy_O2, ded_O2, tau_O2, Nded_O2, Ntau_O2, ht_O2,
yLO_O2, yHI_O2);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(1))==0 & k~=0
        [MVs(1),xi_O2,flag_O2]=controller(SP_O2, yyy_O2(Ntau_O2), MVs(1), xi_O2, mode_O2, K_O2, Ti_O2,
hc_O2,...
            yLO_O2, yHI_O2, uLO_O2, uHI_O2, act_O2, Ponly_O2, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(2))==0 & k~=0
        [xxx_C2H4, yyy_C2H4]=Transmit(y_ss(12), xxx_C2H4, yyy_C2H4, ded_C2H4, tau_C2H4, Nded_C2H4,
Ntau_C2H4, ht_C2H4, yLO_C2H4, yHI_C2H4);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(2))==0 & k~=0
        [MVs(2),xi_C2H4,flag_C2H4]=controller(SP_C2H4, yyy_C2H4(Ntau_C2H4), MVs(2), xi_C2H4, mode_C2H4,
K_C2H4, Ti_C2H4, hc_C2H4,...
            yLO_C2H4, yHI_C2H4, uLO_C2H4, uHI_C2H4, act_C2H4, Ponly_C2H4, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(3))==0 & k~=0
        [xxx_HAc, yyy_HAc]=Transmit(y_ss(23), xxx_HAc, yyy_HAc, ded_HAc, tau_HAc, Nded_HAc, Ntau_HAc,
ht_HAc, yLO_HAc, yHI_HAc);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(3))==0 & k~=0
        [MVs(3),xi_HAc,flag_HAc]=controller(SP_HAc, yyy_HAc(Ntau_HAc), MVs(3), xi_HAc, mode_HAc, K_HAc,
Ti_HAc, hc_HAc,...
            yLO_HAc, yHI_HAc, uLO_HAc, uHI_HAc, act_HAc, Ponly_HAc, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(4))==0 & k~=0
```

```
        [xxx_LevelVap, yyy_LevelVap]=Transmit(y_ss(2), xxx_LevelVap, yyy_LevelVap, ded_LevelVap,
tau_LevelVap, Nded_LevelVap, Ntau_LevelVap, ht_LevelVap, yLO_LevelVap, yHI_LevelVap);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(4))==0 & k~=0
        [MVs(4),xi_LevelVap,flag_LevelVap]=controller(SP_LevelVap, yyy_LevelVap(Ntau_LevelVap), MVs(4),
xi_LevelVap, mode_LevelVap, K_LevelVap, Ti_LevelVap, hc_LevelVap,...
            yLO_LevelVap, yHI_LevelVap, uLO_LevelVap, uHI_LevelVap, act_LevelVap, Ponly_LevelVap, 0, 0,
0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(5))==0 & k~=0
        [xxx_PresVap, yyy_PresVap]=Transmit(y_ss(1), xxx_PresVap, yyy_PresVap, ded_PresVap,
tau_PresVap, Nded_PresVap, Ntau_PresVap, ht_PresVap, yLO_PresVap, yHI_PresVap);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(5))==0 & k~=0
        [MVs(5),xi_PresVap,flag_PresVap]=controller(SP_PresVap, yyy_PresVap(Ntau_PresVap), MVs(5),
xi_PresVap, mode_PresVap, K_PresVap, Ti_PresVap, hc_PresVap,...
            yLO_PresVap, yHI_PresVap, uLO_PresVap, uHI_PresVap, act_PresVap, Ponly_PresVap, 0, 0, 0,
0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(6))==0 & k~=0
        [xxx_ReactorInletTemp, yyy_ReactorInletTemp]=Transmit(y_ss(4), xxx_ReactorInletTemp,
yyy_ReactorInletTemp, ded_ReactorInletTemp, tau_ReactorInletTemp, Nded_ReactorInletTemp,
Ntau_ReactorInletTemp, ht_ReactorInletTemp, yLO_ReactorInletTemp, yHI_ReactorInletTemp);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(6))==0 & k~=0
        [MVs(6),xi_ReactorInletTemp,flag_ReactorInletTemp]=controller(SP_ReactorInletTemp,
yyy_ReactorInletTemp(Ntau_ReactorInletTemp), MVs(6), xi_ReactorInletTemp, mode_ReactorInletTemp,
K_ReactorInletTemp, Ti_ReactorInletTemp, hc_ReactorInletTemp,...
            yLO_ReactorInletTemp, yHI_ReactorInletTemp, uLO_ReactorInletTemp, uHI_ReactorInletTemp,
act_ReactorInletTemp, Ponly_ReactorInletTemp, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(7))==0 & k~=0
        [xxx_TRCT, yyy_TRCT]=Transmit(y_ss(5), xxx_TRCT, yyy_TRCT, ded_TRCT, tau_TRCT, Nded_TRCT,
Ntau_TRCT, ht_TRCT, yLO_TRCT, yHI_TRCT);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(7))==0 & k~=0
        [MVs(7),xi_TRCT,flag_TRCT]=controller(SP_TRCT, yyy_TRCT(Ntau_TRCT), MVs(7), xi_TRCT, mode_TRCT,
K_TRCT, Ti_TRCT, hc_TRCT,...
            yLO_TRCT, yHI_TRCT, uLO_TRCT, uHI_TRCT, act_TRCT, Ponly_TRCT, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(8))==0 & k~=0
        [xxx_LevelSep, yyy_LevelSep]=Transmit(y_ss(9), xxx_LevelSep, yyy_LevelSep, ded_LevelSep,
tau_LevelSep, Nded_LevelSep, Ntau_LevelSep, ht_LevelSep, yLO_LevelSep, yHI_LevelSep);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(8))==0 & k~=0
        [MVs(8),xi_LevelSep,flag_LevelSep]=controller(SP_LevelSep, yyy_LevelSep(Ntau_LevelSep), MVs(8),
xi_LevelSep, mode_LevelSep, K_LevelSep, Ti_LevelSep, hc_LevelSep,...
            yLO_LevelSep, yHI_LevelSep, uLO_LevelSep, uHI_LevelSep, act_LevelSep, Ponly_LevelSep, 0, 0,
0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(9))==0 & k~=0
        [xxx_TempSep, yyy_TempSep]=Transmit(y_ss(10), xxx_TempSep, yyy_TempSep, ded_TempSep,
tau_TempSep, Nded_TempSep, Ntau_TempSep, ht_TempSep, yLO_TempSep, yHI_TempSep);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(9))==0 & k~=0
        [MVs(9),xi_TempSep,flag_TempSep]=controller(SP_TempSep, yyy_TempSep(Ntau_TempSep), MVs(9),
xi_TempSep, mode_TempSep, K_TempSep, Ti_TempSep, hc_TempSep,...
            yLO_TempSep, yHI_TempSep, uLO_TempSep, uHI_TempSep, act_TempSep, Ponly_TempSep, 0, 0, 0,
0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(11))==0 & k~=0
        [xxx_CompOutletTemp, yyy_CompOutletTemp]=Transmit(y_ss(11), xxx_CompOutletTemp,
yyy_CompOutletTemp, ded_CompOutletTemp, tau_CompOutletTemp, Nded_CompOutletTemp, Ntau_CompOutletTemp,
ht_CompOutletTemp, yLO_CompOutletTemp, yHI_CompOutletTemp);
    end
```

```
    if rem(k,model_sampling_frequency/controller_sampling_frequency(11))==0 & k~=0
        [MVs(11),xi_CompOutletTemp,flag_CompOutletTemp]=controller(SP_CompOutletTemp,
yyy_CompOutletTemp(Ntau_CompOutletTemp), MVs(11), xi_CompOutletTemp, mode_CompOutletTemp,
K_CompOutletTemp, Ti_CompOutletTemp, hc_CompOutletTemp,...
            yLO_CompOutletTemp, yHI_CompOutletTemp, uLO_CompOutletTemp, uHI_CompOutletTemp,
act_CompOutletTemp, Ponly_CompOutletTemp, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(12))==0 & k~=0
        [xxx_LevelAbs, yyy_LevelAbs]=Transmit(y_ss(13), xxx_LevelAbs, yyy_LevelAbs, ded_LevelAbs,
tau_LevelAbs, Nded_LevelAbs, Ntau_LevelAbs, ht_LevelAbs, yLO_LevelAbs, yHI_LevelAbs);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(12))==0 & k~=0
        [MVs(12),xi_LevelAbs,flag_LevelAbs]=controller(SP_LevelAbs, yyy_LevelAbs(Ntau_LevelAbs),
MVs(12), xi_LevelAbs, mode_LevelAbs, K_LevelAbs, Ti_LevelAbs, hc_LevelAbs,...
            yLO_LevelAbs, yHI_LevelAbs, uLO_LevelAbs, uHI_LevelAbs, act_LevelAbs, Ponly_LevelAbs, 0, 0,
0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(14))==0 & k~=0
        [xxx_CircOutletTemp, yyy_CircOutletTemp]=Transmit(y_ss(14), xxx_CircOutletTemp,
yyy_CircOutletTemp, ded_CircOutletTemp, tau_CircOutletTemp, Nded_CircOutletTemp, Ntau_CircOutletTemp,
ht_CircOutletTemp, yLO_CircOutletTemp, yHI_CircOutletTemp);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(14))==0 & k~=0
        [MVs(14),xi_CircOutletTemp,flag_CircOutletTemp]=controller(SP_CircOutletTemp,
yyy_CircOutletTemp(Ntau_CircOutletTemp), MVs(14), xi_CircOutletTemp, mode_CircOutletTemp,
K_CircOutletTemp, Ti_CircOutletTemp, hc_CircOutletTemp,...
            yLO_CircOutletTemp, yHI_CircOutletTemp, uLO_CircOutletTemp, uHI_CircOutletTemp,
act_CircOutletTemp, Ponly_CircOutletTemp, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(16))==0 & k~=0
        [xxx_ScrubOutletTemp, yyy_ScrubOutletTemp]=Transmit(y_ss(15), xxx_ScrubOutletTemp,
yyy_ScrubOutletTemp, ded_ScrubOutletTemp, tau_ScrubOutletTemp, Nded_ScrubOutletTemp,
Ntau_ScrubOutletTemp, ht_ScrubOutletTemp, yLO_ScrubOutletTemp, yHI_ScrubOutletTemp);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(16))==0 & k~=0
        [MVs(16),xi_ScrubOutletTemp,flag_ScrubOutletTemp]=controller(SP_ScrubOutletTemp,
yyy_ScrubOutletTemp(Ntau_ScrubOutletTemp), MVs(16), xi_ScrubOutletTemp, mode_ScrubOutletTemp,
K_ScrubOutletTemp, Ti_ScrubOutletTemp, hc_ScrubOutletTemp,...
            yLO_ScrubOutletTemp, yHI_ScrubOutletTemp, uLO_ScrubOutletTemp, uHI_ScrubOutletTemp,
act_ScrubOutletTemp, Ponly_ScrubOutletTemp, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(17))==0 & k~=0
        [xxx_CO2, yyy_CO2]=Transmit(y_ss(31), xxx_CO2, yyy_CO2, ded_CO2, tau_CO2, Nded_CO2, Ntau_CO2,
ht_CO2, yLO_CO2, yHI_CO2);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(17))==0 & k~=0
        [MVs(17),xi_CO2,flag_CO2]=controller(SP_CO2, yyy_CO2(Ntau_CO2), MVs(17), xi_CO2, mode_CO2,
K_CO2, Ti_CO2, hc_CO2,...
            yLO_CO2, yHI_CO2, uLO_CO2, uHI_CO2, act_CO2, Ponly_CO2, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(18))==0 & k~=0
        [xxx_C2H6, yyy_C2H6]=Transmit(y_ss(33), xxx_C2H6, yyy_C2H6, ded_C2H6, tau_C2H6, Nded_C2H6,
Ntau_C2H6, ht_C2H6, yLO_C2H6, yHI_C2H6);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(18))==0 & k~=0
        [MVs(18),xi_C2H6,flag_C2H6]=controller(SP_C2H6, yyy_C2H6(Ntau_C2H6), MVs(18), xi_C2H6,
mode_C2H6, K_C2H6, Ti_C2H6, hc_C2H6,...
            yLO_C2H6, yHI_C2H6, uLO_C2H6, uHI_C2H6, act_C2H6, Ponly_C2H6, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(19))==0 & k~=0
        [xxx_FEHE, yyy_FEHE]=Transmit(y_ss(8), xxx_FEHE, yyy_FEHE, ded_FEHE, tau_FEHE, Nded_FEHE,
Ntau_FEHE, ht_FEHE, yLO_FEHE, yHI_FEHE);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(19))==0 & k~=0
```

```
        [MVs(19),xi_FEHE,flag_FEHE]=controller(SP_FEHE, yyy_FEHE(Ntau_FEHE), MVs(19), xi_FEHE,
mode_FEHE, K_FEHE, Ti_FEHE, hc_FEHE,...
            yLO_FEHE, yHI_FEHE, uLO_FEHE, uHI_FEHE, act_FEHE, Ponly_FEHE, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(20))==0 & k~=0
        [xxx_H2OCol, yyy_H2OCol]=Transmit(y_ss(28), xxx_H2OCol, yyy_H2OCol, ded_H2OCol, tau_H2OCol,
Nded_H2OCol, Ntau_H2OCol, ht_H2OCol, yLO_H2OCol, yHI_H2OCol);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(20))==0 & k~=0
        [MVs(20),xi_H2OCol,flag_H2OCol]=controller(SP_H2OCol, yyy_H2OCol(Ntau_H2OCol), MVs(20),
xi_H2OCol, mode_H2OCol, K_H2OCol, Ti_H2OCol, hc_H2OCol,...
            yLO_H2OCol, yHI_H2OCol, uLO_H2OCol, uHI_H2OCol, act_H2OCol, Ponly_H2OCol, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(21))==0 & k~=0
        [xxx_TempCol, yyy_TempCol]=Transmit(y_ss(22), xxx_TempCol, yyy_TempCol, ded_TempCol,
tau_TempCol, Nded_TempCol, Ntau_TempCol, ht_TempCol, yLO_TempCol, yHI_TempCol);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(21))==0 & k~=0
        [MVs(21),xi_TempCol,flag_TempCol]=controller(SP_TempCol, yyy_TempCol(Ntau_TempCol), MVs(21),
xi_TempCol, mode_TempCol, K_TempCol, Ti_TempCol, hc_TempCol,...
            yLO_TempCol, yHI_TempCol, uLO_TempCol, uHI_TempCol, act_TempCol, Ponly_TempCol, 0, 0, 0,
0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(22))==0 & k~=0
        [xxx_DecanterTemp, yyy_DecanterTemp]=Transmit(y_ss(20), xxx_DecanterTemp, yyy_DecanterTemp,
ded_DecanterTemp, tau_DecanterTemp, Nded_DecanterTemp, Ntau_DecanterTemp, ht_DecanterTemp,
yLO_DecanterTemp, yHI_DecanterTemp);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(22))==0 & k~=0
        [MVs(22),xi_DecanterTemp,flag_DecanterTemp]=controller(SP_DecanterTemp,
yyy_DecanterTemp(Ntau_DecanterTemp), MVs(22), xi_DecanterTemp, mode_DecanterTemp, K_DecanterTemp,
Ti_DecanterTemp, hc_DecanterTemp,...
            yLO_DecanterTemp, yHI_DecanterTemp, uLO_DecanterTemp, uHI_DecanterTemp, act_DecanterTemp,
Ponly_DecanterTemp, 0, 0, 0, 0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(23))==0 & k~=0
        [xxx_Organic, yyy_Organic]=Transmit(y_ss(18), xxx_Organic, yyy_Organic, ded_Organic,
tau_Organic, Nded_Organic, Ntau_Organic, ht_Organic, yLO_Organic, yHI_Organic);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(23))==0 & k~=0
        [MVs(23),xi_Organic,flag_Organic]=controller(SP_Organic, yyy_Organic(Ntau_Organic), MVs(23),
xi_Organic, mode_Organic, K_Organic, Ti_Organic, hc_Organic,...
            yLO_Organic, yHI_Organic, uLO_Organic, uHI_Organic, act_Organic, Ponly_Organic, 0, 0, 0,
0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(24))==0 & k~=0
        [xxx_Aqueous, yyy_Aqueous]=Transmit(y_ss(19), xxx_Aqueous, yyy_Aqueous, ded_Aqueous,
tau_Aqueous, Nded_Aqueous, Ntau_Aqueous, ht_Aqueous, yLO_Aqueous, yHI_Aqueous);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(24))==0 & k~=0
        [MVs(24),xi_Aqueous,flag_Aqueous]=controller(SP_Aqueous, yyy_Aqueous(Ntau_Aqueous), MVs(24),
xi_Aqueous, mode_Aqueous, K_Aqueous, Ti_Aqueous, hc_Aqueous,...
            yLO_Aqueous, yHI_Aqueous, uLO_Aqueous, uHI_Aqueous, act_Aqueous, Ponly_Aqueous, 0, 0, 0,
0);
    end

    if rem(k,model_sampling_frequency/transmitter_sampling_frequency(25))==0 & k~=0
        [xxx_ColButtom, yyy_ColButtom]=Transmit(y_ss(21), xxx_ColButtom, yyy_ColButtom, ded_ColButtom,
tau_ColButtom, Nded_ColButtom, Ntau_ColButtom, ht_ColButtom, yLO_ColButtom, yHI_ColButtom);
    end
    if rem(k,model_sampling_frequency/controller_sampling_frequency(25))==0 & k~=0
        [MVs(25),xi_ColButtom,flag_ColButtom]=controller(SP_ColButtom, yyy_ColButtom(Ntau_ColButtom),
MVs(25), xi_ColButtom, mode_ColButtom, K_ColButtom, Ti_ColButtom, hc_ColButtom,...
            yLO_ColButtom, yHI_ColButtom, uLO_ColButtom, uHI_ColButtom, act_ColButtom, Ponly_ColButtom,
0, 0, 0, 0);
    end
```

```
end
toc
%plot graphics
my_label=['   %O2  ';'  Pres  ';'  HAc-L ';'  Vap-L ';'  Vap-P ';'  Pre-T ';'  RCT-T ';'  Sep-L ';'
Sep-T ';'  Sep-V ';'  Com-T ';'  Abs-L ';'  Cir-F ';'  Cir-T ';'  Scr-F ';'  Scr-T ';'  %CO2  ';'
%C2H6 ';' FEHE-T ';'  %H2O  ';'  Col-T ';'  Org-L ';'  Aqu-L ';'  Col-L ';' Vap-In ';' Dect-T ';'%VAc
E-3'];
MV_label=[' F-O2 ';'F-C2H4';' F-HAc';' Q-Vap';' F-Vap';'Q-Heat';'ShellT';'F-SepL';' T-Sep';'F-SepV';'Q-
Comp';'F-AbsL';'F-Circ';'Q-Circ';'F-Scru';'Q-Scru';' F-CO2';' Purge';'bypass';'Reflux';'Q-Rebo';'F-
Orga';'F-Aque';' F-Bot';'Q_Cond';'F-Tank'];
setpoint=[SP_O2;SP_C2H4;SP_HAc;SP_LevelVap;SP_PresVap;150;SP_TRCT;SP_LevelSep;SP_TempSep;16.1026;80;SP_
LevelAbs;15.1198;25;0.756;25;SP_CO2;SP_C2H6;SP_FEHE;SP_H2OCol;SP_TempCol;0.5;0.5;SP_ColButtom;45.845;2.
1924];
save normal.mat
warning off
if selected_ID==1
    McAvoy_Plot_1(y_history,u_history,setpoint,my_label,MV_label,storage_sampling_frequency);
elseif selected_ID==2
    McAvoy_Plot_2(y_history,u_history,setpoint,my_label,MV_label,storage_sampling_frequency);
elseif selected_ID==3
    McAvoy_Plot_3(y_history,u_history,setpoint,my_label,MV_label,storage_sampling_frequency);
elseif selected_ID==4
    McAvoy_Plot_4(y_history,u_history,setpoint,my_label,MV_label,storage_sampling_frequency);
else
    Transient_Plot(y_history,u_history,setpoint,my_label,MV_label,storage_sampling_frequency);
end
return
```

# APPENDIX C

## M.FILE FOR MODEL PREDICTIVE CONTROL MODEL