INTELLIGENT PARAMETER ADAPTATION FOR CHEMICAL PROCESSES

John C. Sozio

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

July 8, 1999
Blacksburg, Virginia

Keywords: Tennessee Eastman, Genetic Algorithm,
Decentralized Process Control, Fuzzy Logic

Intelligent Parameter Adaptation for Chemical Processes

John C. Sozio

(ABSTRACT)

Reducing the operating costs of chemical processes is very beneficial in decreasing a company's bottom line numbers. Since chemical processes are usually run in steady-state for long periods of time, saving a few dollars an hour can have significant long term effects. However, the complexity involved in most chemical processes from nonlinear dynamics makes them difficult processes to optimize. A nonlinear, open-loop unstable system, called the Tennessee Eastman Chemical Process Control Problem, is used as a test-bed problem for minimization routines. A decentralized controller is first developed that stabilizes the plant to setpoint changes and disturbances.

Subsequently, a genetic algorithm calculates input parameters of the decentralized controller for minimum operating cost performance. Genetic algorithms use a directed search method based on the evolutionary principle of "survival of the fittest". They are powerful global optimization tools; however, they are typically computationally expensive and have long convergence times. To decrease the convergence time and avoid premature convergence to a local minimum solution, an auxiliary fuzzy logic controller was used to adapt the parameters of the genetic algorithm. The controller manipulates the input and output data through a set of linguistic IF-THEN rules to respond in a manner similar to human reasoning. The combination of a supervisory fuzzy controller and a genetic algorithm leads to near-optimum operating costs for a dynamically modeled chemical process.

Acknowledgements

I would like to thank all of my committee members for providing support and background knowledge required to complete this project. I would especially like to thank Dr. VanLandingham for his patience in my progress and allowing me to use up a lot of his time. His suggestions really assisted me in finishing this thesis, especially when suggesting methods to avoid. I believe I have done enough process control projects to realize it is not what I want to do for a living right now. One of Dr. VanLandingham's graduate students, Farooq Azam, also deserves thanks for recommending suggestions and sharing his knowledge about simulation techniques. I would like to thank Dr. Rony for introducing me to the Tennessee Eastman process and some current research in the field. This project made me realize some of the complexities that can be involved in chemical processes, and why researchers recommend simple control structures. Unfortunately, the introductory chemical engineering class did not help as much as hoped.

I must also thank my parents, Karen and Francesco Sozio for their continued support over the years. I don't think I could have gotten this far without their backing and encouragement. I have to also thank my siblings, Stephen and Jackie, for pressuring me to attain more schooling than they have completed. I have to also thank my friends, those in Maryland and Virginia, for understanding when I took a hiatus from my normal life to complete this thesis.

Table of Contents

Abstract ii
Acknowledgements iii
Table of Contents iv
List of Figures vii
List of Tables ix

CHAPTER 1 INTRODUCTION TO THE TENNESSEE EASTMAN CHEMICAL PROCESS 1

1.1 History of Chemical Process Control 1
1.2 Features of the Tennessee Eastman Process 2
1.2.1 Chemical Reactions in the System 5
1.2.2 Modes of Plant Operation 5
1.2.3 Control Objectives of the TE Process 6
1.2.4 Plant Assumptions 8
1.3 Features of the FORTRAN Code 8
1.3.1 Concealing Proprietary Plant Information 9
1.3.2 Inputs and Outputs of the System 9
1.3.3 Measurement Noise and Plant Shutdown 10
1.3.4 Plant Operating Costs 11
1.4 Simulating the System 12
1.4.1 State Integration Routine 12
1.4.2 MATLAB MEX Interface 13
1.5 Research Objectives 13
1.6 Thesis Organization 14

CHAPTER 2 CONTROL AND OPTIMIZATION OF THE TENNESSEE EASTMAN PROCESS 15

2.1 Review of Decentralized Control Structures 15
2.1.1 Decentralized Controller of Luyben 16
2.1.2 Decentralized Controller of McAvoy and Ye 17
2.1.3 Decentralized Controller of Banerjee and Arkun 17
2.1.4 Decentralized Controller of Lyman and Georgakis 18
2.1.5 Decentralized Controller of Ricker 19
2.2 Optimal Operating Costs of the Base-Case System 20
2.2.1 Formulation of the Optimal Cost Solution 23
2.2.1 Optimal Cost from Analysis by Ricker 24
2.2.2 Comparison of the SOLNP and MINOS Programs on the TE Process 26
2.2.3 Minimum Operating Costs with Fewer Constraints 26
2.3 Conclusions 27

CHAPTER 3 THE STABILIZING DECENTRALIZED CONTROLLER 28

3.1 The PID Controller in Chemical Process Control 28

3.1.1 Derivation of the PID Equation 29

3.1.2 Use of the PID Equation in Chemical Processes 30

3.1.3 Calculating Controller PI Gains 31

3.2 Development of the PI Decentralized Controller 31

3.2.1 Design of the Inner Loop Cascade Controllers Around the Base-Case 32

3.2.2 Loop Pairings for Level Controllers 38

3.2.3 Design of the Outer Loop Cascade Controllers Around the Base-Case 39

3.2.3.1 Material Balance and Mass Transfer 40

3.2.3.2 Loop Pairings Through the Relative Gain Array (RGA) 41

3.2.3.3 Tuning Procedure for PI controllers 44

3.2.4 Performance Validation of the Decentralized Control System 50

3.3 Overview of the Decentralized Control Scheme as Applied to the TE Process 59

CHAPTER 4 GENETIC ALGORITHM IMPLEMENTATION 60

4.1 Beginnings of the Genetic Algorithm 60

4.2 Parameters of the Genetic Algorithm 62

4.2.1 Creation of the Initial Population 64

4.2.2 Reproduction Operator of the Genetic Algorithm 64

4.2.3 Crossover Operator of the Genetic Algorithm 66

4.2.4 Mutation Operator of the Genetic Algorithm 67

4.2.5 Other Performance Parameters of the Genetic Algorithm 68

4.3 Application of the GA to the TE Process 68

4.3.1 Available Degrees of Freedom in the TE Process 69

4.3.2 Converting Genetic Algorithm Data to Input Data 72

4.3.3 Crossover Sites for the TE Process 73

4.3.4 Mutation Sites for the TE Process 73

4.3.5 Variations in the Simulation Procedure for the Genetic Algorithm 74

4.4 Results and Conclusions from GA Simulations 75

4.4.1 Genetic Algorithm Results with a Single Crossover Site 77

4.4.2 Genetic Algorithm Results with Two Crossover Sites 77

4.4.3 Genetic Algorithm Results with a Single Crossover Point Per Variable 78

4.4.4 Conclusions from Genetic Algorithm Operation on the TE Process 84

CHAPTER 5 FUZZY LOGIC AND ITS APPLICATION TO THE GENETIC ALGORITHM 85

5.1 Introduction to Fuzzy Logic 85

5.1.1 Fuzzy Membership Sets 87

5.1.2 Decision Making Logic Connectives 88

5.1.3 Fuzzy Relation Operator 90

5.1.4 Defuzzification Techniques 93

5.2 Fuzzy Logic Applied to the Tennessee Eastman Problem 95

5.2.1 Rule Base for the Fuzzy Logic Controller 95

5.2.2 Implementing the Fuzzy Logic Rules *98*
5.3 Results and Conclusions 100
5.3.1 Comparing the Fuzzy Logic Controlled GA System to the Basic GA System *100*
5.3.2 Additional Fuzzy Parameters *101*

CHAPTER 6 RESULTS AND CONCLUSIONS 103

6.1 Performance Evaluation of the Controlled System 103
6.1.1 Parameters Adjusted to Conform to the Control Objectives *104*
6.1.2 Results from Disturbances and Setpoint Changes *109*
6.2 Conclusions 112
6.3 Future Work 113

REFERENCES 116

APPENDIX A COMPUTER CODE 118

VITA 134

List of Figures

Figure 1.1: Plant diagram of the Tennessee Eastman chemical process 4
Figure 1.2: Black box model of the TE process 9
Figure 1.3: Typical decentralized control loop........ 10
Figure 3.1: SISO closed loop system with PID controller 28
Figure 3.2: Cascaded decentralized system........ 32
Figure 3.3: Schematic of a decentralized control loop 35
Figure 3.4: Flowsheet with flow and temperature controllers........ 37
Figure 3.5: Example of the ATV algorithm 46
Figure 3.6: Plant diagram of decentralized controlled system 49
Figure 3.7: System inputs for change from 50/50 to 40/60 G/H ratio........ 51
Figure 3.8: Selected outputs for change from 50/50 to 40/60 G/H ratio........ 52
Figure 3.9: Selected outputs for change from 50/50 to 40/60 G/H ratio........ 53
Figure 3.10: Input signals for recommended setpoint changes 55
Figure 3.11: Selected outputs for recommended setpoint changes 56
Figure 3.12: Input signals with IDV(8) active........ 57
Figure 3.13: Selected outputs with IDV(8) active........ 58
Figure 4.1: Difficult first order problem for most optimization routines........ 61
Figure 4.2: Flow chart of the genetic algorithm routine........ 63
Figure 4.3: Defining genetic algorithm parameters........ 64
Figure 4.4: Roulette wheel for simple example with four individuals........ 65
Figure 4.5: Implementation of the reproduction operator in computer code........ 66
Figure 4.6: Example of the crossover operator 67
Figure 4.7: Example of the mutation operator 68
Figure 4.8: Determining the lower bound for the compressor power setpoint........ 71
Figure 4.9: Determining the upper bound for the compressor power setpoint........ 72
Figure 4.10: Interaction between GA, controller, and plant model........ 76
Figure 4.11: Genetic algorithm simulation results using a single crossover point........ 77
Figure 4.12: Genetic algorithm simulation results using two crossover points 78
Figure 4.13: Genetic algorithm simulation results using one crossover point per variable 79
Figure 4.14: System inputs for near-optimal case 81
Figure 4.15: Selected outputs for near-optimal case 82
Figure 4.16: Selected outputs for near-optimal case 83
Figure 5.1: Fuzzy logic procedure........ 86
Figure 5.2: Crisp membership function........ 87
Figure 5.3: Triangular shaped fuzzy set 88
Figure 5.4: Graphical interpretation of Mandami inference........ 92
Figure 5.5: Graphical interpretation of max-product rules........ 92
Figure 5.6: Output membership functions for Mandami and max-product inferences........ 93
Figure 5.7: Adaptive fuzzy controlled genetic algorithm system........ 96
Figure 5.8: Results from simple fuzzy controller........ 99
Figure 5.9: Input and output fuzzy sets for crossover operator........ 99
Figure 5.10: Input and output fuzzy sets for mutation operator........ 100

Figure 5.11: Genetic algorithm convergence to the minimum-cost solution 101
Figure 6.1: Inputs for the minimum-cost solution... 106
Figure 6.2: Selected outputs for the minimum-cost solution 107
Figure 6.3: Selected outputs for the minimum-cost solution 108
Figure 6.4: Input signals with IDV(8) active.. 110
Figure 6.5: Selected outputs with IDV(8) active.. 111

List of Tables

Table 1-1: Plant Operating Modes 6
Table 1-2: Disturbance Scenarios of the Tennessee Eastman Process 7
Table 1-3: Process Operating Constraints 10
Table 2-1: Output Values of the TE Process 21
Table 2-2: Input Values of the TE Process 22
Table 2-3: Costs per Hour of Operation of the TE Process 22
Table 3-1: Nominal Gains for Flows 33
Table 3-2: PI Controller Gains for Flow and Temperature Loops 34
Table 3-3: PI Controller Gains for Level Control Loops 39
Table 3-4: Important Controlled and Manipulated Variables 43
Table 3-5: RGA for Tier Two Controlled and Manipulated Variables 44
Table 3-6: Loop Pairings for the Second Tier 44
Table 3-7: Initial PI Controller Gains from ATV Algorithm 47
Table 3-8: Final PI Controller Gains for the Second Tier 48
Table 4-1: Degrees of Freedom Available for Tuning 70
Table 5-1: Typical Rule Table for Linguistic Variables 89
Table 5-2: Fuzzy Rule Base for Crossover 96
Table 5-3: Fuzzy Rule Base for Mutation 97
Table 5-4: Rule Base Using Population Diversity 102
Table 6-1: Optimal Setpoints and Agitation Rate for the Decentralized Controlled System..... 104

Chapter 1

Introduction to the Tennessee Eastman Chemical Process

Over the years, the processes involved in manufacture and production of purified chemicals have become increasingly more complex. As they have increased in complexity, there has also been a desire to reduce the costs while creating better quality chemicals. Systems that were once controlled by operators based on the system performance have been converted to automatic control, where a mechanical or electrical controller adjusts valves and pumps. With the development of automatic control techniques, the chemical engineering control community has had more freedom to develop complex control algorithms and implement new technological advances. One of the main holdups though has been verifying that the new control algorithms work in realistic settings as expected. In an effort to increase research in the field, several industrial companies have taken the initiative of offering their realistic systems as test-bed problems for the academic community. Testing and development of control algorithms around their systems may help in development of future process control strategies. This study addresses one of these benchmark problems, in particular, the Tennessee Eastman Process [6]. Derivation of a control system that stabilizes the process and reduces the operating costs will be explained and verified through dynamic simulations.

1.1 History of Chemical Process Control

Engineers have learned that to develop the control algorithms for large complex systems, it is important to first create and test a model of the system offline. Two reasons to do this in chemical plants are (i) to protect humans and the environment from unsafe conditions and (ii) to shorten the design and analysis of the system, since typical time constants of chemical plants can be very large. In simulations a plant shutdown, such as a level overflow, cannot cause damage, but assists the control designer in determining configurations to avoid. In computer simulations a real-time hour of plant operation may take as few as a couple of milliseconds, providing the engineer more design time. In addition, with a simulated plant, it can be operated in test conditions that would be highly alarming if they occurred in the real setting.

These benefits of offline tuning can only be done if there is a good model of the system. Typically, any chemical system can be represented by a set of mathematical equations that map the inputs and outputs of the system with a set of state variables. The main set of equations for normal operation of chemical processes are the dynamic equations associated with the reactions, unit operations, and flows of the system.

One important test-bed system posed for study is the Shell Process Control Problem [17]. The objective was to control a heavy oil fractionator to operate within defined constraints. Several researchers took the opportunity to apply their theoretical control techniques to the system to prove or disprove that the algorithms work for that type of process. Since the Shell Process Control Problem [17] several other realistic industrial problems have been provided to the academic community for development and testing of control algorithms.

One of the most interesting and challenging problems posed is a system provided by the Eastman Chemical Company. For nearly a decade now the plant model of the Tennessee Eastman process proposed by Downs and Vogel [6] has challenged engineers to derive efficient controllers. As a benchmark problem it provides an opportunity for designers to test their controllers against those of others on a comparable basis. In this chapter the model assumptions as laid out in [6] are reviewed.

1.2 Features of the Tennessee Eastman Process

The Tennessee Eastman (TE) Process, as presented by Downs and Vogel in [6], is based on an actual system, with slight changes made to protect the identity of the reactants and products, that was released to the process control academic community. The system is a well-posed problem for analysis and control design of a nonlinear, open-loop unstable chemical process. The plant consists of five major operations as shown in Figure 1.1: a two phase reactor, a product condenser, a vapor/liquid separator, a recycle compressor and a product stripper. The nonlinear dynamics of the plant are mainly due to the chemical reactions within the reactor.

Chemicals A, D, and E enter the system and are sent directly to the reactor. Chemical C enters the stripper and reaches the reactor through a recycle stream. The reactor converts the gases to liquid vapors that are passed to a condenser. Once in the condenser, the vapors are converted to liquid and passed on to the separator. The chemicals still in a vapor phase within the separator are sent back to the reactor or exit in a purge stream. The liquefied chemicals from the separator are purified by the stripper before they are sent downstream to other processes.

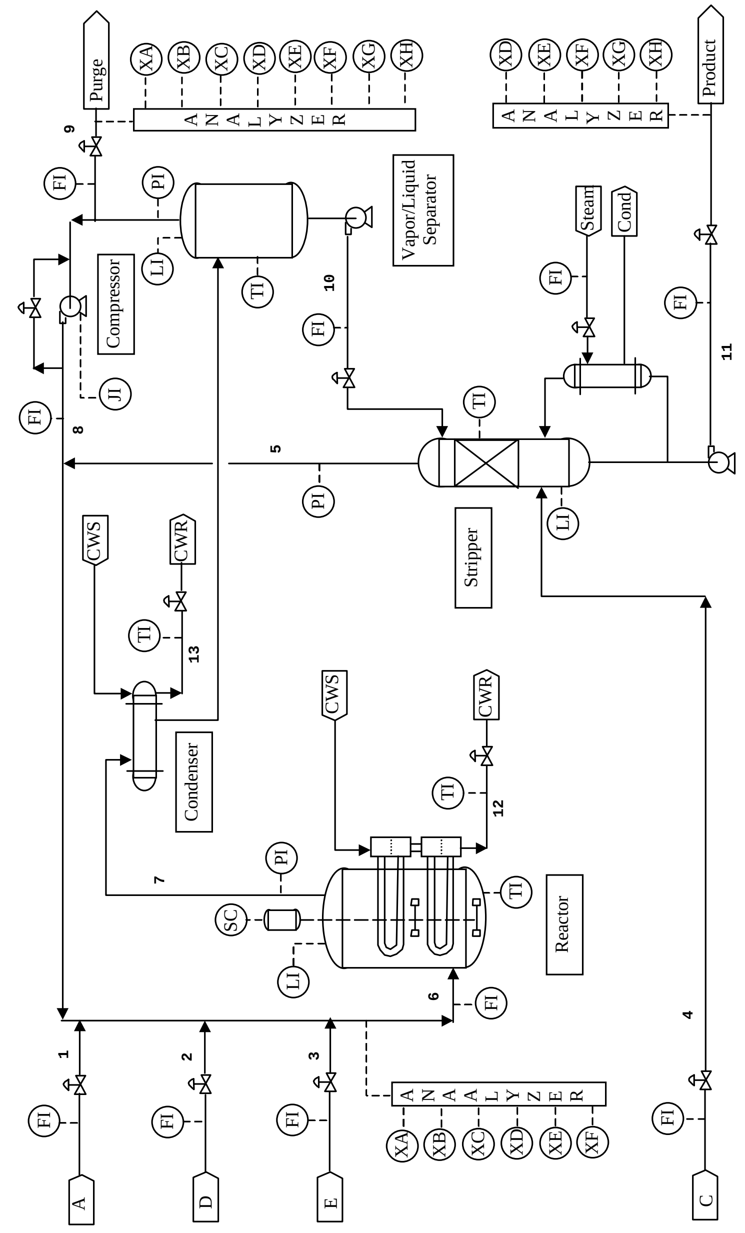


Figure 1.1: Plant diagram of the Tennessee Eastman chemical process

1.2.1 Chemical Reactions in the System

The reactions of the system occur in the stirred tank reactor and are purified in the rest of the process. The main chemical reactions in the system are shown in Equations 1.1 to 1.4.

$$A(g) + C(g) + D(g) \rightarrow G(liq) \tag{1.1}$$

$$A(g) + C(g) + E(g) \rightarrow H(liq) \tag{1.2}$$

$$A(g) + E(g) \rightarrow F(liq) \tag{1.3}$$

$$3D(g) \rightarrow 2F(liq) \tag{1.4}$$

The primary objective of the control strategy is to produce the correct mixture of the G and H products at a specified rate. Equations 1.3 and 1.4 show the byproduct chemical reactions that affect the product quality. In addition to the above chemical reactions, there is an inert gas B that enters through the A and C Feed (stream 4) which must exit through the purge stream.

The gas-to-vapor chemical reactions are all exothermic and irreversible. Reaction rates for this system closely follow a function of temperature through an Arrhenius expression:

$$\text{k} = \alpha \text{exp(-E / RT)} \tag{1.5}$$

where: k = reaction rate

$\propto$ = constant

E = activation energy

T = absolute temperature

R = 1.99 cal/g-mol/K

1.2.2 Modes of Plant Operation

The plant production rates and ratios of the G to H liquids define the six modes or configurations of the plant. The operating modes are listed below in Table 1-1. The first mode is used most frequently and as a result several researchers have focussed their attention on this particular mode. The six modes have very different dynamics, therefore developing a controller to handle all modes becomes a very difficult task.

Table 1-1: Plant Operating Modes

Mode	G/H Mass Ratio	Production Rate
1	50/50	7038 kg/hr G and 7038 kg/hr H (base-case)
2	10/90	1408 kg/hr G and 12669 kg/hr H
3	90/10	10000 kg/hr G and 1111 kg/hr H
4	50/50	Maximum Production Rate
5	10/90	Maximum Production Rate
6	90/10	Maximum Production Rate

1.2.3 Control Objectives of the TE Process

The input gases for streams 1 (A feed), 2 (D feed) and 4 (A and C feed) have been created from other production facilities that do not allow these inputs to vary significantly. Reducing the variability of these streams is one of several recommended control objectives. Downs and Vogel [6] proposed the following control objectives for this system:

1. Maintain process variables at desired values
2. Keep process operating conditions within equipment constraints
3. Minimize variability of product rate and product quality during disturbances
4. Minimize movement of the gas valves which affect other processes
5. Recover quickly and smoothly from disturbances, production rate changes, or product mix changes

A suitable controller must be able to achieve at least these control objectives. Downs and Vogel [6] have provided several tests that can be conducted on the plant to verify that the control designs are robust to realistic changes in the system. The tests they proposed, based on the above control objectives, are:

1. A production rate step change of -15%
2. A product mix step change from the 50/50 G/H ratio to a 40/60 G/H ratio
3. A step change in the reactor pressure from 2705 kPa to 2645 kPa
4. A step change in the B component of the purge from 13.82% to 15.82%
5. A load disturbance on the A/C feed ratio of stream 4 (IDV(1))
6. A load disturbance on the reactor cooling water temperature (IDV(4))
7. A random disturbance on the A, B, and C compositions (IDV(8))
8. A simultaneous load disturbance on the condenser cooling water temperature and a sticking condenser cooling water valve (IDV(12) and IDV(15))

The controlled system must be able to handle at least these eight scenarios. Other possible disturbance scenarios can be seen in Table 1-2, but the ones that occur most frequently are shown above.

Table 1-2: Disturbance Scenarios of the Tennessee Eastman Process

Disturbance Name	Variable Under Disturbance	Type of Disturbance
IDV(1)	A/C Feed Ratio, B Composition	Step
IDV(2)	B Composition, A/C Ratio Constant	Step
IDV(3)	D Feed Temperature	Step
IDV(4)	Reactor Cooling Water Inlet Temperature	Step
IDV(5)	Condenser Cooling Water Inlet Temperature	Step
IDV(6)	Loss of A Feed	Step
IDV(7)	Loss of C Header Pressure - Reduced Availability	Step
IDV(8)	A, B, C Feed Composition	Random Variation
IDV(9)	D Feed Temperature	Random Variation
IDV(10)	C Feed Temperature	Random Variation
IDV(11)	Reactor Cooling Water Inlet Temperature	Random Variation
IDV(12)	Condenser Cooling Water Inlet Temperature	Random Variation
IDV(13)	Reaction Kinetics	Slow Drift
IDV(14)	Reactor Cooling Water Valve	Sticking
IDV(15)	Condenser Cooling Water Valve	Sticking
IDV(16)	Unknown???	Not Specified
IDV(17)	Unknown???	Not Specified
IDV(18)	Unknown???	Not Specified
IDV(19)	Unknown???	Not Specified
IDV(20)	Unknown???	Not Specified

1.2.4 Plant Assumptions

While developing the plant model, several assumptions had to be made in addition to the choice of model equations. The primary assumptions are listed below:

1. All vapors behave as ideal gases
2. The vapor/liquid equilibrium follows Raoult's Law with the vapor pressure calculated using the Antoine equation *Pressure = exp(A-B/(Temp +C))*
3. All vessels are well-mixed
4. The reactor tank is agitated
5. The relationship between the flow through the compressor and difference in inlet to outlet pressure follows a typical centrifugal compressor curve

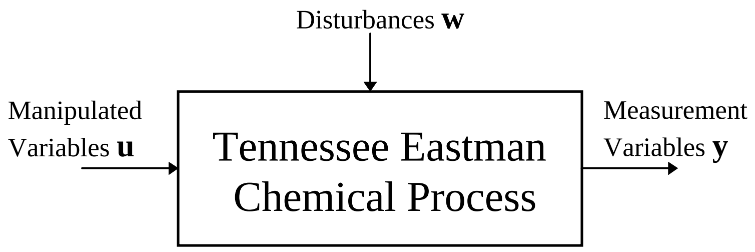
Several other assumptions were made, but these are the primary ones that affect plant simulation and control design. The assumptions were formed based on experience and input/output data from the physical system.

1.3 Features of the FORTRAN Code

From the physical system setup, a FORTRAN program to simulate operation of the process was developed by Downs and Vogel [6]. Although the actual system will not behave exactly as the modeled system, the two should be "close enough" that a good robust design created from the plant model will work on the actual system. The authors note that the model of the system is based on first-order principles [6], therefore the actual plant and system model should have some discrepancies. Several of the equations used in the current chemical/chemical engineering fields to approximate the input/output characteristics of a system were incorporated in the code. One extensively used equation is the Antoine equation which relates the vapor pressure to temperature. The model equations are time-invariant, although in the actual plant, time could affect the flows and device sensitivity.

1.3.1 Concealing Proprietary Plant Information

Downs and Vogel [6] wanted to conceal the plant's actual function for proprietary reasons, therefore slightly changed some of the system properties and purposely made the FORTRAN code obscure. Since the equations involved are unknown, the system can be considered a "black box" where only certain pieces of information are known about the system. Although the plant functions as a "black box", enough information is given to understand the system structure and develop control algorithms. A diagram of the system inputs and outputs is shown below.



Disturbances **w**
Manipulated Variables **u**
Tennessee Eastman Chemical Process
Measurement Variables **y**

Figure 1.2: Black box model of the TE process

1.3.2 Inputs and Outputs of the System

The plant model has 12 manipulated variables (**u**), 50 states (**x**), and 41 measurements (**y**) from different locations in the plant. The 20 disturbance scenarios (**w**) can be turned on and off to test the robustness of the system. Of the measured values, there are 22 that are continuously monitored. The other 19 measured values are retrieved from gas chromatographs (analyzers) that give information only at certain intervals. The three analyzers give information about chemical components in the reactor feed, the purge gas, and the product with sampling delays of 0.1 hour, 0.1 hour and 0.25 hour, respectively.

The 50 states of the model closely correspond to the actual plant operation, representing the significant system dynamics. Fast dynamics, such as transmitter lags, are modeled as straight gains because the effects of these transients are minimal. By neglecting the very fast dynamics, the simulation time is drastically reduced. The 12 manipulated variables in the TE process allow the designer 12 degrees of freedom with 9 flow valves, 2 temperature control valves, and control of the reactor agitation rate. Typically, manipulated variables are provided for closed-loop feedback control of the streams or temperatures of the system; the setpoints of these devices become the available inputs as shown in Figure 1.3.

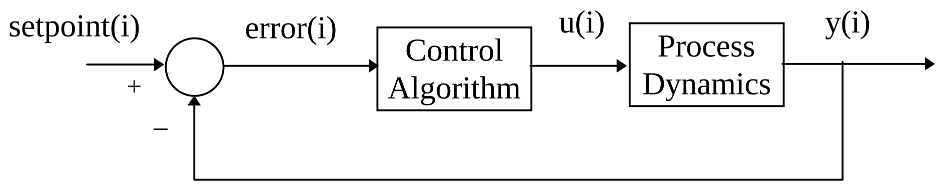


Figure 1.3: Typical decentralized control loop

1.3.3 Measurement Noise and Plant Shutdown

Downs and Vogel [6] have also incorporated two additional features that make the model of the system more realistic. The first feature is that each of the measurement signals contain random gaussian noise typical for its measuring device. The other important feature is a built in controller that shuts down the plant when one of the limits of Table 1-3 are violated.

Table 1-3: Process Operating Constraints

	Normal Operating Limits		**Shut Down Limits**	
Process Variable	**Low Limit**	**High Limit**	**Low Limit**	**High Limit**
Reactor Pressure	none	2895 kPa	none	3000 kPa
Reactor Level	50% (11.8 m^3)	100% (21.3 m^3)	2.0 m^3	24.0 m^3
Reactor Temperature	none	150 °C	none	175 °C
Product Separator Level	30%(3.3 m^3)	100%(9.0 m^3)	1.0 m^3	12.0 m^3
Stripper Base Level	30%(3.5 m^3)	100%(6.6 m^3)	1.0 m^3	8.0 m^3

Plant shutdown is implemented in computer code by making all of the state derivatives go to zero. The model is not intended to be representative of what occurs for plant startup and shutdown conditions.

1.3.4 Plant Operating Costs

A realistic monetary cost function for evaluating production costs per hour of plant operation is also provided. One of the main goals of systems that run at steady-state for significant amounts of time is to reduce the operating costs. For this system, the cost equation, broken up into individual components, is shown below. If the system is operated at a constant level for a long period of time, the operating costs become a very important factor.

$$\begin{aligned} \text{cost / hr} &= (\text{purge costs})(\text{purge rate}) + (\text{product stream costs})(\text{product rate}) \\ &\quad + (\text{compressor costs})(\text{compressor work}) + (\text{steam costs})(\text{steam rate}) \end{aligned} \tag{1.6a}$$

$$\begin{aligned} \text{cost / hr} &= (0.44791\text{y}_{10})(2.206y_{29} + 6.177y_{31} + 22.06y_{32} \\ &\quad + 14.56y_{33} + 17.89y_{34} + 30.44y_{35} + 22.94y_{36}) \\ &\quad + (4.541x_{46})(0.2206y_{37} + 0.1456y_{38} + 0.1789y_{39}) \\ &\quad + (0.0536y_{20}) + (0.0318y_{19}) \end{aligned} \tag{1.6b}$$

The system in the base-case mode has an operating cost of $170.61/hour. One of the major components that can be manipulated for reduced-cost operation is the contents and quantity of the purge stream. The equation above is a slight variation from the one provided by Downs and Vogel [6]. In Equation 1.6b, state x_{46} is the true product flow in kmol per hour. In the original equation, the product flow y_{17}, in units of kgmol per cubic meter, is used for the product stream costs. At the base case, the two equations provide equivalent results, but when the concentrations of the chemicals in the product change, the measurement value y_{17} is slightly inaccurate due to a small error in the code. Although the values of the states should not be used in control design, state x_{46} is used here because of the modeling error.

1.4 Simulating the System

Even with the plant model being available, there are a few things the designer must provide to simulate the system. A discrete integration routine must be available to integrate the states forward in time. The designer must also provide the controller for the system which can either be continuous, discrete, or a combination of the two. The designer selects the operating mode and which disturbances are active.

1.4.1 State Integration Routine

A linearization and analysis of the base-case system reveals that the largest negative eignenvalue is 1968 hr^{-1}, giving a time constant of approximately 1.8 seconds [6]. To keep the discrete integration routine stable, the states can be updated with a sampling time of one second to meet the Nyquist criteria. Several integration routines are available, such as the Runge-Kutta algorithm and Adams-Moulton solver, but several chemical engineering researchers have argued that for complex chemical systems the Euler integration routine can be as good as more computationally intensive routines [11]. The Euler algorithm uses the current state and its derivative to determine the next state. It approximates the derivative of the state by

$$\frac{dx}{dt} \approx \frac{\Delta x}{\Delta t} = \frac{x(k) - x(k-1)}{\Delta t} \tag{1.7}$$

Rearranging Equation 1.7 to isolate *x(k)* leads to the current state output

$$x(k) = x(k-1) + \Delta t \cdot \frac{dx}{dt} \tag{1.8}$$

For a small *Δt*, the Euler integration routine can be implemented to update all the plant states with respect to time. One of the main reasons this simplistic routine is used is that chemical plants typically have both very fast and very slow dynamics. Accuracy of the states from the fast dynamics is sacrificed with this simplistic routine, however, the slow dynamics will be very accurate. Due to several very small eigenvalues that result in slow dynamics, the TE process should be simulated for at least 24 plant operation hours to verify that the transients have reduced significantly and the system has approximately reached steady-state.

1.4.2 MATLAB MEX Interface

MATLAB 5.0 [14] has the ability to compile FORTRAN code into files called MEX files. These MEX files can be run similar to MATLAB function calls. Given the correct FORTRAN interface, called gateway routines, the MEX files produce outputs equivalent to the outputs generated from the original code. The MEX files are used for testing in the design cycle of this research to initialize the plant, propagate the plant, as well as to control it. The main reasons to use the MEX files are for the built in MATLAB routines and MATLAB's convenient user interface. When the final controller is designed, the plant initialization, simulation, and control will be performed through an executable file generated from FORTRAN code.

The gateway routines for initialization and propagation were adapted from code first developed by Srinivas Palavajjhala of Washington University [24]. Comparing the MEX file plant to the original system from compiled FORTRAN code, it has been shown that the two plant models produce very similar responses. A small insignificant difference has been identified due to quantization from converting data from FORTRAN to MATLAB.

1.5 Research Objectives

The main goal of this research is to determine a procedure that minimizes the operating costs of a chemical process. This procedure applies to mode one of the TE process, although the procedure could be followed for other processes or any of the other modes of the TE process. The main objectives as applied to the TE process are to create a robust, stable, closed-loop controller at the 50/50 G/H ratio, 14076 kg/hr mode, and try to reduce the plant's overall operating costs. The controlled system must meet the control objectives outlined above that avoids large changes in variables and plant shutdown. Verification that the criteria are met will be determined by dynamic simulations of the model and control system.

1.6 Thesis Organization

In the following chapter the different possible control techniques for the TE process are investigated. The majority of the proposed solutions use decentralized control to try to create a stable system, although more complex control strategies have also been explored. Since one of the goals of this research is to reduce the operating costs, the minimum operating cost will be determined through a Lagrangian search technique.

Before the system can be controlled for minimum-cost operation, the system must be stable and meet the control objectives outlined above. The techniques and associated values to create the stabilizing controller are outlined in the third chapter. Once the stabilizing controller is in place, more complex control algorithms can be implemented to reduce steady-state operating costs.

In Chapters 4 and 5, the system is modified to reduce the operating costs. In Chapter 4 a genetic algorithm is implemented to find input values that minimize the cost function. In the fifth chapter a fuzzy controller is developed to increase the performance of the genetic algorithm. It adjusts certain parameters of the genetic algorithm to avoid unstable pairings, reduce the possibility of converging to sub-optimal minimums, and reduce the processing time of the genetic algorithm.

In the final chapter, conclusions from the entire process and possibilities for future work are summarized. The advantages and disadvantages of the optimal cost solution is examined. Modifications of the procedure are investigated for applications to different operating conditions and other processes.

Chapter 2

Control and Optimization of the Tennessee Eastman Process

Since its first publication, researchers have attempted various control techniques on the TE process. Although some of the results use similar procedures, the solutions have been quite different. One of the reasons of the disparity between methods and results is that Downs and Vogel in [6] did not provide a principal goal for the controlled system. Since the control objectives were not specified in relative degrees of importance, researchers have tried to control certain aspects of the problem more thoroughly than other aspects. There are many criteria that the researchers can try to meet, but to meet all of the possible control specifications is a very difficult task.

In this chapter, various proposed decentralized control schemes are presented. The system designs are compared to each other using the criteria recommended by Downs and Vogel [6]. One of the control objectives is to reduce the operation costs of the system, therefore derivation of a minimal-cost steady-state solution will be examined.

2.1 Review of Decentralized Control Structures

The majority of the control solutions of the TE process use decentralized control schemes. There are several reasons for this choice, but the most compelling is that operating personnel are familiar with SISO controllers. Complex schemes, such as model-predictive control, require more educated operators. When more in-depth control algorithms are implemented, the process becomes increasingly difficult to determine potential problems. In addition, the more complex routines generally require a very accurate model of the system. A reliable model is typically not available for chemical processes mainly because there are several nonlinear phenomena which are difficult to identify and model. In this section, several of the proposed decentralized control schemes will be examined. Although all have some similar characteristics, most are drastically different due to the researchers interpretations of the control objectives.

The decentralized control designs are divided into two tiers. The first tier, or inner loop, controls the faster dynamics of the process; the second tier, or outer loop, controls the slower dynamics. A key component in the designs is selection of the process variable that controls the production rate. Determining the production rate variable is one of the first steps in design of decentralized process control systems.

2.1.1 Decentralized Controller of Luyben

When constructing the actual process at Eastman Chemical Company, two researchers were consulted in modeling and controlling the process. William Luyben, was one of these researchers, and has published his own recommended control scheme using a decentralized control structure. One of the main goals in the control design of [12] was to create a simplistic control scheme that is easy for operating personnel to understand and debug. Other goals include making the system respond quickly to changes in operating parameters, and minimizing the amount of analyzer data required. His interpretation of the design goals suggests the main objective is to reduce product flow variation for downstream processes. For this reason, the stripper product flow valve is fixed, forcing the product flow to be constant. The remaining control loops revolve around this initial choice.

To make the control algorithm as simple as possible, Luyben in [12] implements his controller using proportional-only type SISO controllers on all of the feedback loops. Tuning the loops did not require a model, but only heuristics and insight into the process dynamics based on the flowsheet and previous experience. The controlled system performed within limits for all the recommended setpoint changes and disturbance scenarios, but did have a problem with one type of disturbance not recommended by Downs and Vogel [6]. Although the control is intended to operate only around the base-case settings, it still requires several overrides to deal with a loss of the A feed (IDV(6)). This control scheme may not be as stable and complex as some other control schemes; however, it does handle all of the recommended setpoint changes and disturbance scenarios.

2.1.2 Decentralized Controller of McAvoy and Ye

One of the first published decentralized control solutions for the TE process was completed by McAvoy and Ye [15]. Their control solution is also conditioned around the base-case mode of the TE process. The control objective they perceived as being most important is reducing the variability of the gas streams and the product. Since one of the control objectives is to minimize the variability from upstream and downstream processes, the authors tried to design the control system to reduce fluctuations in the gas feeds and product stream. The solution uses proportional-integral controllers to provide decentralized controllers for both the inner and outer loop controllers. The production rate of McAvoy and Ye [15] is controlled by the large gas stream 4 (A and C feed).

Despite its large flow rate capability, using stream 4 as the manipulated variable for the production rate is undesirable if trying to operate over all modes. The reactor contains an excess amount of the A and C reactants. If the system is changed to one of the modes where production rate must be maximized, either the D or E feed limit the production rate. McAvoy and Ye [15] attempted control around the first mode, therefore the limited availability of the input streams was not a significant concern. In the base-case mode, McAvoy and Ye were able to handle all of the recommended setpoint changes and disturbance scenarios. When the A feed is lost, the production rate must be reduced to avoid plant shutdown conditions.

2.1.3 Decentralized Controller of Banerjee and Arkun

Banerjee and Arkun [3] have also pursued a decentralized control scheme around the base-case mode using a procedure they call *control configuration design (CCD)*. The *CCD* procedure uses a frequency analysis to determine the singular values of the linearized multivariable system. From the singular values, a sensitivity function is established which governs the system bandwidth. The procedure leads to several potential control configurations for the TE process which are compared by simulation of the plant model. The final selection in [3] from the *CCD* procedure is much different than those recommended by others, but exhibits similar responses. The production rate and G/H ratio are controlled by the D and E feed setpoints. Despite a very detailed analysis and procedure, the results were no better than other traditional methods; and the design procedure required a highly accurate model of the system.

2.1.4 Decentralized Controller of Lyman and Georgakis

Similar to Luyben in [12], Lyman and Georgakis [13] attempted to design a decentralized controller for the TE process without using the dynamic model of the system. The control algorithms were designed with the following goals in decreasing order of importance: sustain production and inventory control, support product specification control, stay within equipment and operating constraints, and improve the economic performance. This is consistent with the other control methods, where the primary goal is system stability and robustness. Once stability is ensured and the system operates within its constraints, the plant economics can be addressed. The central point of each of the control structures in [13] is the production rate manipulator. The control schemes work in an outward direction from the production rate manipulator.

Of the four resulting structures in [13], the best performing scheme around the base-case mode uses the condenser cooling water flow to control the production rate. Unlike several of the other decentralized control schemes, no overrides are necessary to handle the setpoint changes and disturbance scenarios. However, the design requires several of the analyzers to be providing reliable data on-line. One major difference from other control schemes is that the reactor pressure, which is a critical inventory control element, does not have its own controller. The pressure is indirectly controlled through the other controllers. Although not explained by the authors, letting the reactor pressure float is a viable choice because (i) several components affect the reactor pressure, therefore a single manipulated variable cannot completely control the pressure, and (ii) the reactor pressure exhibits an inverse response where the pressure decreases before it increases to a steady-state value, and vice versa. The control loops were implemented and the gains chosen based on experience with similar systems to give well-conditioned responses as would be done in a realistic, unmodeled system.

2.1.5 Decentralized Controller of Ricker

The researcher who has completed the most thorough design and analysis of the TE process is Dr. Larry Ricker of the University of Washington. Ricker investigated the use of several control techniques on the TE process, including non-linear model-predictive control [20] and decentralized control [18]. His decentralized control scheme is superior to other researcher's because it operates over all modes and can handle all setpoint changes and disturbance scenarios. The only detriments of the control structure in [18] are (i) the relatively slow response of the system to changes in operating conditions, and (ii) the complexity of the control algorithm. The key difference between the decentralized controller of Ricker and other decentralized designs is that the production rate variable is used in ratio controllers to control all of the flows. This is one of the main reasons why the system is capable of operating over all modes. In the development of the control scheme, data was used from a steady-state analysis that minimized operating costs [19]. Since the controller met the other more important control objectives, Ricker had the luxury of trying to optimize the plant operating costs. The only time plant operating costs are a considerable factor is when the plant is operating in a particular mode for long periods of time. The controller responds to changes in operating conditions at a slower rate than other controllers because the control system was designed assuming the plant to be operating in steady-state for the majority of the time. The controllers mostly use PI gains, developed from rules for similar systems, but additional controllers such as feed-forward controllers and ratio controllers are also required. Despite being a complex control algorithm, the system uses very little analyzer data, minimizes the operating costs, and creates a stable system over all modes.

2.2 Optimal Operating Costs of the Base-Case System

Creating a robust, stable system is of primary importance. Once the system is found to be stable and meets the other control objectives, the designer can attempt to minimize the operating costs and/or maximize the production rate. This study involves reducing the steady-state operating costs per hour, calculated by Equation 1.6, around the base-case mode. The base-case outputs, inputs, and costs are displayed in the corresponding columns of Tables 2-1 to 2-3. Examining the base-case costs of Table 2-3 reveals that the majority of the operating costs are associated with the purge. In achieving a minimum-cost steady-state system, the purge as well as the steam, compressor, and product costs can be reduced without significantly affecting the product quality or flow rate. In a reduced-cost configuration, the system must still avoid plant shutdown.

Table 2-1: Output Values of the TE Process

#	Measurement Variable	Base-Case Settings	Mode 1 Optimal Costs of N.L. Ricker	Mode 1 Optimal Costs with N.L. Ricker Constraints	Mode 1 Optimal Costs
1	A Feed	0.2505	0.2668	0.2795	0.2688
2	D Feed	3664	3657	3649	3646
3	E Feed	4509	4440	4445	4414
4	A and C Feed	9.348	9.236	9.227	9.214
5	Recycle Flow	26.90	32.18	21.28	22.90
6	Reactor Feed Rate	42.34	47.36	36.38	37.89
7	Reactor Pressure	2705	2800	2800	2895
8	Reactor Level	75.00	65.00	65.00	50.00
9	Reactor Temperature	120.4	122.9	125.0	124.1
10	Purge Rate	0.3371	0.2109	0.2066	0.1802
11	Product Separator Temperature	80.11	91.70	75.44	74.78
12	Product Separator Level	50.00	50.00	50.00	45.22
13	Product Separator Pressure	2634	2706	2755	2848
14	Product Separator Underflow	25.16	25.28	24.07	23.64
15	Stripper Level	50.00	50.00	50.00	46.95
16	Stripper Pressure	3102	3326	3050	3159
17	Stripper Underflow	22.95	22.89	22.60	22.57
18	Stripper Temperature	65.73	66.54	54.12	54.21
19	Stripper Steam Flow	230.3	4.738	0.000	0.000
20	Compressor Power	341.4	278.9	332.6	342.0
21	Reactor Cool Water Outlet Temp	94.60	102.4	99.20	98.36
22	Condenser Cool Water Outlet Temp	77.30	91.99	51.25	60.87
23	Reactor Feed – Component A	32.19	32.21	35.91	35.68
24	Reactor Feed – Component B	8.893	14.93	13.15	15.53
25	Reactor Feed – Component C	26.38	18.75	21.71	21.42
26	Reactor Feed – Component D	6.882	6.032	7.395	7.083
27	Reactor Feed – Component E	18.78	16.71	14.27	12.96
28	Reactor Feed – Component F	1.657	4.043	3.443	3.237
29	Purge Gas – Component A	32.96	32.73	39.15	38.44
30	Purge Gas – Component B	13.82	21.83	22.27	25.50
31	Purge Gas – Component C	23.98	13.11	15.22	15.12
32	Purge Gas – Component D	1.257	0.9027	0.6168	0.5537
33	Purge Gas – Component E	18.58	16.19	11.81	10.08
34	Purge Gas – Component F	2.263	5.388	4.877	4.470
35	Purge Gas – Component G	4.844	6.618	4.115	3.967
36	Purge Gas – Component H	2.299	3.226	1.950	1.876
37	Product – Component D	0.0179	0.0109	0.0127	0.0116
38	Product – Component E	0.8357	0.5822	0.7936	0.6890
39	Product – Component F	0.0986	0.1876	0.3177	0.2961
40	Product – Component G	53.72	53.83	53.64	53.71
41	Product – Component H	43.83	43.91	43.76	43.82

Table 2-2: Input Values of the TE Process

#	Input Variable	Base-Case Settings	Mode 1 Optimal Costs of N.L. Ricker	Mode 1 Optimal Costs with N.L. Ricker Constraints	Mode 1 Optimal Costs
1	D Feed Flow (stream 2)	63.05	62.93	62.79	62.75
2	E Feed Flow (stream 3)	53.98	53.15	53.21	52.84
3	A Feed Flow (stream 1)	24.64	26.25	27.50	26.44
4	A and C Feed Flow (stream 4)	61.30	60.57	60.51	60.43
5	Compressor Recycle Valve	22.21	1.00	60.70	57.18
6	Purge Valve (stream 9)	40.06	25.77	21.23	17.91
7	Separator Liquid Flow (stream 10)	38.10	37.27	36.81	36.31
8	Stripper Liquid Flow (stream 11)	46.53	46.44	46.60	46.54
9	Stripper Steam Valve	47.45	1.00	0.00	0.00
10	Reactor Cooling Water Flow	41.11	35.99	37.81	38.21
11	Condenser Cooling Water Flow	18.11	12.43	60.33	32.69
12	Agitator Speed	50.00	100.00	48.76	48.62

Table 2-3: Costs per Hour of Operation of the TE Process

	Base-Case Settings	Mode 1 Optimal Costs of N.L. Ricker	Mode 1 Optimal Costs with N.L. Ricker Constraints	Mode 1 Optimal Costs
Compressor Costs	$18.30	$14.95	$17.83	$18.33
Steam Costs	$7.32	$0.15	$0.00	$0.00
Purge Costs	$114.72	$73.75	$57.65	$46.87
Product Costs	$30.27	$25.46	$37.08	$32.94
Total Operating Costs per Hour	**$170.62**	**$114.31**	**$112.55**	**$98.14**

2.2.1 Formulation of the Optimal Cost Solution

The FORTRAN code of Downs and Vogel [6] presents the system in the form

$$\frac{d\boldsymbol{x}}{dt} = \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}, \boldsymbol{u}, t) \qquad (2.1)$$

$$\boldsymbol{y}(t) = h(\boldsymbol{x}, t) \qquad (2.2)$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ is the input vector, and $\boldsymbol{y}$ is the output vector. The functions f and h are nonlinear functions that relate the states, inputs, and time to the state derivatives and measurement values. A state x_i is considered to be at a steady-state value when its first derivative is zero. For the TE process, the states of the system must be greater than zero for all time and the 12 inputs, $\boldsymbol{u}$, are constrained between 0 and 100. Downs and Vogel in [6] prepared the code such that at time $t=0$ the output variables do not include measurement noise. Operating at initial time with the derivatives equal to zero, the steady-state values can be determined. The constraints above and the cost function per hour can be formatted as an objective function in a nonlinear constraint optimization problem. The optimization problem has the form

$$\min_{\boldsymbol{x},\boldsymbol{u}} \ \text{cost}(\boldsymbol{x}, \boldsymbol{u}, t) \qquad (2.3)$$

subject to

$$\frac{d\boldsymbol{x}}{dt} = 0 \qquad (2.4)$$

$$0 \le u_i \le 100 \quad \forall\ i = 1,2,\ldots,12 \qquad (2.5)$$

$$0 \le x_j \quad \forall\ j = 1,2,\ldots,50 \qquad (2.6)$$

Additional constraints are required to guarantee that the system is operating in the correct mode and within normal operating limits. The extra constraints are:

$$p_G = 2.8153 x_{46} y_{40} \tag{2.7}$$

$$p_H = 3.4510 x_{46} y_{41} \tag{2.8}$$

$$y_7 \leq 2895 \tag{2.9}$$

$$50 \leq y_8 \leq 100 \tag{2.10}$$

$$30 \leq y_{12} \leq 100 \tag{2.11}$$

$$30 \leq y_{15} \leq 100 \tag{2.12}$$

where p_G is the mass fraction of product G and p_H is the mass fraction of product H. In the first mode p_G and p_H equal 7038 kg/hr. Equations 2.9 to 2.12 keep the reactor pressure and tank levels within normal operating limits.

As previously mentioned, the state vector contains 50 states, but at steady-state the last 12 are equivalent to the system inputs $\boldsymbol{u}$. Using this information, the optimization problem of Equation 2.3 at $t=0$ reduces to finding the solution of

$$\min_{\boldsymbol{x}} \ \mathrm{cost}(\boldsymbol{x},0) \tag{2.13}$$

subject to the constraints of Equations 2.4 through 2.12. The problem has now been posed as an optimization problem with 50 input values corresponding with the states. There are several computer routines that use dynamic programming to solve these types of problems, but few of them can solve a problem with this many inputs.

2.2.1 Optimal Cost from Analysis by Ricker

Ricker has identified through a Lagrangian based minimization routine, called MINOS [16], the optimal steady-state values of the states to achieve both minimum-plant cost and maximum production over all modes [19]. Both routines use the entire plant state vector as an input and minimize/maximize the appropriate objective function until steady-state values are found that meet the system constraints. Besides the constraints of Equations 2.4 to 2.8, the inequality constraints of Equations 2.9 through 2.12 are substituted with Equations 2.14 to 2.17 to avoid operation close to a stability limit.

$$y_7 \leq 2800 \tag{2.14}$$

$$65 \leq y_8 \leq 100 \tag{2.15}$$

$$y_{12} = 50 \tag{2.16}$$

$$y_{15} = 50 \tag{2.17}$$

Ricker has shown that for the first mode with the above constraints, a local minimum cost per hour exists at $114.31 [19]. The corresponding values are shown in the fourth column of Tables 2-1 and 2-2 and the third column of Table 2-3. These results are for the model of the plant, which is not necessarily the same as the actual system. Although he is confident in the results, this cost may not be a global minimum. The states of the system would probably be unavailable in the realistic system without a complex observer. The optimization routine allows the system to be tested outside of the constraints, therefore this analysis could only be completed using a simulation of the system.

This data is important to this research which is attempting to find the minimum operating costs for the controlled system. It is assumed that the model of the system is fairly accurate due to the amount of testing and analysis done by the authors [6]. Even if the actual system is significantly different, the gains associated with decentralized controllers can be changed drastically and still produce a stable response. It was attempted to replicate the work of Ricker [19] at the base-case mode. The MINOS program [16] worked directly with the FORTRAN code of the plant model. A similar routine that can solve large-scale optimization problems, SOLNP [22], was used in MATLAB, but instead of using the FORTRAN code, the MEX file version of the model was used. SOLNP has been found to give better results than the MINOS program in several optimization problems [22]. The SOLNP program solves a linearly constrained optimization problem by transforming the original nonlinear function into an augmented Lagrangian objective function [22]. Sequential quadratic programming techniques are used to solve the cost objective function, which include a gradient vector and Hessian matrix estimation per iteration. For a complete reference on the performance of the SOLNP program, refer to [22].

2.2.2 Comparison of the SOLNP and MINOS Programs on the TE Process

As mentioned in Chapter 1, the MATLAB MEX file output is slightly different from the FORTRAN equivalent due to quantization of values. The equality constraint of Equation 2.4 requires the derivatives to equal zero. Numerically, the MINOS program assumes the derivatives to be zero if the maximum derivative has magnitude less that 10^{-3}. The SOLNP file was formatted such that the derivatives were assumed zero when the maximum derivative has magnitude less than 10^{-6}. Once the minimum costs were located using the SOLNP program with the MEX files, the steady-state values were tested using the FORTRAN code. Typically, all the derivatives using the FORTRAN code were less than $5*10^{-3}$, therefore the steady-state values were very close. Using the SOLNP program with proper convergence parameters, the minimum found by Ricker was located. Several iterations of the SOLNP program led to a smaller cost being found of $112.55. This tends to show that the SOLNP program works more efficiently than the MINOS program for large-scale optimization problems. The values found for optimization of the costs are displayed in Tables 2-1 through 2-3 adjacent to the Ricker settings. The MATLAB m-files “opt2_n.m” and “cost.m” in Appendix A were used to obtain these results. Although the SOLNP was performed several times with different initial conditions, global minimization still cannot be ensured.

2.2.3 Minimum Operating Costs with Fewer Constraints

After verifying that the results of Ricker [19] could be replicated, the constraints of Equations 2.14 to 2.17 were changed to those of Equations 2.9 to 2.12. The reactor pressure was allowed to increase to 2895 kPa, and the tank levels were allowed to operate over their entire normal operating range.

The SOLNP program was run several times using various initial conditions on this set of equations with p_G and p_H equal to 7038 kg/hr. The minimum-cost solution found was $98.14. The states and measurement values are shown in the last columns of Tables 2-1, 2-2 and 2-3. With these values placed in the FORTRAN code, the maximum derivative at $t=0$ is $2.5*10^{-3}$, therefore a steady-state has approximately been reached. A simulation using these settings without measurement noise verified the system is in steady-state. After 100 hours of plant simulation, the output variables did not change even 0.1% from their preset initial values.

2.3 Conclusions

This chapter emphasized that even restricting the control system of the TE process to use a decentralized control method, there are still various parameters that can be adjusted. There is some debate in the literature over which control objectives are most important. For the TE process, the control objectives of [6] must be met, but there are other issues to examine including product variation, speed of response, and production costs. Trying to accomplish all the control objectives to some degree is important, but the control engineer usually must use his knowledge and experience to determine the most critical constraints.

An analysis of steady-state costs was described to prepare for a controller where the production costs are reduced. The above findings will be used as the best-case scenario for plant operation around the base-case mode. If the inputs can drive the states to these values without causing hazardous conditions, then creating an optimized reduced-cost system is possible.

Chapter 3

The Stabilizing Decentralized Controller

In chemical process design one important rule is to keep the control process simple. Complex control structures are almost always disregarded as unrealistic systems [11]. Plant operators like to have systems that are familiar, so complex control systems may cause operators to switch the control loops to manual mode whether the system is operating incorrectly, or not. Although other authors have explored more complex control algorithms such as linear model-predictive control and other centralized control methods such as that in [20], this study focuses on a decentralized scheme that uses single-input single-output (SISO) control structures based on pairings of manipulated variables and measurement outputs. An advantage of the SISO structure is that each controller can be implemented separately in the physical system. The decentralized controllers in this study use common proportional/integral/derivative (PID) controllers with additional overrides to compensate for disturbances.

3.1 The PID Controller in Chemical Process Control

In many systems, control is achieved by implementing a relatively simple controller called a PID controller. A PID controller manipulates the plant input variable, u, based on the difference between a setpoint and the loop's controlled variable. A typical closed-loop SISO PID controlled system is shown in Figure 3.1.

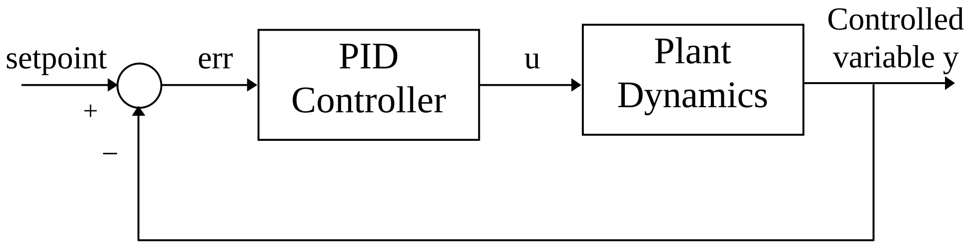


Figure 3.1: SISO closed loop system with PID controller

3.1.1 Derivation of the PID Equation

In the diagram above the continuous-time (CT) PID controller has the form

$$u(t) = K_c\left[err(t) + \frac{1}{T_I}\int_0^t err(\tau)d\tau + T_D\frac{derr(t)}{dt}\right] \tag{3.1}$$

Conversion to the Laplace domain for frequency response analysis leads to the transfer function

$$\frac{U(s)}{ERR(s)} = K_c\left[1 + \frac{1}{T_I s} + T_D s\right] \tag{3.2}$$

The PID controller can be implemented as a discrete-time (DT) controller with a fixed sampling time T_s. Given a small enough sampling time, the DT PID controller gives a response very similar to that of the CT PID controller. A simple conversion from continuous to discrete-time uses the circumscribed polygon method to approximate the integral term of the error at time *k*.

$$\int_0^t err(\tau)d\tau \approx T_s\sum_{j=1}^{k} err(j) + err(0) \tag{3.3}$$

The derivative term is approximated from the change in the signal over one sample period.

$$\frac{derr(t)}{dt} \approx \frac{err(k) - err(k-1)}{T_s} \tag{3.4}$$

Replacing the integral and derivative terms in Equation 3.1 with the discrete-time approximations of 3.3 and 3.4 yields the discrete-time PID controller in "position" form.

$$u(k) = K_c\left[err(k) + \frac{T_s}{T_I}\sum_{j=1}^{k} err(j) + err(0) + \frac{T_D}{T_s}[err(k) - err(k-1)]\right] \tag{3.5}$$

Subtracting the current manipulated variable *u(k)* from the previous sampled manipulated variable

$$u(k-1) = K_c\left[err(k-1) + \frac{T_s}{T_I}\sum_{j=1}^{k-1} err(j) + err(0) + \frac{T_D}{T_s}[err(k-1) - err(k-2)]\right] \tag{3.6}$$

gives the change in manipulated variable over one sample period. This new form of the equation is called the "velocity" form where the updated variable is not the manipulated variable, but the change in the manipulated variable. The “velocity” form of the PID equation is

$$\Delta u(k) = K_c\left[\Delta err(k) + \frac{T_s}{T_I}err(k) + \frac{T_D}{T_s}\left[err(k) - 2err(k-1) + err(k-2)\right]\right] \tag{3.7}$$

where *Δu(k)* and *Δerr(k)* are the changes in a single sample interval. Moving the previous sample of the manipulated variable to the right side of Equation 3.7, the “velocity” form of the PID equation can be rewritten to output the new manipulated variable, *u(k)*.

$$u(k) = u(k-1) + K_c\left[err(k) - err(k-1) + \frac{T_s}{T_I}err(k) + \frac{T_D}{T_s}\left(err(k) - 2err(k-1) + err(k-2)\right)\right] \tag{3.8}$$

Equations 3.5 and 3.8 are equivalent for determining the new manipulated variable, but the “velocity” form is computationally less intensive to implement in computer code. Equation 3.8 will be used whenever a DT PID controller is required.

3.1.2 Use of the PID Equation in Chemical Processes

In the PID equation the proportional, integral, and derivative terms can be independently adjusted to change the dynamic response of the system. The proportional gain creates an output signal that is a constant gain of the error and is therefore usually necessary for a well-conditioned system response. The integral term provides asymptotic tracking of a reference input, or setpoint in chemical processes, by making the error signal approach zero. The derivative term mainly adjusts the speed of response of the loop. In chemical processes the derivative term is often times disregarded as part of the control scheme because derivative control enhances the noise characteristics in systems. The derivative term can lead to a large, quick change in the manipulated variable which could saturate a valve. The speed of response is less important in chemical systems than other systems because systems are usually run in steady-state for long periods of time. For these reasons, the decentralized controllers will not have derivative terms; the controllers will be PI controllers.

3.1.3 Calculating Controller PI Gains

There are several general techniques and rules to calculate controller gains for a particular system. Two of the more common techniques used for design are the Ziegler-Nichols rules and Internal Model Control rules. These rules give first guess process controller gains to stabilize the system. Once the system is stable, the gains can be adjusted based on a dynamic simulation or plant operation. Although there are simple rules as mentioned above, controller gains today are still largely based on rules of thumb and experience [11].

3.2 Development of the PI Decentralized Controller

In the previous chapter it was shown that the entire system can be viewed as a black box. This will be a key component later in this study because although a simulation program of the system can be written, the performance of the system may not be easily identifiable, especially when nonlinearities are involved. Given the information in the TE plant description, general rules for controller stabilization can be used.

The first step in the controller design is to determine possible loop pairings and design controllers around the 12 inputs of the system. The pairings lead to controllers that control the flows, temperatures, levels, and pressures (inventory control). This procedure can be broken into two stages or tiers. The first tier, or inner loop, will control the sections of the plant that have the faster dynamics of the system. The second tier, or outer loop, will control the slower dynamics and may use the inner loop setpoints in cascade as manipulated variables. A diagram of a cascaded system for the two tiered design is shown below.

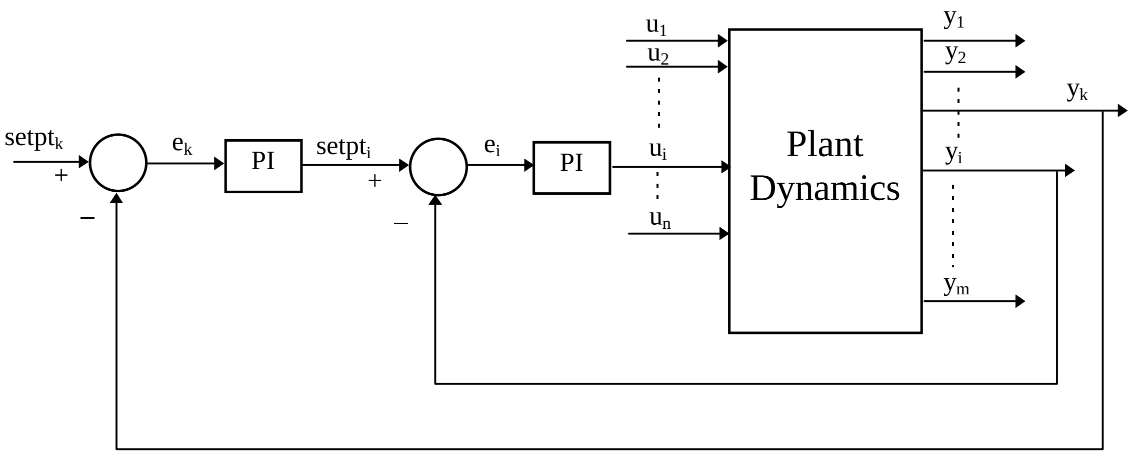


Figure 3.2: Cascaded decentralized system

The second major step in the controller design is to develop control algorithms that stabilize the system to disturbances, measurement noise, and changes in operating conditions. The control loops will be part of the second tier as the system dynamics for these controllers should respond slower than the first tier controllers.

3.2.1 Design of the Inner Loop Cascade Controllers Around the Base-Case

The eight flow inputs, **u**, were each independently given step changes of ±10% about their base-case values and the output measurements recorded. Although the system is nonlinear and the outputs are highly coupled, the impact of a change in an input normally affects only a few measurement values. The resulting flow pairings shown in Table 3-1 make logical sense because the main reason the measurement devices are placed in these locations are for control of the neighboring valves. The flow loops could have been closed without this extensive investigation, although the gains and speed of response would have been unknown. In the TE process, the measurement devices are placed before the valves to reduce the effects of flow disturbances. A disturbance in one of the flows will be recognized by the controller before reaching the valve.

For the entire design section, the measurement noise from the simulator was removed while tuning the control loops. Table 3-1 shows the magnitude of the change in response (nominal gains) from input to the significantly affected output signals. The first gain value of each row is for a positive step change; the second gain value is for a negative step change. Since a step input on flows have immediate step changes on the measurement devices, the gain is basically a conversion from valve percentage open to the appropriate measurement units. The equation that calculates the nominal gain for each of the flow devices is

$$K = \frac{\Delta y}{\Delta u} \tag{3.9}$$

The error signal from a feedback control system has the same units as the controlled variable. By incorporating a proportional controller with a gain of *1/K,* the error signal units are converted to a percentage before being manipulated by the controller. The process control field often times represents gains in terms of proportional band to include conversion factors and limits on the device ranges. When the input and output have the same units, the controller gain K_c is

$$K_c = \frac{100}{\% PB} \tag{3.10}$$

where %*PB* is the proportional band value. In this case, the *100* in the numerator of Equation 3.10 is replaced by *100*1/K* to incorporate the conversion factor mentioned above.

Table 3-1: Nominal Gains for Flows

Controlled Variable *y*	Manipulated Variable *u*	Gain *Δy/Δu*
A Feed (XMEAS1)	A Feed Flow (XMV3)	0.01017 0.01017
D Feed (XMEAS2)	D Feed Flow (XMV2)	58.1 58.1
E Feed (XMEAS3)	E Feed Flow (XMV1)	83.5 83.5
C Feed (XMEAS4)	C Feed Flow (XMV4)	0.1525 0.1525
Purge Flow/Rate (XMEAS10)	Purge Valve (XMV6)	0.00840 0.00840
Separator Underflow (XMEAS14)	Separator Pot Liquid Flow (XMV7)	0.664 0.663
Stripper Underflow (XMEAS17)	Stripper Product Flow (XMV8)	0.493 0.493
Stripper Steam Flow (XMEAS19)	Stripper Steam Valve (XMV9)	4.03 4.15

From the nominal gains in Table 3-1 suitable flow controller gains can be calculated as shown in Table 3-2. The nominal gains were calculated using a proportional band of 150% which is an acceptable proportional band for flow controllers. A rule of thumb based on experience has shown that flow controllers should have a wide proportional band from 100 to 200% to reduce the effects of noise disturbances in flow signals. In flow controllers, the integral time is usually small to get fast tracking of a loop setpoint. Similar process control experiments have shown that a suitable reset time for flow controllers is approximately 0.1 minutes [11]. Each of the flow controllers have T_I set to 0.1 minutes. The flow controllers can be continuous or discrete, but for the discrete case, since flows comprise the fastest dynamics of the system, they must have small sampling times to avoid violation of the Nyquist criteria.

Table 3-2: PI Controller Gains for Flow and Temperature Loops

Controlled	**Manipulated**	**K_c**	**T_I (min.)**
A Feed (XMEAS1)	A Feed Flow (XMV3)	65	0.1
D Feed (XMEAS2)	D Feed Flow (XMV2)	0.0115	0.1
E Feed (XMEAS3)	E Feed Flow (XMV1)	0.008	0.1
C Feed (XMEAS4)	C Feed Flow (XMV4)	4.4	0.1
Purge Flow/Rate (XMEAS10)	Purge Valve (XMV6)	79	0.1
Separator Underflow (XMEAS14)	Separator Pot Liquid Flow (XMV7)	1	0.1
Stripper Underflow (XMEAS17)	Stripper Product Flow (XMV8)	1.35	0.1
Stripper Steam Flow (XMEAS19)	Stripper Steam Valve (XMV9)	0.16	0.1
Reactor Cooling Water Temp (XMEAS21)	Reactor Cooling Water Flow (XMV10)	–5	0.5
Condenser Cooling Water Temp (XMEAS22)	Condenser Cooling Water Flow (XMV11)	–5	0.5

A similar procedure can be performed on the temperature inputs of the system. When the two cooling water valves are adjusted, there are several outputs that change drastically, but the responses are considerably slower than the flow loops. Examination of the system diagram shows that the measurements for the reactant cooling water outlet temperature (RCWOT) and the condenser cooling water outlet temperature (CCWOT) are adjacent to the corresponding input valves. The measurements are therefore good candidates for feedback control. The temperature feedback control loops are closed using the cooling water to control the water outlet temperatures. The PI gains, which are also listed in Table 3-2, are determined by rules of thumb and trial and error using the dynamic simulation of the system.

The two remaining system inputs come from the compressor recycle valve and the agitation rate. The compressor valve has much slower dynamics than the flow and temperature loops, therefore its use will be examined in the second tier. It has a significant response on three variables: the recycle flow, the reactor feed rate, and the compressor power. The agitation rate affects the speed of the reactions within the reactor and has little effect on the operating cost. Its use in controller design will also be examined in the second tier since the system response to a change in agitation is much slower than a change in the flows or temperatures.

The diagram of Figure 3.1 can be redrawn to reflect the practice commonly found on process and instrumentation drawings. The controller is grouped into a controller block with the setpoint and controlled value as inputs. The output value of the flow controller is the manipulated variable to control the valve. An example is given below for a flow control system. The FI (flow indicator) sends the measurement value to the flow controller for comparison to the setpoint. The flow controller manipulates the valve position based on the difference between the flow setpoint and its corresponding flow value.

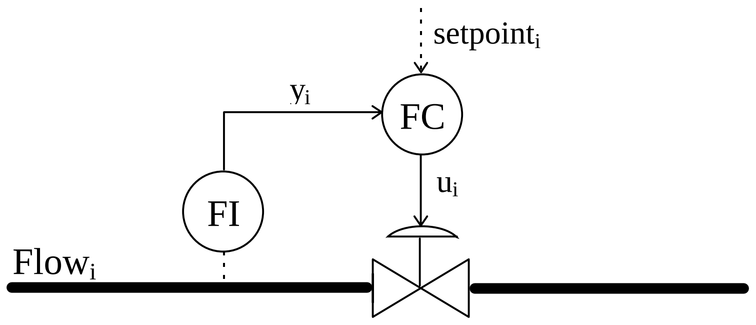


Figure 3.3: Schematic of a decentralized control loop

Using this notation, the schematic of the controlled system is shown in Figure 3.4 where each of the setpoints of the flow and temperature controllers have yet to be defined. Using the flow and temperature indicators for feedback control of the valves makes the setpoints of these ten devices become the degrees of freedom for the system.

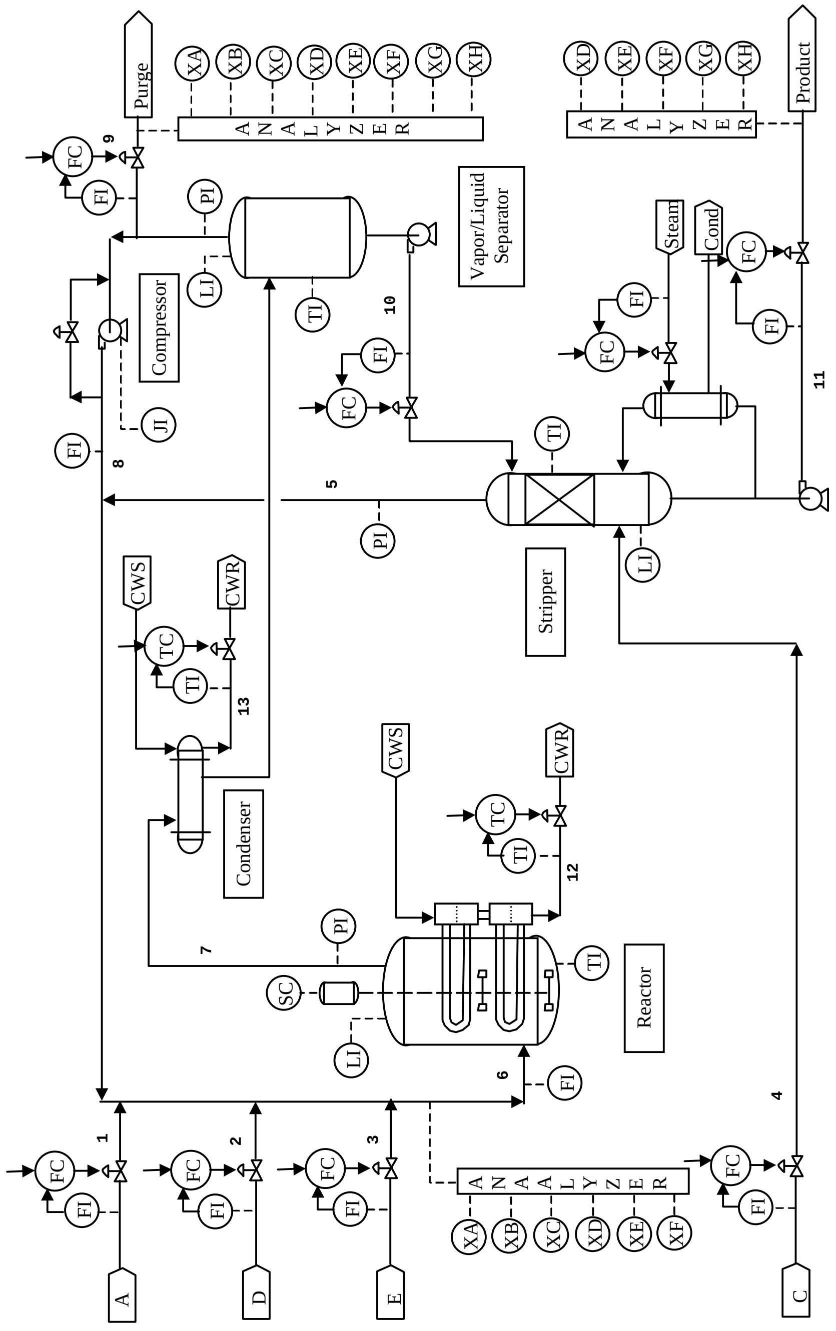


Figure 3.4: Flowsheet with flow and temperature controllers

3.2.2 Loop Pairings for Level Controllers

Closing the flow and temperature loops in the first tier does not stabilize the system. The reactor pressure builds up and in 3.6 hours causes the plant to shut down. To stabilize the system, the three tank levels of the process must be placed under feedback control. Tank levels in chemical processes are sometimes allowed to float because as long as the tanks do not overflow or run dry, they are at an acceptable level. For these reasons designers find it justifiable to use a simple proportional controller for tank levels. In this particularly difficult process the controllers of the tank levels contain small integral gains to achieve setpoint tracking, avoid plant shutdown conditions, and maximize the operating costs.

From the system diagram the best possibilities for controlling the separator level are either the condenser cooling water or separator bottoms flow valve. An increase in the condenser cooling water increases the amount of liquefied chemicals, forcing the separator level to increase. The separator bottoms flow valve directly controls the separator level, therefore is the better choice. The controller should be reverse-acting making the proportional gain negative. As the valve is opened in the positive direction, the tank level decreases because the measurement device is downstream from the controlled variable.

From the system diagram the best manipulated variables for controlling the stripper level is either the flow in or the flow out of the tank. The flows into the tank are steam 4 and the separator output stream. Of these two inputs, flow from the separator is much more significant, but is already being used for the separator level control. Since stream 4 is small in comparison, the logical choice for stripper level control is the flow output. The setpoint of the product flow controller of tier one will be the manipulated variable for stripper level control. This controller will also have a negative gain because the controller is located downstream from the controlled variable.

The reactor lacks a valve on the input or output of its tank as in the separator and stripper, consequently a different methodology must be used. To control the reactor level, one of the input gas streams should be manipulated. Feed stream 3 (component E) has more holdup from upstream processes than streams 1, 2, and 4 (components A, C, and D) [6]. Where streams 1, 2, and 4 have limitations on their variability, stream 3 is not constrained. Therefore, the choice for the reactor level control is the setpoint of the E feed.

Another choice for reactor level control is to use the condenser cooling water to adjust the amount of the reactants to be recycled. If the temperature in the condenser is increased, more reactants stay in the vapor phase. The extra vapors create a larger recycle flow that increases the reactor level. Similarly, a lower condenser temperature produces more liquid product, resulting in fewer recycled vapors. This could lead to suitable configurations, but is not explored here because the condenser cooling water has large effects on other process variables.

The PI gains for the tank levels are adjusted by experience and trial and error. The integral time for each of the level controllers is set to 2 hours because exact setpoint matching is not necessary. The proportional gains are selected to avoid oscillations in the tank levels; their controller gains are listed in Table 3-3. The level responses are considerably slower than those of the flows and temperatures, therefore the controller variables are updated on intervals of 30 seconds.

Table 3-3: PI Controller Gains for Level Control Loops

Controlled	Manipulated	K_c	T_I (min.)
Stripper Level (XMEAS15)	Stripper Underflow Setpoint (SETPT17)	–0.25	120
Separator Level (XMEAS12)	Separator Underflow Setpoint (SETPT14)	–1	120
Reactor Level (XMEAS8)	E Feed Setpoint (SETPT3)	400	120

3.2.3 Design of the Outer Loop Cascade Controllers Around the Base-Case

Once the above controllers are applied, the system is stable for the base-case and does not depend on any of the analyzers. Since analyzers are fairly unreliable devices, the control algorithm must result in stable configurations without analyzer data. The next step is to create controllers to handle the various operating conditions and disturbances. Ideally, the system should operate over all six modes, even with the disturbances active. This is not a realistic goal for this system, consequently the main goal as mentioned before is to operate in the first mode with as few overrides as possible.

3.2.3.1 Material Balance and Mass Transfer

Two important system objectives are to allow operation with different G to H ratios and production rates. Downs and Vogel [6] provide Heat and Material Balance Data for the base-case. McAvoy and Ye [15] have done an extensive study using a simplified material balance to determine the variables for product quality and production ratio. A similar analysis is done here that finds the same conclusions. To keep the system stable, the system at steady state must meet a material balance in that no liquids or temperatures can be increasing or decreasing. Base-case results show that the purge flow only makes up 2.7% mass flow of the total output. Since the majority of the components of the four inputs (A, C, D, and E streams) exit through the product, a simplified material balance leads to neglecting the purge. In the product flow, the D, E, and F components make up less than 1% of the total mass flow, therefore can be disregarded in the simplified material balance. Liquid products G and H make up the majority of the product stream. The resulting simplified equations are the first two chemical reactions in Equations 1.1 and 1.2 which are repeated here.

$$A(g)+C(g)+D(g)\rightarrow G(liq) \tag{3.11}$$

$$A(g)+C(g)+E(g)\rightarrow H(liq) \tag{3.12}$$

Since the A and C products are present in both equations, the A and C reactants are potential candidates for production rate. The A reactant is a small quantity in comparison to the total product and enters the system in both streams 1 and stream 4. The C reactant is a much larger quantity and only enters in stream 4, consequently stream 4 is used to control the production rate.

In the above equations it can also be seen that one mole of the D reactant is a component in production of one mole of the G product. Similarly, one mole of the E reactant leads to one mole of the H product. A ratio controller for the D/E setpoint is the logical choice for control of the G/H ratio. Since the E feed already controls the reactor level, the D feed will control the product quality through the ratio controller. The setting on the D feed setpoint will be determined as follows from the “velocity” PI equation.

$$err(k) = \left(G/H\right)_{stpt}(k) - \left(G/H\right)_{meas}(k) \tag{3.13}$$

$$\left(D/E\right)_{stpt}(k) = \left(D/E\right)_{stpt}(k-1) + K_c\left[err(k) - err(k-1) + \frac{T_s}{T_I}err(k)\right] \tag{3.14}$$

$$D_{stpt} = \left(D/E\right)_{stpt} \cdot E_{stpt} \tag{3.15}$$

Since the D flow has limits on its variability, the D setpoint also cannot change too quickly. If the E setpoint changes to rapidly, the E setpoint of Equation 3.15 should be replaced by an averaged version of the E setpoint.

The PI controller gains for the product rate and G/H ratio are determined empirically. Initial guesses assume that the loops should respond slower than the flow and temperature loops, but faster than the level control loops. The tuned gains are shown in Table 3-8.

3.2.3.2 Loop Pairings Through the Relative Gain Array (RGA)

A complete derivation of the RGA can be found in [11]. It is a technique that has been used in several industrial processes to determine loop pairings based on interaction of controlled variables with manipulated variables. In an RGA analysis, one first determines, around a linearized version of the system, steady-state open-loop gains of outputs for step changes in the inputs. The outputs to be controlled must be selected beforehand based on the system specifications. In multi-input multi-output chemical processes, a change in an input value affects several controlled variables, but each input influences some variables more than others. The steady-state gains form a matrix K_p which are used to create the RGA matrix whose elements are defined by

$$\beta_{ij} = \left(ij^{th} \text{ element of } K_p\right)\left(ij^{th} \text{ element of } \left[K_p^{-1}\right]^T\right) \tag{3.16}$$

Once all elements of the RGA matrix are determined, each row and column sums to one. RGA elements with gains near unity are good candidates for pairings. Values less than 0.5 show little interaction between the controlled and manipulated variable in comparison to other pairings.

To verify the steady-state gain matrix K_p leads to stable controllers, a test using the Niederlinski index (NI) [11] is often times used. Although the Niederlinski index does not guarantee stability, a system that passes the Niederlinski test, is almost always stable. The Niederlinski index is defined as

$$NI = \frac{Det(K_p)}{\prod K_{p_{jj}}} \tag{3.17}$$

where K_p is the gain matrix and j goes from one to the rank of K_p. If NI is a positive result, the system should be stable.

The remaining important controlled variables are listed in Table 3-4 along with nominal gains for step changes of ±1% in the manipulated variables. Each of the selected controlled variables of column one in Table 3-4 are justified below. In a plant all of the levels, temperatures, and pressures should behave in a controlled manner. Since the tank temperatures should be controlled to prevent a buildup of overall temperatures in the system, each of the temperature measurements are potential controlled variables. Pressures are also very important, and in this case if the reactor pressure increases beyond 3000 kPa, the plant shuts down. One of the test disturbances recommended by Downs and Vogel [6] increases the amount of the inert gas B in the purge stream. The inert gas should be controlled to prevent excess reactant gases building up. The compressor power and recycle flow are selected as controlled variables because they have significant effects on plant economics. The available inputs to control these variables are the degrees of freedom remaining after system stabilization and product flow manipulation. The RGA must be a square matrix to determine loop pairings, so originally, the separator temperature was selected as a controlled variable instead of the recycle flow. It was found through an RGA analysis that the separator temperature does not have any viable loop pairings.

From the steady-state gain analysis, as shown in Table 3-4, it is apparent that there is a strong correlation between reactor pressure, separator pressure, and stripper pressure; the three measured pressures cannot be controlled independently. Of these three pressures, reactor pressure has an upper limit that can cause the plant to shutdown; the reactor pressure will be one of the controlled variables, and the other two will stay within acceptable limits. There is also a strong correlation between the responses to a change in the agitation rate and a change in the reactor cooling. Note that the nominal gains in Table 3-4 for the reactor cooling is approximately –5 times that of the agitation rate. Since the reactor cooling is a significant portion of the operating costs, the agitation rate will temporarily be dropped as a manipulated variable. The problem now becomes a 6x6 problem where the RGA analysis can be performed. The value of NI for the 6x6 problem using Equation 3.17 is 41.5, verifying the pairings lead to a stable configuration given proper controller gains.

Table 3-4: Important Controlled and Manipulated Variables

		Manipulated Variables						
		A feed	**Steam**	**Recycle**	**Purge**	**RCWOT**	**CCWOT**	**Agitation**
Controlled Variables	**Reactor Temperature**	7.9834	0	0	0	1.0994	0.0259	–0.22
		7.9834	0	0	0	1.1205	0.0259	–0.22
	Stripper Temperature	45.505	0.0369	–0.041	19.874	1.8034	0.4075	–0.360
		47.102	0.0369	–0.036	20.467	1.9281	0.4179	–0.362
	Separator Temperature	67.061	–0.002	–0.194	21.061	2.1829	0.2820	–0.432
		69.056	–0.002	–0.198	20.467	2.3171	0.2885	–0.438
	Reactor Pressure	–3074	0.0434	7.6542	–949.2	–28.01	12.420	6.2
		–3153	0.0434	7.6542	–978.9	–35.10	11.514	6.0
	Stripper Pressure	–3353	0.0434	3.602	–1038	–30.66	13.455	6.8
		–3433	0.0434	3.602	–1038	–38.37	12.42	6.6
	Separator Pressure	–2994	0.0434	8.5547	–919.6	–27.48	12.290	6
		–3114	0.0434	8.5547	–949.2	–34.46	11.385	6
	B Comp in Purge	0	0	0	–40.64	0	0	0
		0	0	0	–41.23	0	0	0
	Compressor Power	127.73	0.0130	2.9716	–26.70	–4.366	2.5615	0.88
		123.74	0.0087	2.9716	–29.66	–4.672	2.4839	0.86
	Recycle Flow	–10.78	0.0009	–0.014	–6.822	–0.614	0.2704	0.126
		–11.58	0.0004	–0.023	–7.119	–0.687	0.2587	0.124

Table 3-5: RGA for Tier Two Controlled and Manipulated Variables

		Manipulated					
		A feed	**Steam**	**Recycle**	**Purge**	**RCWOT**	**CCWOT**
Controlled	**Reactor Temperature**	–0.0144	0	0	0	**0.9577**	0.0567
	Stripper Temperature	0.0004	**1.0228**	0.0016	0	0.0028	–0.0277
	Reactor Pressure	**0.9778**	0.0161	0.1636	0	–0.0763	–0.0811
	B Comp in Purge	0	0	0	**1**	0	0
	Compressor Power	0.1065	–0.0094	**0.7743**	0	0.0316	0.0971
	Recycle Flow	–0.0703	–0.0294	0.0606	0	0.0842	**0.9550**

The RGA matrix in Table 3-5 shows the results from the RGA algorithm once the nominal gains for positive and negative steps are averaged. There are no large numbers in the matrix and several of the elements have gains close to unity showing strong interaction between manipulated and controlled variables. The numbers in bold are used to develop the following pairings:

Table 3-6: Loop Pairings for the Second Tier

Controlled Variable	Manipulated Variable
Reactor Temperature	RCWOT Setpoint
Stripper Temperature	Stripper Steam Flow Setpoint
Reactor Pressure	A Feed Setpoint
B Composition in Purge	Purge Rate Setpoint
Compressor Power	Compressor Recycle Valve
Recycle Flow	CCWOT Setpoint

3.2.3.3 Tuning Procedure for PI controllers

Once pairings have been established, the PI controllers gains must be determined. A common method used in SISO or decentralized systems is use of the Ziegler-Nichols rules [11]. The Ziegler-Nichols rules usually lead to systems that are quick responding, but have slightly underdamped responses. To use the Ziegler-Nichols rules the two variables needed are the ultimate gain K_u and the ultimate period P_u. The first attempt at PI controller gains are

$$K_c = \frac{K_u}{2.2} \tag{3.18}$$

$$T_I = \frac{P_u}{1.2} \tag{3.19}$$

for a PI controller with the transfer function

$$Out(s) = K_c(1 + \frac{1}{T_I s})In(s) \tag{3.20}$$

One way to calculate the ultimate period and gain is to close the loop on a proportional only controller and increase the gain until the output oscillates. To reduce the effects of the system nonlinearities a better way, which is used here, follows the autotune variation (ATV) procedure [11]. In the ATV method, the manipulated variable is given a fixed positive step change and the output variable's response is examined. Once the output variable crosses its corresponding setpoint, the polarity on the manipulated variable's step change is reversed. The output response will begin to change direction. When the output reaches its setpoint again, the polarity of the change in the manipulated variable is reversed again. After a few periods the transients in the output signal will asymptotically approach zero and a limit cycle will have been produced. The period of the oscillation is the ultimate period P_u. The ultimate gain K_u is calculated by

$$K_u = \frac{4}{\pi}\frac{\Delta m}{\Delta h} \tag{3.21}$$

where Δm is the change in the manipulated variable and Δh is the change in the controlled variable. An example is shown below using the TE process where the recycle valve is the manipulated variable and the compressor power is the output variable.

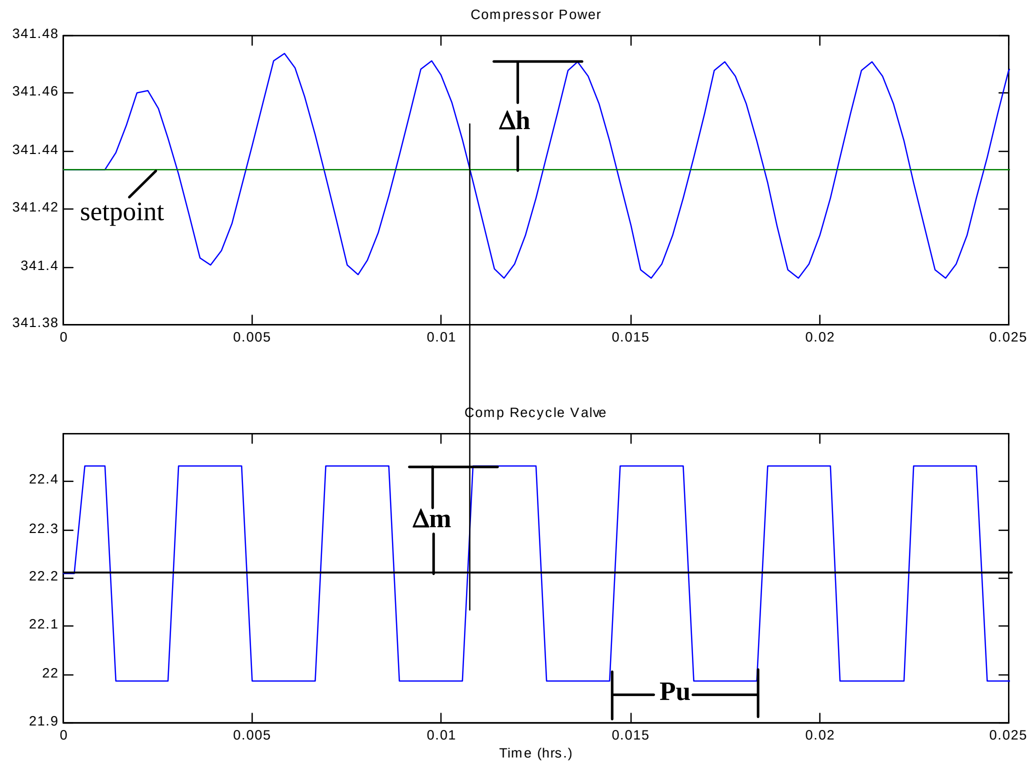


Figure 3.5: Example of the ATV algorithm

The measured values from the autotune method on the compressor are

$$K_u = \frac{4}{\pi}\frac{\Delta m}{\Delta h} = \frac{4}{\pi}\frac{2.2}{0.36} = 7.85 \tag{3.22}$$

$$P_u = (0.0039 \text{ hr})(60 \ \frac{\text{min}}{\text{hr}}) = 0.234 \text{ min} \tag{3.23}$$

The corresponding PI controller gains using Ziegler-Nichols rules become

$$K_c = \frac{K_u}{2.2} = 3.57 \tag{3.24}$$

$$T_I = \frac{P_u}{1.2} = 0.2 \text{ min} \tag{3.25}$$

When the ATV method is used on the loop pairings of Table 3-6, the results in Table 3-7 were found. For this analysis the flow, temperature, and level controllers are in place to stabilize the system. To verify that these gains are sufficient, a dynamic simulation must be performed. Once simulated it was apparent the gains are not sufficient mainly because the Ziegler-Nichols rules usually gives underdamped responses which in this case causes plant shutdown. For multivariable processes the Ziegler-Nichols based controllers usually have to be detuned even more than for SISO processes. Using the Ziegler-Nichols rules as a starting point, the PI gains were adjusted through dynamic simulations to give good responses to the disturbances and other changes in operating conditions. The final PI controller gains for the second tier of the TE process (excluding level controllers) are shown in Table 3-8. The controllers are implemented using the "velocity" form of the PI equation with a sample time of 30 seconds. The setpoints of some of the controlled variables must be fixed to keep a stable system around the operating parameters, but the majority can be changed to reduce operating costs.

Table 3-7: Initial PI Controller Gains from ATV Algorithm

Controlled	**Manipulated**	**K_c**	**T_I (min.)**
Reactor Temperature (XMEAS9)	RCWOT Setpoint (SETPT21)	6	1.25
Stripper Temperature (XMEAS18)	Stripper Steam Flow (SETPT19)	80	2.5
Reactor Pressure (XMEAS7)	A Feed Setpoint (SETPT1)	–0.003	150
B Composition in Purge (XMEAS30)	Purge Rate (XMEAS10)	–1.3	30
Compressor Power (XMEAS20)	Compressor Recycle Valve (XMV5)	3.6	0.2
Recycle Flow (XMEAS5)	CCWOT Setpoint (SETPT22)	33	0.7

Table 3-8: Final PI Controller Gains for the Second Tier

Controlled	Manipulated	K_c	T_I (min.)
G/H Ratio (XMEAS40/XMEAS41)	D/E Setpoint (STPT2/STPT3)	0.1	50
Product Flow Setpoint (STPT17)	A&C Feed Setpoint (SETPT4)	0.1	50
Reactor Temperature (XMEAS9)	RCWOT Setpoint (SETPT21)	0.375	20
Stripper Temperature (XMEAS18)	Stripper Steam Flow (SETPT19)	8	25
Reactor Pressure (XMEAS7)	A Feed Setpoint (SETPT1)	–0.003	150
B Composition in Purge (XMEAS30)	Purge Rate (XMEAS10)	–0.13	300
Compressor Power (XMEAS20)	Compressor Recycle Valve (XMV5)	0.036	20
Recycle Flow (XMEAS5)	CCWOT Setpoint (SETPT22)	1	25

The final controlled system is displayed in Figure 3.6. The tier two design measurement devices have been redrawn as controller devices where the dotted lines indicate setpoint changes in another control loop. This is one of hundreds of configurations that could create a stable system, but this particular configuration meets the performance criteria and is relatively fast responding. In mode one it also handles all of the recommended disturbances and setpoint changes without the plant shutting down. Of all the listed disturbances, IDV(6) is the only one that creates a plant shutdown condition. This disturbance is not one of the disturbances recommended for testing by Downs and Vogel [6], but rejection of the disturbance will be addressed later. It was attempted to change the system to the other operating modes, but was found that several intermediate steps are needed to change between modes.

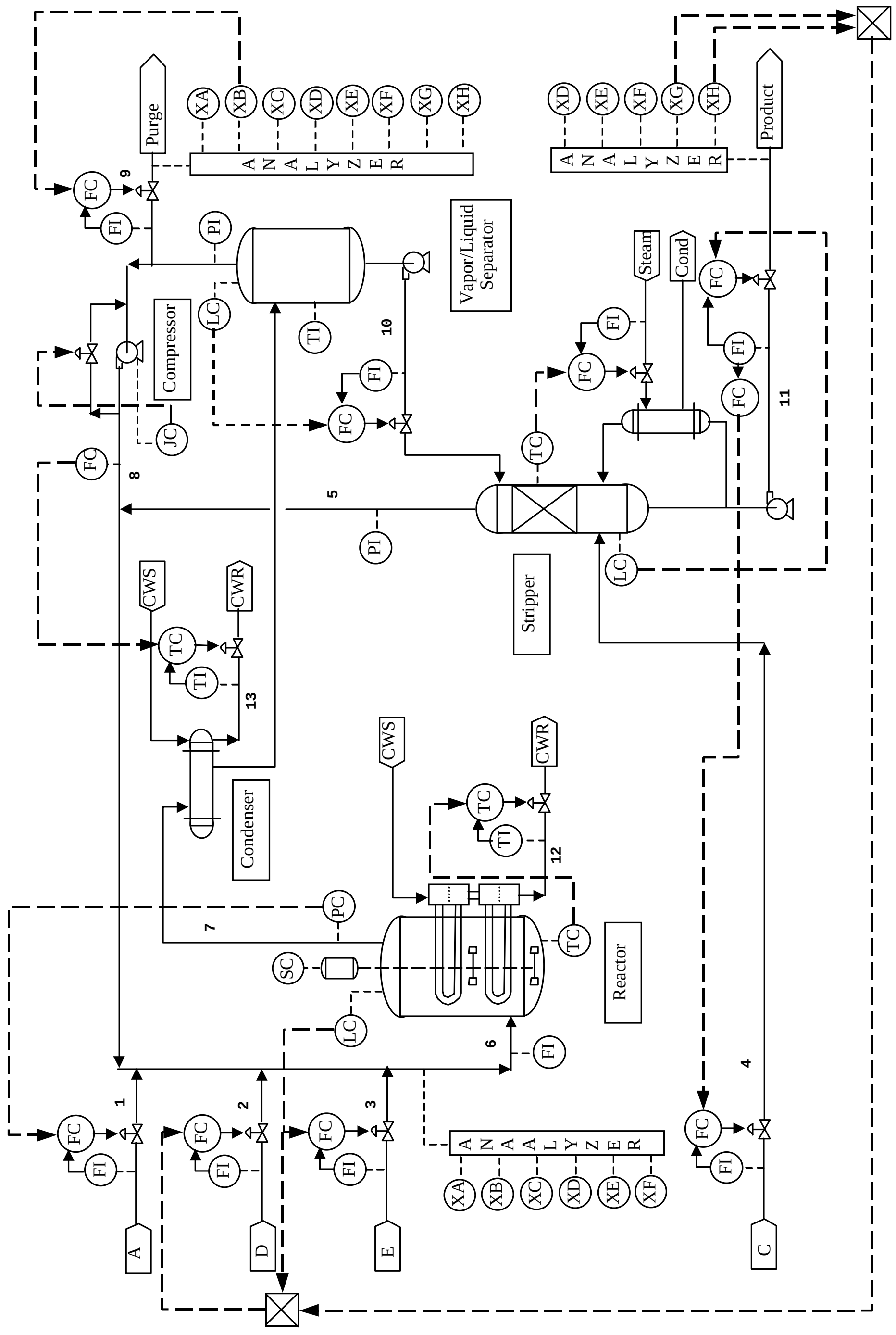


Figure 3.6: Plant diagram of decentralized controlled system

3.2.4 Performance Validation of the Decentralized Control System

The results from a typical dynamic simulation of the controlled system are presented in Figures 3.7 through Figures 3.9. The simulation outputs are for a change from the 50/50 G/H ratio to a 40/60 G/H ratio. The data is plotted every 30 minutes. Figure 3.7 shows the effects of the change in G/H ratio on the input variables. Although the signals appear noisy, none of the valves saturate at the fully-open or fully-closed positions. The agitation rate is a horizontal line because it has not been used in a feedback control scheme. For each measurement output of Figures 3.8 and 3.9 there are two lines. One line represents the measurement output; the other represents the loop setpoint. The eight flow outputs of Table 3-1 respond quick enough that the setpoints appear equivalent to the output. Inspection of the stripper underflow measurement reveals that the production rate does not change significantly for a change in the G/H ratio. For this simulation, the setpoints that are not under feedback control are set at their base-case values. To ensure that inventory control is achieved, the tank levels, pressures, and temperatures must be examined. The reactor, separator, and stripper tank levels stay well within the normal operating limits of Table 1-3. The three pressures follow similar trajectories, as expected. The reactor pressure stays below the shutdown limit of 3000 kPa, therefore the other pressures are adequate. Of the three temperatures, the reactor temperature stays below the normal operating limit of 150°C. Since the stripper and separator temperatures on average are not increasing or decreasing, they also meet inventory control specifications. Examination of the G/H ratio and product G reveals that the 40/60 ratio is achieved within 15 hours without much fluctuation. Even though considerable measurement noise is present, there are no significant oscillations. The controlled system meets all of the necessary performance criteria for a change in the G/H ratio.

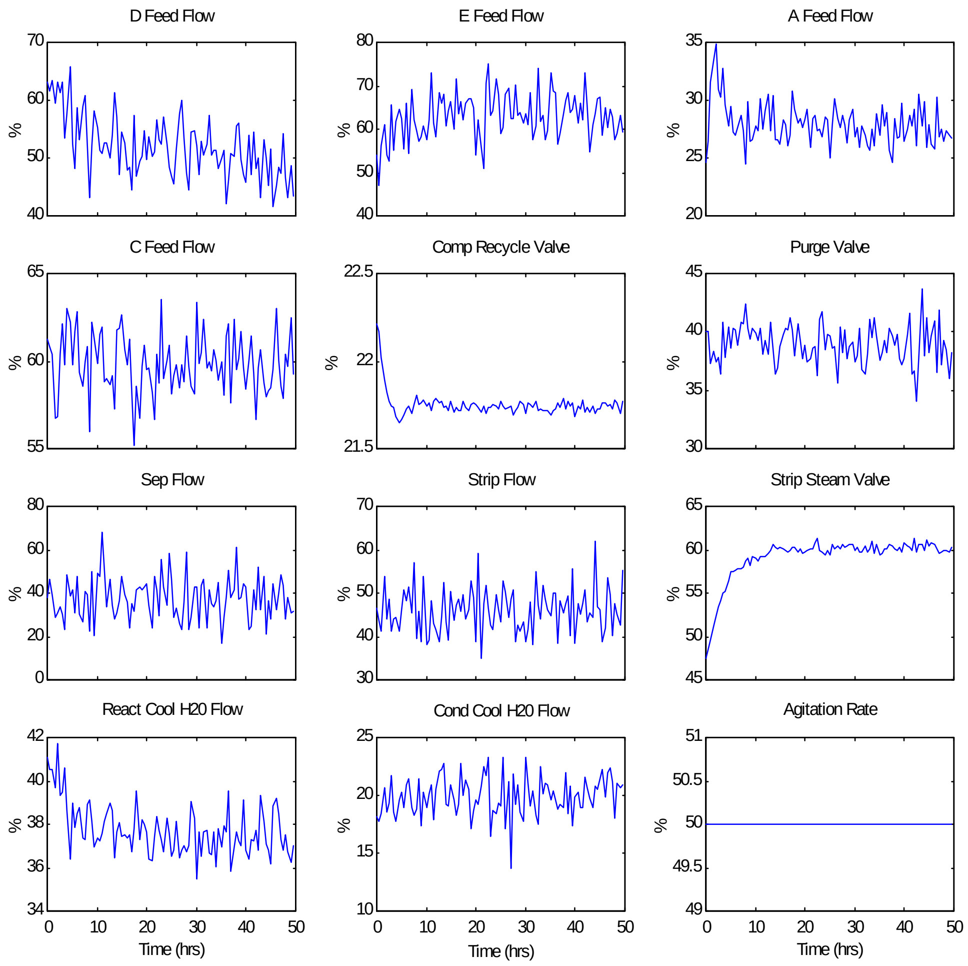


Figure 3.7: System inputs for change from 50/50 to 40/60 G/H ratio

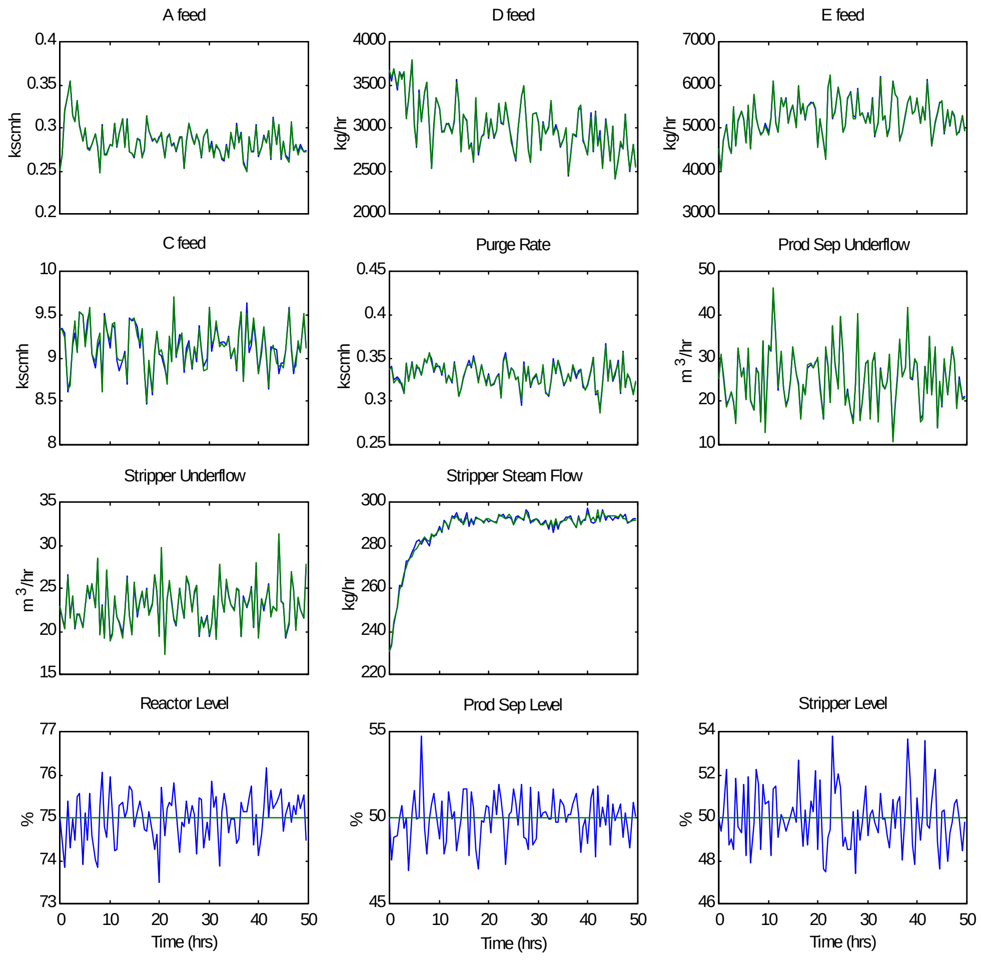


Figure 3.8: Selected outputs for change from 50/50 to 40/60 G/H ratio

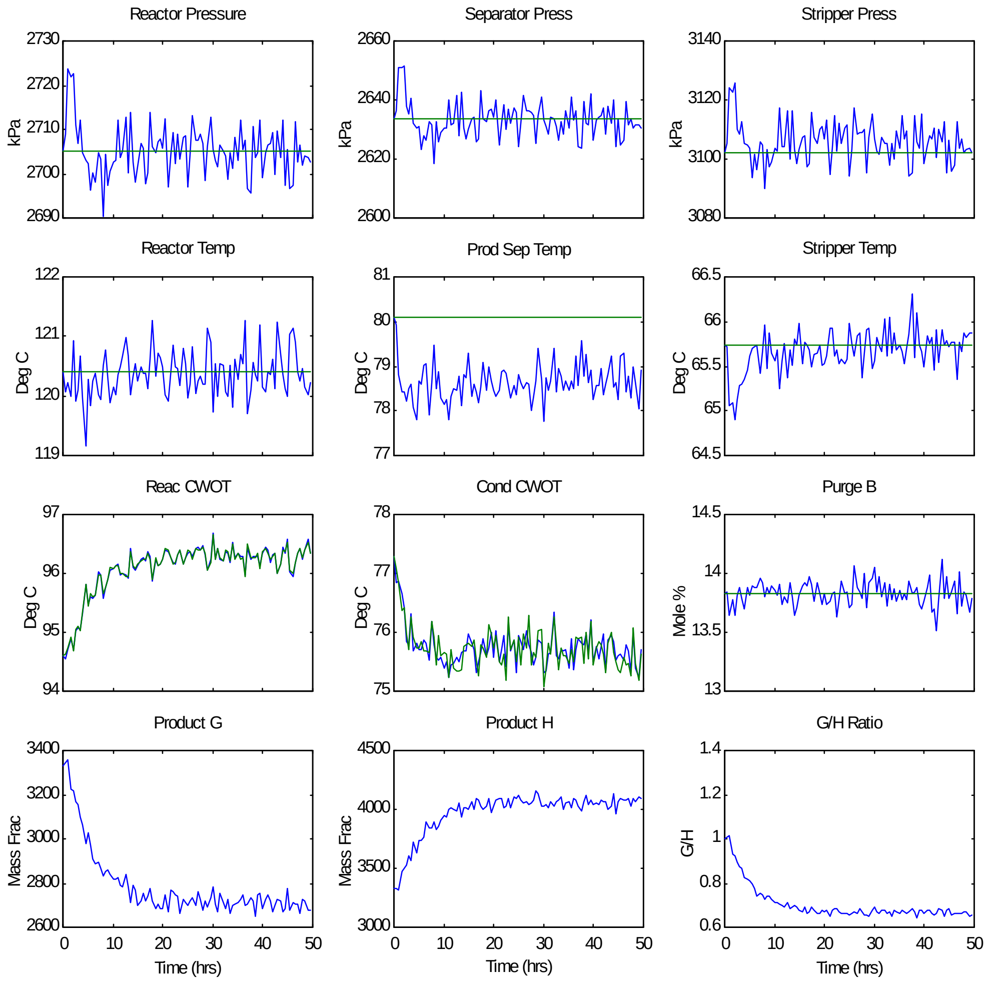


Figure 3.9: Selected outputs for change from 50/50 to 40/60 G/H ratio

The controlled system is robust enough to handle all of the recommended setpoint changes, simultaneously. Figures 3.10 and 3.11 display selected outputs when the production rate decreases by 15%, the product mix changes from a 1:1 to a 2:3 G to H ratio, the reactor pressure is decreased by 60 kPa, and the composition of B in the purge is increased by 2%. No significant oscillations or fluctuations of the product and inventory variables occur when the system converges to the new setpoints. The outputs are far away from the plant shutdown constraints and the valves do not saturate when any of the setpoint changes occur.

The controlled system also stays within the control limits when any of the recommended disturbances are active. The only disturbance that has significant output effects is IDV(8). Results from this disturbance for 100 hours of operation are displayed in Figures 3.12 and 3.13. The figures display variability in some of the variables, due to the disturbance being a random variation of one of the feed streams. Figure 3.12 shows that the A feed briefly saturates at its lower limit while controlling the reactor pressure. This may potentially cause a problem in the actual implementation, but is unlikely to cause any significant harm to the system.

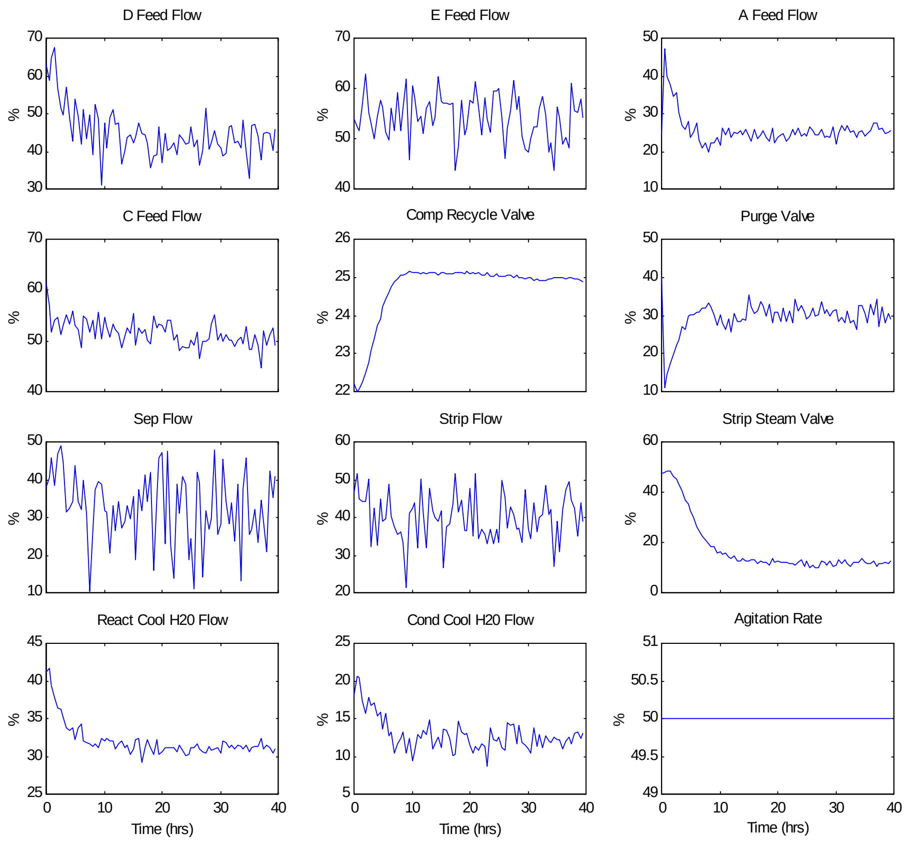


Figure 3.10: Input signals for recommended setpoint changes

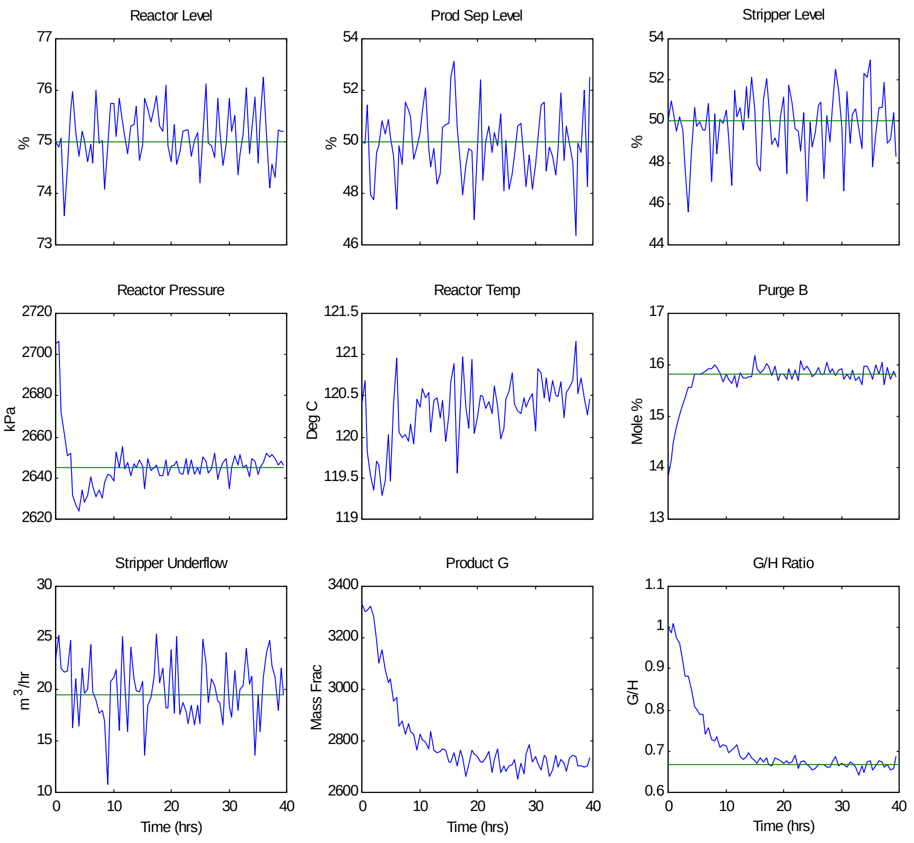


Figure 3.11: Selected outputs for recommended setpoint changes

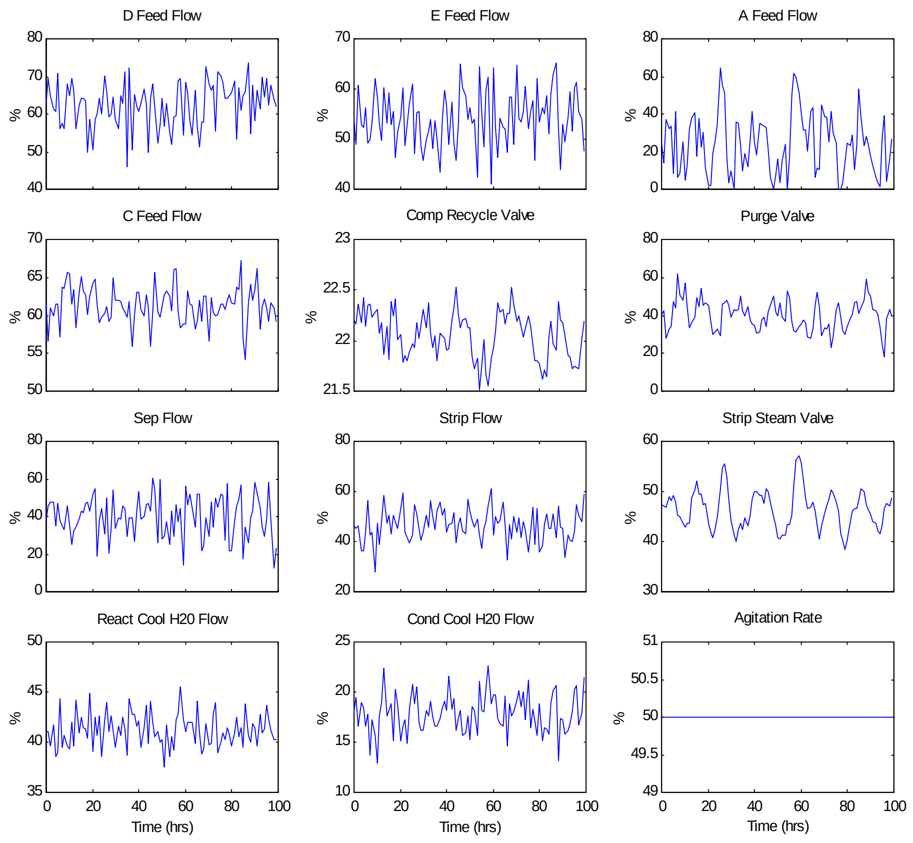


Figure 3.12: Input signals with IDV(8) active

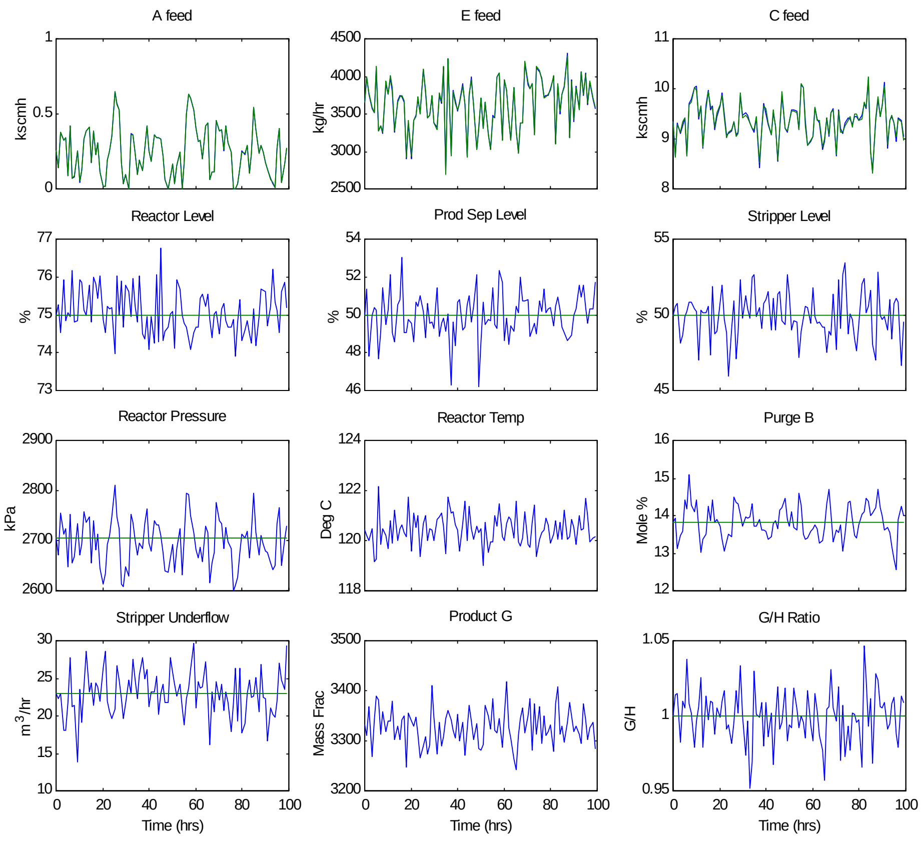


Figure 3.13: Selected outputs with IDV(8) active

3.3 Overview of the Decentralized Control Scheme as Applied to the TE Process

In this chapter the basic decentralized controller for the TE process was developed. The procedures for identifying the system and possible control configurations were examined and implemented. The procedures led to base-case PI controller gains that were tuned to give well-conditioned dynamic responses to disturbances and setpoint changes. The overall controlled system was found to be robust for operating changes recommended by Downs and Vogel [6].

In the next section, the setpoints of the system will be varied to keep the system stable and reduce the operating costs. The 12 relative degrees of freedom limit the number of variables that can be adjusted in manipulation of the plant economics. In addition, some of these degrees of freedom must be fixed to guarantee system stability. A genetic algorithm along with the dynamic simulation of the system is used to minimize the system operating costs around the base-case conditions.

Chapter 4

Genetic Algorithm Implementation

In the previous chapter the TE process was controlled to handle disturbances and other plant changes. The PI controllers stabilized the system around the base-case settings. The setpoint values of the outer loop controllers and agitation rate have yet to be specified. In this section, these variables are manipulated through an optimization routine to provide a stable system that minimizes the steady-state plant operating costs. The optimization routine that will be utilized is a genetic algorithm (GA). A GA is a non-gradient based algorithm that searches over a sample space to find solutions that minimize or maximize objective functions.

4.1 Beginnings of the Genetic Algorithm

Genetic algorithms use probabilistic search methods to minimize or maximize the specified objective function. GAs have their foundations in the ideas of natural selection and natural genetics where fit individuals tend to propagate their genetic makeup to produce fit offspring. The algorithm uses random choice as a means to search over large spaces, but is not a purely random search method. It is typically a much more robust routine than other minimization routines such as a gradient search and hill climbing methods. Those methods rely on calculus-base schemes for parameter convergence, but GAs have the advantage that continuity is not required. When properly developed they do not converge and stay at local minimums or maximums. The example shown in Figure 4.1 is one where most optimization routines have difficulty finding the global minimum. Given the function $f(x)$ as shown, it is attempted to determine the minimum on the interval of $0 \leq x \leq 1$. Optimization routines normally require an initial condition to begin their search procedure. If the initial point is chosen at $x(0)=0.5$ most optimization routines will converge to the local minimum at $x=0.75$, whereas the genetic algorithm has a higher likelihood of finding the global minimum at $x=0.25$. Given enough time and the proper structural parameters, GAs can provide a minimum or maximum-cost solution to a particular problem. Their advantage over pure random search methods is achieved by using a directed search in their parameter convergence.

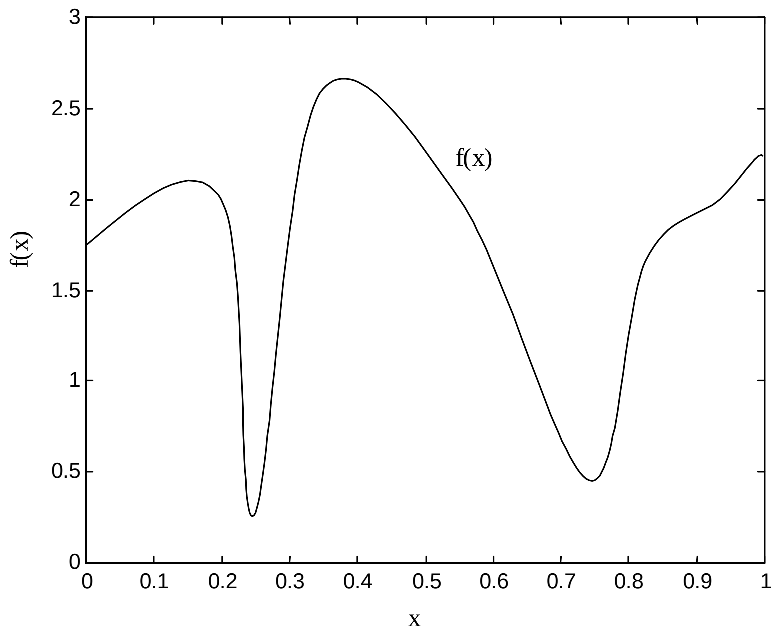


Figure 4.1: Difficult first order problem for most optimization routines

De Jong [5] was the first researcher to explore in detail the idea of a genetic algorithm. In his dissertation, De Jong proved that GAs could solve five difficult mathematical expressions that other routines may, or may not, be able to solve. After De Jong's analysis, John Holland [9] published a book in which he described the four main components of the genetic algorithm: reproduction, crossover, mutation and inversion. Later researchers found that the process of inversion has little effect in most applications. The other three operators are almost always used, with the probabilities of each action depending on the application. Their schemata will be explained later in the context of a simple example.

Following the work of Holland, Goldberg in [7] and Davis in [4] showed how to apply the GA to a wide range of problems. In Goldberg's book, he explains how to prepare the GA in programming code and the structures to use for particular applications. Where Holland [9] had given a theoretical background, Goldberg and Davis show how to implement genetic algorithms. Many of today's genetic algorithms are based on the guidelines presented in their books [7,4].

Since these two groundbreaking books, the genetic algorithm has been applied to a wide range of applications from optimization of composite structures to pattern recognition. Development of the computer code is relatively simple, and its implementation has generally been successful. The biggest drawback of the genetic algorithm in past developments has been the amount of processing time it takes for the algorithm to converge to optimal values. With computer speeds increasing and the use of parallel processing becoming more popular, GAs are becoming more acceptable and a superior alternative to other search algorithms.

4.2 Parameters of the Genetic Algorithm

The basic genetic algorithm can be separated into several steps. A flowchart for the GA process is shown below in Figure 4.2. The main sections of the GA routine that are different from other routines are the functions of reproduction, crossover, and mutation.

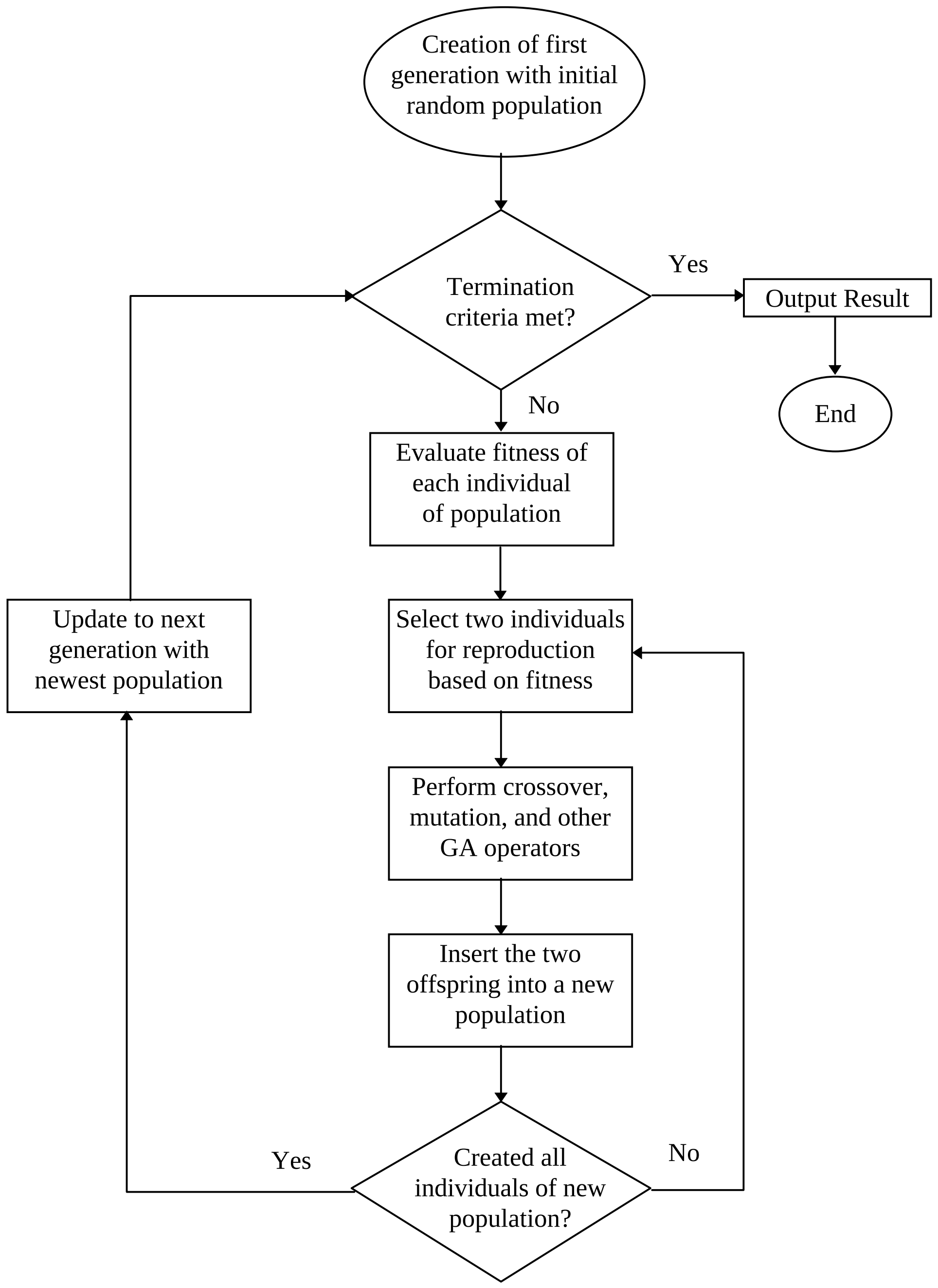

Creation of first generation with initial random population
Termination criteria met?
Yes
Output Result
End
No
Evaluate fitness of each individual of population
Select two individuals for reproduction based on fitness
Update to next generation with newest population
Perform crossover, mutation, and other GA operators
Insert the two offspring into a new population
Yes
Created all individuals of new population?
No

Figure 4.2: Flow chart of the genetic algorithm routine

4.2.1 Creation of the Initial Population

To begin the system operation, an initial random population is created that spans the operating space of the problem. The initial population consists of several individuals defined by their chromosomes. Each individual (chromosome) consists of several genes. To simplify the discussion, think of an individual as a string of bits with each bit corresponding to a gene. The connection to the optimization problem is in how the individual represents a solution. Usually, several genes from the chromosome string construct substrings, or fields, representing parameters of the solution. Figure 4.3 illustrates a substring of an individual that might correspond to a value of a key solution parameter.

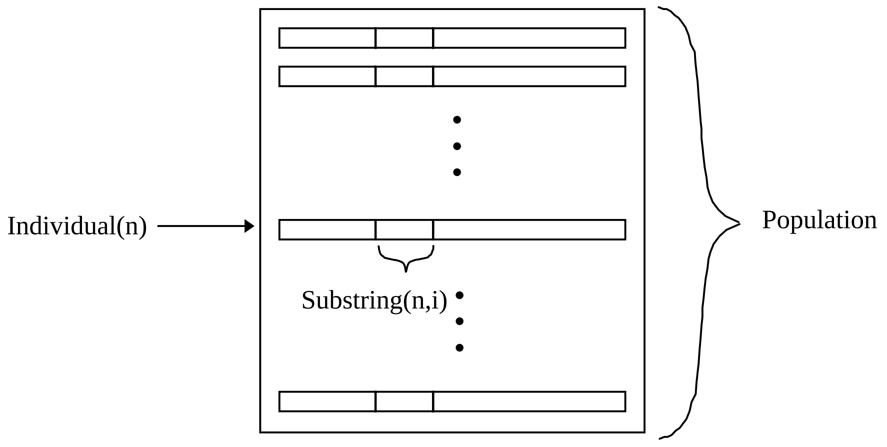


Figure 4.3: Defining genetic algorithm parameters

4.2.2 Reproduction Operator of the Genetic Algorithm

After system propagation, each individual is evaluated based on a fitness function. The fitness of each individual determines which individuals are selected in the reproduction step. There are several methods to implement reproduction, but the one most frequently used is the roulette wheel. The first step in the roulette wheel procedure is to sum together all the fitness function values. The individuals can be placed on a biased circle where each individual takes up an arc of the circle proportional to its fitness value. A random number corresponds to points on the circle. Uniformly random values are selected to fill the mating pool with the corresponding individuals. Once the mating pool is filled, the GA process continues to the next step.

An example of the reproduction procedure using the roulette wheel method is given below. Suppose a population consists of only four individuals with fitness values of 40, 50, 70, and 40. The four values sum to 200. Therefore, on the biased circle individual 1 uses 20% of the area, individual 2 uses 25% of the area, individual 3 uses 35% of the area, and individual 4 uses 20% of the area. The wheel is diagrammed below where each individual has the area corresponding to its fitness value. By selecting several random points along the circle it should be apparent that individual 3 will be chosen most frequently.

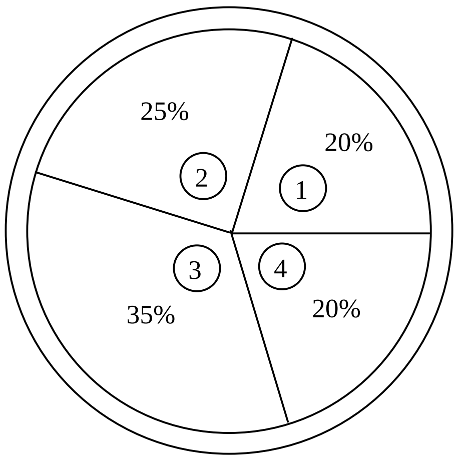


Figure 4.4: Roulette wheel for simple example with four individuals

To implement the roulette wheel reproduction operator in computer code the procedure outlined in Goldberg [7] is computationally fast and efficient. Each individual fitness, *f*, has an associated area that begins at the end of the previous fitness value. The wheel of Figure 4.4 is stretched out to form a biased table normalized between zero and one as shown in Figure 4.5. A uniform random number between zero and one is sequentially compared to the allotted space for each individual. When the upper bound of the individual is larger than the random number, the corresponding individual is selected for the mating pool.

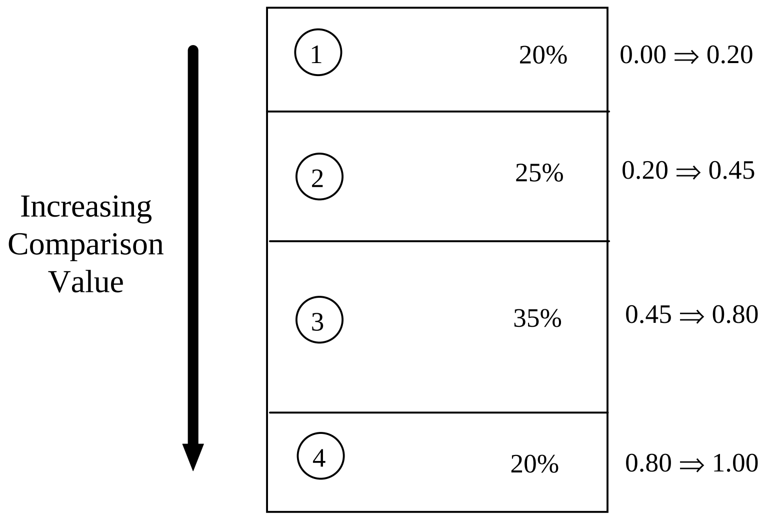


Figure 4.5: Implementation of the reproduction operator in computer code

Tournament selection is another popular method to fill the mating pool where a predetermined number of individuals from the population are randomly selected with replacement. The most fit individual of the selected individuals is placed in the mating pool. This procedure is repeated until the mating pool is filled. Using tournament selection, as opposed to the roulette wheel method, typically leads to quicker convergence, but the population diversity is drastically reduced.

4.2.3 Crossover Operator of the Genetic Algorithm

Two individuals selected from the mating pool proceed to the crossover operation. Not all of the individuals from the mating pool crossover their genes to avoid the new population being worse than the previous population. By reducing the probability of crossover from 100%, the errors from selecting unfit individuals to mate with fit individuals is reduced. If the individuals were not probabilistically chosen for crossover, they are passed on directly to the next generation. If the individuals are scheduled for crossover, simple crossover mates them to produce two new individuals for the next generation. A random point is selected on the chromosome for crossover to occur. The two offspring contain a portion of each parents' genetic makeup. The first child contains the first parent's genes up to the crossover site. After the crossover site, the child has the genes of the second parent. Similarly, the second child has the second parent's genes up to the crossover point and the first parent's genes after the crossover point. The crossover procedure repeats until the mating pool is depleted. An example of the crossover operator is shown below where the random crossover site is displayed by the small arrows.

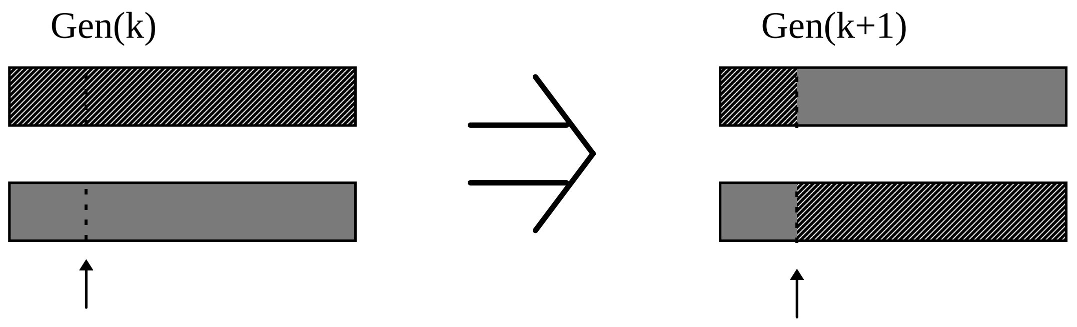


Figure 4.6: Example of the crossover operator

4.2.4 Mutation Operator of the Genetic Algorithm

The last main operator in genetic algorithms based on natural selection is mutation. A mutated individual may have an advantage over other individuals of the population, but in nature very rarely occurs. Since genetic algorithms follow natural selection where mutation happens very infrequently, mutation should have a very small probability of occurrence; its effect should play a secondary role to reproduction and crossover. The main purpose of mutation in genetic algorithms is to prevent the system from reaching and staying at local minima or maxima. If the mutation rate of the system becomes too large, the GA becomes more of a random search algorithm as mutation would then overshadow the reproduction and crossover operators. Mutation is usually implemented in the GA by complementing a random bit (gene) of an individual. An example of mutation is shown below in Figure 4.7 where the individual has been coded using four bits of a binary string. The small arrows in the figure represent the site of mutation.

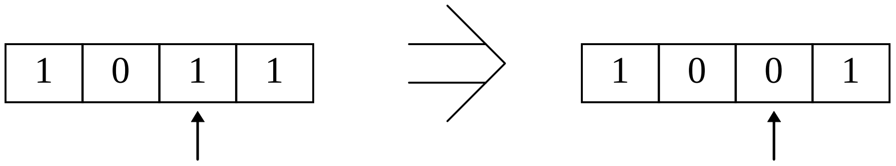


Figure 4.7: Example of the mutation operator

4.2.5 Other Performance Parameters of the Genetic Algorithm

There are other problem-dependent procedures to aid convergence such as fitness sharing, inversion, duplication and deletion, and elitism [7]. However, reproduction, crossover, and mutation, continue to be the fundamental building blocks for almost all genetic algorithms. Implementation of each operator can vary between different applications, but the purpose of each of the basic operators is the same as outlined above.

4.3 Application of the GA to the TE Process

Now that the elementary principles of GAs have been explained, it must be shown how and why it is applied to the Tennessee Eastman problem. The controller presented creates a stable and robust system, but the operating costs are far from optimal. To approach an optimal operating cost, the GA must vary some parameters of the controller.

In genetic algorithms, convergence is usually defined by the convergence of a set of individuals to some good measure of fitness. This is different from other optimization routines where convergence is defined by convergence of a single point to a fit value. For complex systems such as the TE process where there are potentially a large number of inputs, several discontinuities, and several local minima, most optimization routines cannot determine globally optimal parameters. A genetic algorithm searches over the sample space to find the global minimum, instead of starting at a selected point and finding a nearby minimum. It is therefore an ideal procedure for developing the minimum-cost solution of the TE process.

4.3.1 Available Degrees of Freedom in the TE Process

The 12 degrees of freedom of the system are still available as inputs to reduce the plant operating costs. It was attempted not to use the data in Chapter 1 to determine optimal setpoints for two reasons. The first reason is that the values found in Chapter 1 were based on steady-state values for a system without controllers. Although the data found in Chapter 1 was found by a thorough investigation and had a minimum cost less than that found by Ricker [19], there is no way to guarantee that the solution is the global minimum-cost solution. The SOLNP optimization routine largely depends on the initial values like most other optimizing routines, and determining initial values that led to the program converging was a very difficult task.

The second reason the data is not used is that the genetic algorithm should be able to find similar results without prior expert knowledge of the system. The operating parameters of Chapter 1 were obtained by varying the states, and consequently the measurement values. It could be very difficult, or even impossible, to reach these conditions without plant shutdown using only the 12 system inputs. With controllers on the system, some of the steady-state measurement values become fixed, reducing the amount of system freedom. Although the program from Chapter 1 did contain constraints to avoid shutdown, the program did not account for violations of plant shutdown conditions in achieving these values. The values shown in Tables 2-1 to 2-3 are exact points where a slight deviation such as measurement noise may create an unsafe or unstable system.

Of the 12 inputs, at least two of them must have predetermined values; the production rate and G/H ratio are fixed based on the operating mode. Operating mode 1 requires that at steady-state there should be 7038 kg/hr G and 7038 kg/hr H. The remaining 10 degrees of freedom are listed in Table 4-1 along with their recommended operating ranges.

Table 4-1: Degrees of Freedom Available for Tuning

System Input	Input Range
Reactor Pressure Setpoint	2600-2895 kPa
Reactor Level Setpoint	50-100 %
Reactor Temperature Setpoint	120-150 °C
Product Separator Level Setpoint	30-100 %
Stripper Level Setpoint	30-100 %
Agitation Rate	0-100 %
Recycle Flow Setpoint	22-37 kscmh
Compressor Power Setpoint	220-400 kW
B in Purge Setpoint	0-50 Mole %
Stripper Temperature Setpoint	55-75 °C

Of these 10 variables, the first five have limits on their values as shown in Table 1-3. To avoid plant shutdown, the setpoints of these five controllers are constrained within the normal operating limits. Having some of these variables close to the operating limits has significant economic effects, but shutdown must be avoided at all costs. Operating within the normal limits leaves extra margins for transients, disturbances, and changes in operating conditions.

The agitation rate can be varied from zero to 100%. The ranges of the four remaining variables in Table 4-1 were selected based on dynamic responses. The two criteria for determining the range is to determine when the change in the setpoint causes a valve to saturate or the plant to shutdown. If either occurs, an appropriate minimum or maximum setpoint can be concluded. When testing for the range limits, the setpoints (except the one being tested) are fixed at their base-case values.

An example of the range finding procedure for the compressor power is shown below in Figures 4.8 and 4.9. When the setpoint of the recycle controller changes to 230 kW the compressor recycle valve almost saturates to the fully closed position as the measurement value attempts to track the setpoint. Saturation of a valve is undesirable in a plant setting, but by allowing the valves to saturate for this procedure, the end simulation may show the valve should be fully opened or closed. To allow saturation for the compressor recycle flow the lower bound of the compressor power setpoint is 220 kW.

When the compressor power setpoint changes to 410 kW, the additional recycled chemicals cause the reactor pressure to build up, and within 15 hours the plant shuts down. Unlike valve saturation, shutdown must be avoided at all times. The upper bound on the compressor power setpoint is derated to 400 kW.

The range finding procedure is also implemented to find suitable lower limits on the reactor pressure and reactor temperature. When the reactor pressure decreases below 2600 kPa, or when the reactor temperature decreases below 120°C, the system reaches shutdown conditions. These two values become the lower limits of the reactor pressure and temperature setpoints as shown in Table 4-1.

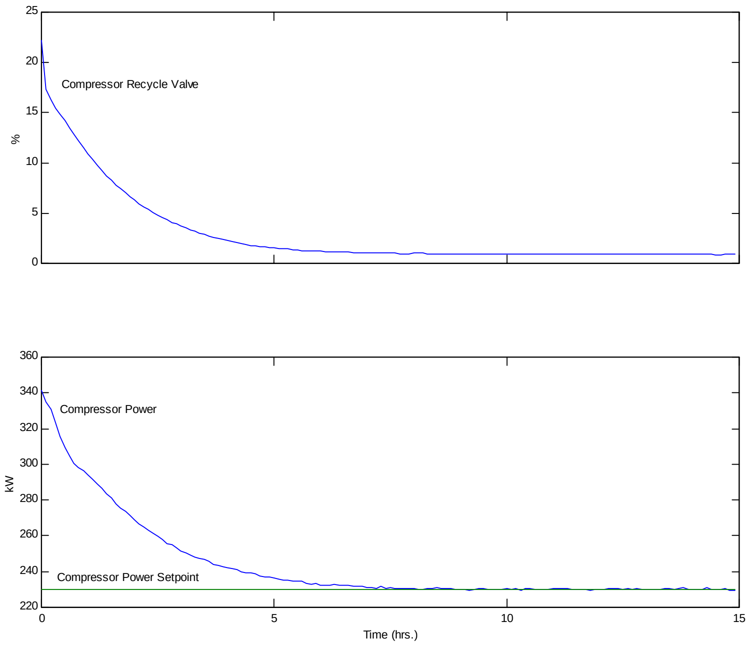


Figure 4.8: Determining the lower bound for the compressor power setpoint

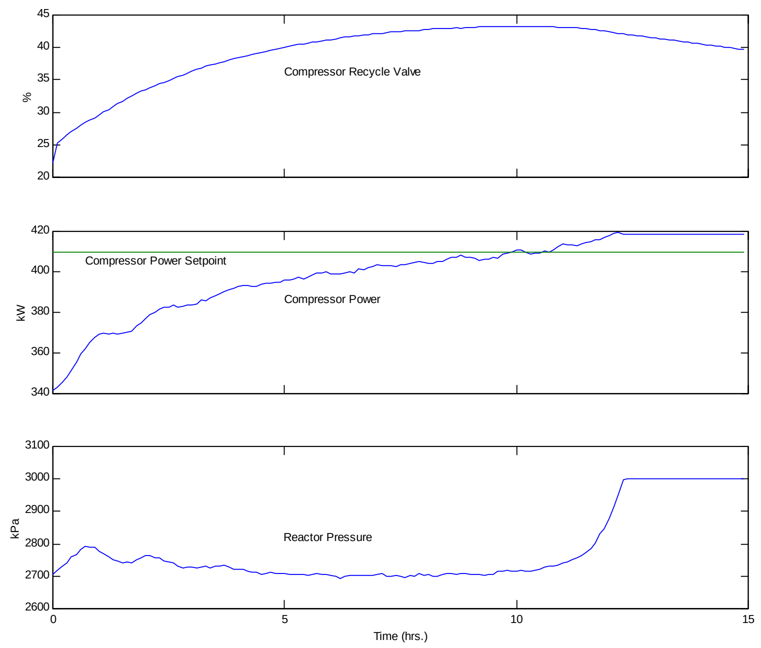


Figure 4.9: Determining the upper bound for the compressor power setpoint

4.3.2 Converting Genetic Algorithm Data to Input Data

The first step in the flowchart of Figure 4.2 is to create the initial random population. The individuals are generated using uniformly distributed random numbers in binary unsigned integer form (1's and 0's). The fixed length individuals will be converted to the setpoints and agitation rate using five bits of resolution per input, making each individual 50 bits long. The binary data next can be converted to base 10 values. For a setpoint with five bits of resolution and the string,

[0 1 1 0 1]

the setpoint becomes

$$\begin{aligned} setpt(i) &= offset(i) + \left(0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0\right)\cdot\left(range\ size(i)\right)/31 \\ &= offset(i) + (13)\left(range\ size(i)\right)/31 \end{aligned} \qquad (4.1)$$

The offset value is the minimum input value calculated in the range finding procedure. When the variable is below this value, no useful data is achieved from a simulation because the plant will shutdown or a valve will saturate. The prefix for the range size using 5 bits of resolution has integer values between zero and 31. More accurate data can be achieved by allowing each degree of freedom more resolution, but for this application five bits is a good trade-off for enough variance and sensible individual sizes.

As mentioned above, certain variables have larger effects on plant stability and operating costs, therefore it is reasonable to allow these variables to have more resolution. The entire input vector is used because by fixing one or more inputs, another input may be limited from achieving its optimal value. The interaction between the inputs is a key component to achieving optimal values. By using all of the inputs with an equal amount of resolution, the system can still be viewed as a black box where each input is given equal weight in the result.

4.3.3 Crossover Sites for the TE Process

The implementation of the genetic algorithm in this study is different than in most other studies. Usually, each individual transforms into one or two substrings with different amounts of resolution, but in this case the algorithm uses 10 substrings all with the same amount of resolution. In certain applications choosing more than one crossover point results in quicker parameter convergence. It was determined through several dynamic simulations that this is an application where more than one crossover point improves convergence speed. Instead of shuffling the individuals by selecting a crossover point for the entire individual, quicker convergence has been found by implementing a crossover on each substring. A crossover point for each substring keeps part of the genetic makeup intact, favoring worthy substrings. Implementation of the multi-point crossover routine allows each input variable to have its own randomly selected crossover point.

4.3.4 Mutation Sites for the TE Process

A typical mutation algorithm complements a single bit of the entire individual to make only minor differences between individuals. The mutation rate is usually small enough that mutation can be ignored. In the TE process, mutation does not change the overall performance if one of the less significant variables is mutated. To make it a significant secondary effect to avoid convergence to local minimums, each string of the individual under evaluation receives its own probability of mutation. It has been seen that the GA prefers to converge its values to find local minimums mainly because the initial population does not have enough random values. Here, the mutation rate is set slightly higher than normal to avoid total convergence to a single individual as well as decrease the convergence time.

4.3.5 Variations in the Simulation Procedure for the Genetic Algorithm

In addition to the three basic operators, the algorithm uses elitism to keep the two most fit individuals for the next generation. Elitism ensures a few of the most fit individuals of the next generation cannot be any worse than their predecessors. Although the goal of the GA is convergence of the population to a good set of individuals, the best individuals can be used in the actual plant setting as optimal values.

A significant observed problem in the simulation was that the initial generation would contain a large number of individuals with conditions that cause plant shutdown. Through the natural selection process these individuals are filtered out, but the problem exists that the sample space of good individuals is relatively small. A typical simulation using the procedure outlined above and the ranges displayed in Table 4-1 leads to more than half of the initial population being unreasonable due to plant shutdown conditions. The system does converge, but converges to sub-optimal solutions. To remedy this problem, a preprocessing step is added to the flow chart of Figure 4.2. The initial random individuals are generated as previously shown, but before the system enters the large loop, each individual’s fitness is evaluated based on 20 hours of plant operation. If the individual is within plant operating limits after 20 hours, it is a candidate for the first generation. If the individual causes a plant shutdown within the first 20 hours, a new random individual replaces the unfit individual. This procedure is repeated until the initial generation has individuals that at 20 hours of plant operation have not caused the plant to shut down.

After 20 hours most of the transients have been significantly reduced, but not enough to assess true steady-state values. Twenty hours was selected for the preprocessing evaluation time to still allow a few potentially unstable configurations into the population. It can be argued that with the preprocessing step the randomness of the GA is corrupted, but individuals that create unstable conditions contain at least some characteristics that are very detrimental to future generations. With the preprocessing step, the algorithm starts with a large set of potentially acceptable solutions instead of a well distributed set of individuals having only a few potentially useful solutions.

4.4 Results and Conclusions from GA Simulations

With the controllers in place the plant begins at its base-case values and adjusts to track the setpoints and agitation rate. The MATLAB [14] code that runs the GA initialization and simulation is displayed in Appendix A labeled "ga99_15_n.m". At the end of 40 hours of plant operation, the fitness of the input individual is evaluated based on the operating costs. If the system reaches plant shutdown conditions within this time, the cost per hour is set at a "large" value of $1000.00. To reduce the processing time and still keep a suitable sample size, each generation contains 50 individuals. To limit the effects of the end transients, the cost was calculated from an average of the measurements over the last hour. The measurement noise was removed to get an accurate cost and decrease the time for the transients to decay. Since the noise for each device can be modeled with a gaussian distributed signal with mean of zero, the average cost at steady-state with the measurement noise is approximately equivalent to the average cost at steady-state without the measurement noise. After 40 hours, most of the transients have decayed to contribute insignificant additional costs. The plant controller and simulation processing make up the bulk of the computational time, therefore they are implemented in an executable file compiled from FORTRAN code. The controller routine for the system is shown in Appendix A under "TEMAIN30.FOR". Even with the controller and plant simulation developed in FORTRAN code, each plant operation hour takes approximately 2.4 real-time seconds on a 233 MHz Pentium computer. As shown in Figure 4.10, the inputs of the controller and plant modules are the controller setpoints and agitation rate; the outputs of these modules are the plant operating costs.

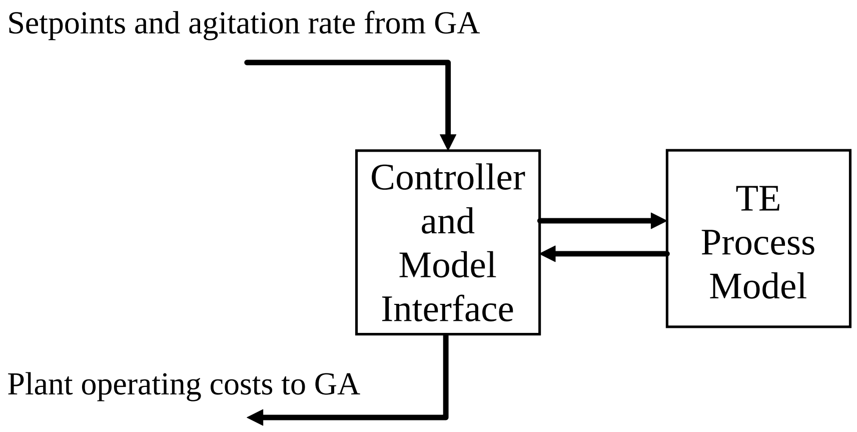

Setpoints and agitation rate from GA
Controller and Model Interface
TE Process Model
Plant operating costs to GA

Figure 4.10: Interaction between GA, controller, and plant model

Prior to the control and simulation loop, the GA written in MATLAB [14] code controls the inputs to the system. Each of the 50 individuals encode the setpoints and agitation rate. The values are pushed through the simulation loop giving an associated cost for each individual. Once all 50 individuals have been evaluated, the GA operators are performed. The termination criteria for the GA is to allow the system to continue for 100 generations before the simulation is stopped. After 100 generations optimal or near-optimal values will have been obtained. If the best solution found is still sub-optimal, the GA should be able to improve through the mutation operator. Since the probability of mutation is small, the GA in this case could take a very long time to increase its performance even a small amount. It has been seen that initial populations that contain a large number of fit individuals converge much quicker to a good set of sub-optimal values. This may justify the choice for a larger number of individuals for the population, but simulation time was still a major factor.

4.4.1 Genetic Algorithm Results with a Single Crossover Site

The first optimization attempt uses a single crossover point with typical probabilities of crossover and mutation of 0.7 and 0.001, respectively. A plot of the overall operating cost verses the generation number are displayed in Figure 4.11 for a typical simulation. The best cost is a non-increasing function due to the use of elitism in the GA. The plot for the average cost is somewhat deceiving because an individual that causes a plant shutdown has a very large economic cost in comparison to individuals that do not cause plant shutdown. When plotting the average cost for just the best 20 individuals, the result does not follow a recognizable pattern. Consequently, it is safe to say the GA using a single crossover point and these parameters does not perform well, although a local minimum cost was found from the initial random values.

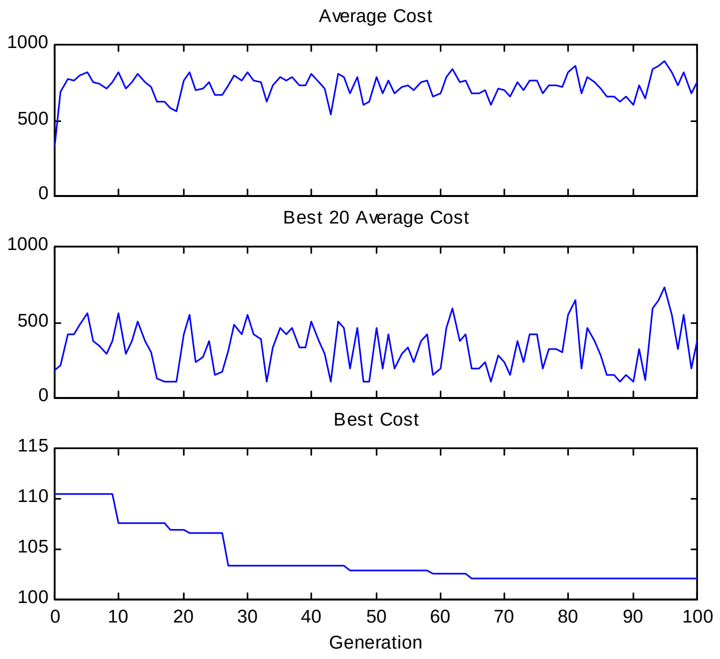


Figure 4.11: Genetic algorithm simulation results using a single crossover point

4.4.2 Genetic Algorithm Results with Two Crossover Sites

A plot for a particular run using the same configuration and initial random set as above, but with two crossover points, is shown in Figure 4.12. The convergence time is much shorter in obtaining a near optimal solution. By switching to a two-point crossover, a drastic difference can be noticed in the average cost. Whereas the simulation with a single crossover contained average costs that stayed random signals, the overall average and average of the best 20 individuals using two crossover points tend to improve their fitnesses. This data and other data collected from previous simulations justify trying a larger number of crossover points per individual. A balance in number of crossover points must be found because too many crossover points make the crossover operator a dominant random process.

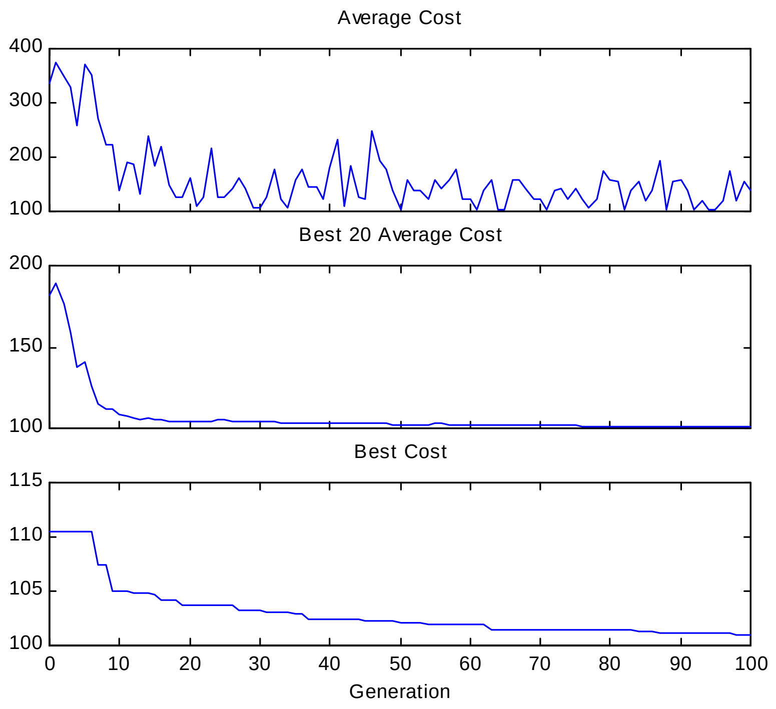


Figure 4.12: Genetic algorithm simulation results using two crossover points

4.4.3 Genetic Algorithm Results with a Single Crossover Point Per Variable

Instead of simply increasing the number of crossover sites, one random crossover point for each variable was implemented. This configuration will be referred to as the "base genetic algorithm". A typical simulation using the same parameters and data as above is shown in Figure 4.13. The overall system converges slightly quicker to near optimal values than the double crossover algorithm. The average fitness also increases (cost decreases) more rapidly than seen using a single or two-point crossover. By examining the overall average cost, it becomes apparent that when crossover and mutation occur, not as many unstable settings develop. The average cost of Figure 4.12 has sharp peaks mainly due to the creation of individuals that cause plant shutdown. The average cost of Figure 4.13 does not show this type of response; unstable input points do occur, although they occur less frequently. For this initial population, the most fit individual after 100 generations using a crossover point per variable was $100.91; the most fit individual using two crossover points was $101.36. For almost all cases tested the crossover point per setpoint resulted in smaller minimum costs and quicker converging times. Examining the best cost plot, it can be seen that at 100 generations, the best cost is still decreasing. Of all the simulations, the best cost found using a large number of generations is $99.68. Simulations were not performed using more than one crossover point per substring because each variable only has five bits of resolution.

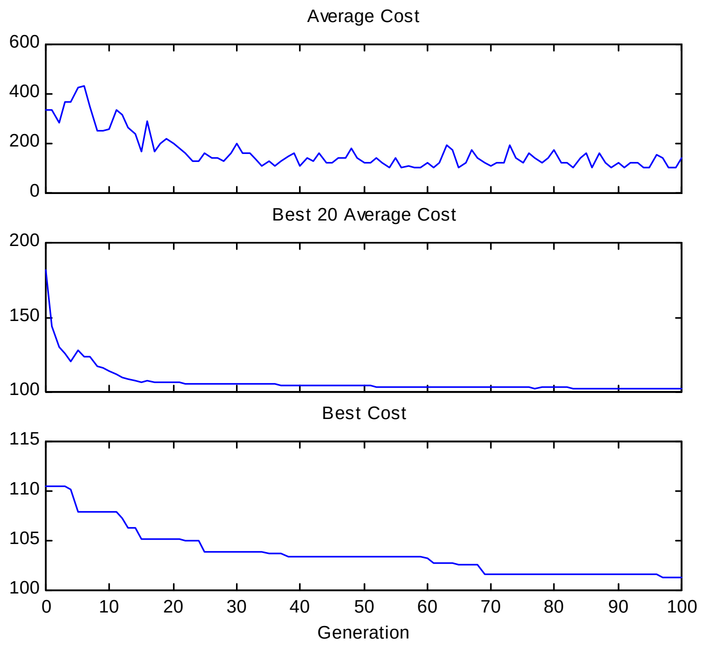


Figure 4.13: Genetic algorithm simulation results using one crossover point per variable

Selected outputs from the dynamic simulation of the system with the near-optimal setpoints of Figure 4.13 are shown in Figures 4.14 to 4.16. Figure 4.14 shows that some of the valves do saturate to the fully closed position. The A feed valve and purge valve saturate, and then return to steady-state values. To compensate for this, the PI controllers should have anti-windup mechanisms incorporated and possibly override controllers. Another option to handle valve saturation is to transfer the control loop to another variable. This was not done mainly to reduce the simulation time.

The other valve that saturates is the stripper steam valve which closes for all time. This tells the operator that steam is not necessary, therefore it is safe to assume that steam is unnecessary in the process. Closing the steam valve reduces the steam costs to $0.00.

Figures 4.15 and 4.16 display a few of the measurement outputs for the system. Although there are initial transients when moving from the initial settings to the new setpoints, the system operates within normal constraints. During the initial transition, the product quality (G/H ratio) does exhibit variation, but this is to be expected. Once the system has reached steady-state, the production rate and G/H ratio return to the defined operating conditions.

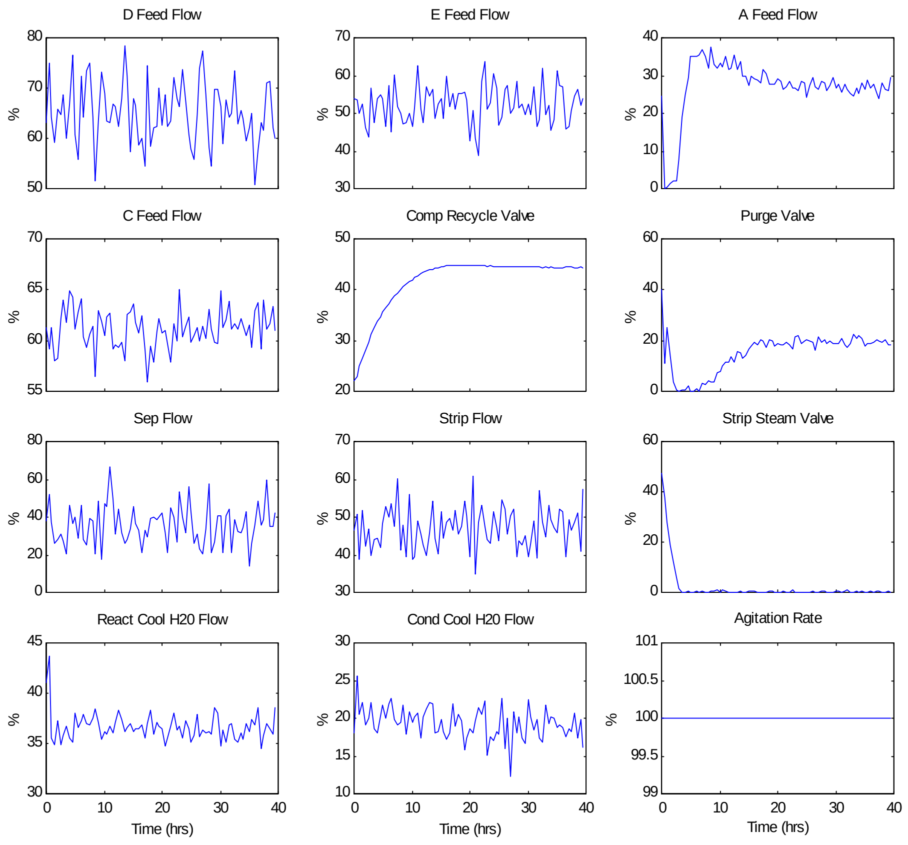


Figure 4.14: System inputs for near-optimal case

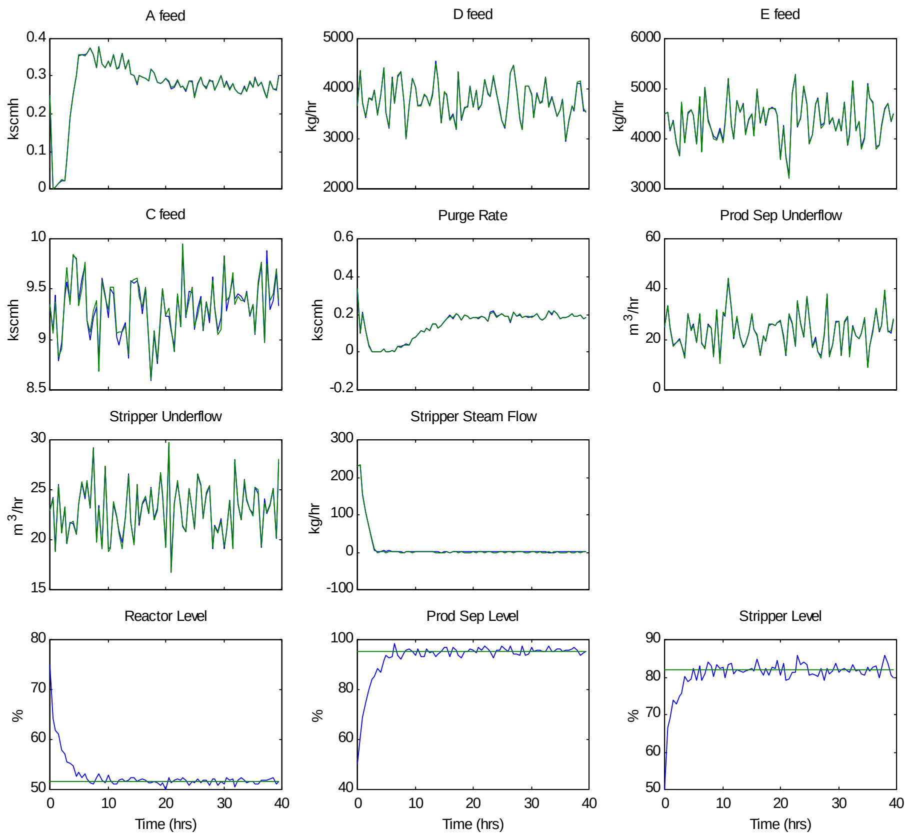


Figure 4.15: Selected outputs for near-optimal case

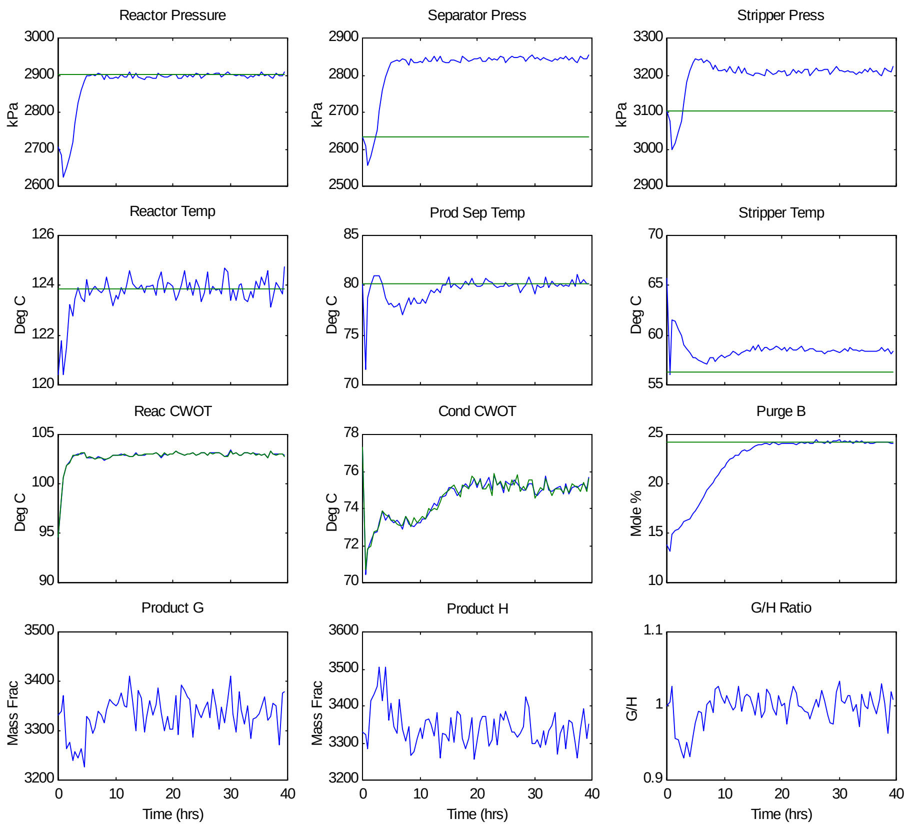


Figure 4.16: Selected outputs for near-optimal case

4.4.4 Conclusions from Genetic Algorithm Operation on the TE Process

The genetic algorithm found a set of individuals that contained characteristics for near-optimal operation. Instead of finding single points, there are a number of solutions that have their fitness values close to optimal. To allow fairness of algorithm comparisons, the SOLNP program [22] was also used to determine the minimum cost. Determining valid initial guesses that lead to the system converging to a minimum, was once again a difficult task. Due to the large number of nonlinearities and discontinuities from plant shutdown, the minimum cost found from the Jacobian based algorithm was $120.47.

Since the input vector for the minimization routine is much smaller than in Chapter 1, another optimization routine that uses a simplex search method was used to solve the minimum-cost problem. For low order systems the file “fmins.m” in MATLAB [14] usually finds minimum/maximum-cost solutions. The files “minplant.m” and “minyp.m” in Appendix A were designed to achieve the minimum-cost solution using the “fmins” routine. The program did find a minimum solution to the problem, but it found a local minimum. The minimum cost found using the base-case values as the initial guess is $109.37. This cost was achieved in approximately the same amount of time that a single run of the genetic algorithm requires to complete 100 generations.

The genetic algorithm is able to find a low cost solution, but from the plots above it should be noted that it takes a long time for the genetic algorithm to converge. There are several methods to reduce the time for convergence to the global minimum and convergence of the average to a set of fit individuals; one way is to adjust the probabilities of the mutation and crossover operators. The mutation and crossover parameters in the above simulations were fixed, based on previous evaluations of the system and similar systems. By adjusting these parameters with increasing generations, the convergence time of the GA can be decreased. In the next chapter, a fuzzy logic “supervisory” controller will be implemented to adapt these two parameters based on fitness values.

Chapter 5

Fuzzy Logic and Its Application to the Genetic Algorithm

In the previous chapter, a genetic algorithm determined a minimum operating cost of the controlled system. The system converged to a low-cost solution, but in general took a long time to accomplish this goal. The genetic algorithm probabilities of crossover and mutation can be optimized to achieve faster convergence to an optimal solution and avoid premature convergence to a local minimum solution.

Fuzzy logic is a control technique that incorporates a set of linguistic rules to adjust parameters. It was first designed to simplify responses for systems with nonlinearities and other complex mathematical properties. The algorithm uses interpolation to incorporate a linguistic set of rules into a control method. In this chapter, the basics of fuzzy logic will first be examined. Subsequently, a fuzzy logic controller will be designed to act in a supervisory role for determining the genetic algorithm parameters associated with the TE process. Comparisons of convergence between the GA and the fuzzy GA systems will be made to evaluate the advantages and disadvantages of each controlled system.

5.1 Introduction to Fuzzy Logic

In fuzzy logic, linguistic or qualitative models create a mathematical result similar to those developed from human deduction. The algorithm has found applications in systems that are difficult to control due to mathematical complexity, ambiguity, imprecision, or system uncertainties. The basis of a fuzzy logic system is to apply a set of fuzzy rules to an input, and output a result which interpolates over the applicable rules. The fuzzy procedure is shown below from [10].

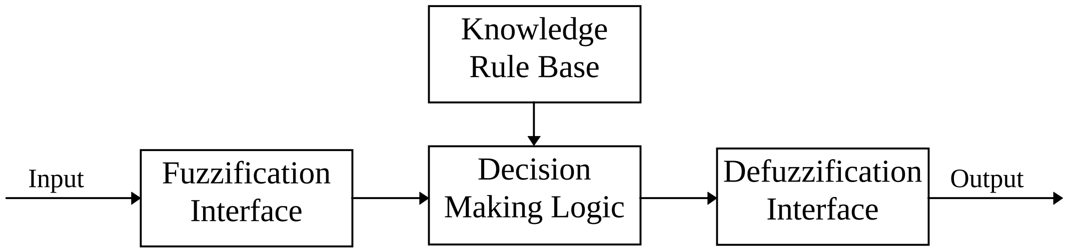


Figure 5.1: Fuzzy logic procedure

The rule base in Figure 5.1 is developed from expert knowledge of the system from physical or intuitive data. The system above is posed as a set of IF-THEN control rules of the form

$$\text{IF } x \text{ is } \mathbf{A} \text{ THEN } y \text{ is } \mathbf{B} \tag{5.1}$$

where *x* and *y* could be crisp (a number) input and output variables, and **A** and **B** are input and output linguistic variables, respectively. The input to the fuzzy logic system is labeled the antecedent or premise; the output is the consequent or conclusion [21]. When converting data to be processed by the fuzzy controller, the linguistic variables are encoded as fuzzy sets. Typically, the linguistic variables **A** and **B** have a form such as low, medium, high, or some other degree of measurement.

Fuzzy logic controllers use approximate reasoning to arrive at a decision for the system response. In approximate reasoning, there are two important fuzzy implication rules labeled the generalized modus ponens (GMP) and the generalized modus tollens (GNT). The implication rules use the fuzzy sets **A**, **A′**, **B**, and **B′**, where the primed sets are said to be "close to" the corresponding unprimed sets. From [10], the GMP is defined as:

$$\begin{array}{l} \text{premise 1: } x \text{ is } \mathbf{A}' \\ \underline{\text{premise 2: if } x \text{ is } \mathbf{A} \text{ then } y \text{ is } \mathbf{B}} \\ \text{consequence: } y \text{ is } \mathbf{B}' \end{array} \tag{5.2}$$

and the GNT is defined as:

$$\begin{array}{l} \text{premise 1: } y \text{ is } \mathbf{B}' \\ \underline{\text{premise 2: if } x \text{ is } \mathbf{A} \text{ then } y \text{ is } \mathbf{B}} \\ \text{consequence: } x \text{ is } \mathbf{A}' \end{array} \tag{5.3}$$

x and *y* are the input and output variables, respectively. Using the above two inference rules, most of the intuitive rules associated with fuzzy logic can be defined. Important factors in the development of the GMP and GNT are the creation of the fuzzy membership sets, developing the fuzzy set operations, and applying the fuzzy rules.

5.1.1 Fuzzy Membership Sets

An important fuzzy logic concept is that of fuzzy membership sets, which were developed from Zadeh's seminal paper on fuzzy sets [25]. Instead of defining an element as part of a crisp set, where the element is either has membership or not, the element can be defined as having a certain "degree of membership". Using a crisp (classical) set **A**, the normal binary issue of membership is

$$X_{\mathrm{A}}(x) = \begin{cases} 1 & x \in \mathbf{A} \\ 0 & x \notin \mathbf{A} \end{cases} \tag{5.4}$$

where $X_{\mathrm{A}}(x)$ is the entire region of values called the universe of discourse. The crisp set can be viewed graphically as shown below for the one-dimensional case for *x* an element of $X_{\mathrm{A}}(x)$.

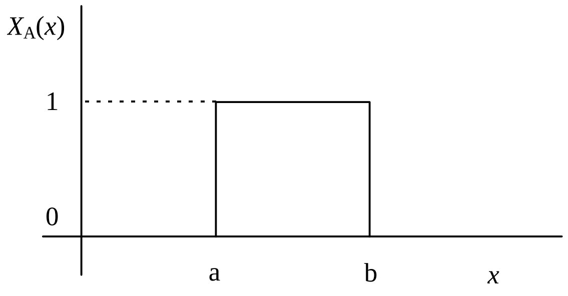


Figure 5.2: Crisp membership function

The problem with crisp sets, as above, is that for a small perturbation ε, where x=a–ε, the membership has a value of zero. For the same small perturbation, x=a+ε maps to one. By contrast, a fuzzy set contains elements that have varying degrees of membership within the set. Instead of having crisp boundaries, Zadeh in [25] proposed creating "fuzzy sets" as shown in Figure 5.3. The fuzzy membership function $\mu_{\mathrm{A}}(x)$ gives relative "degrees of membership" of **A**.

In the fuzzy set shown, the points x=a–ε and x=a+ε almost have identical membership values.

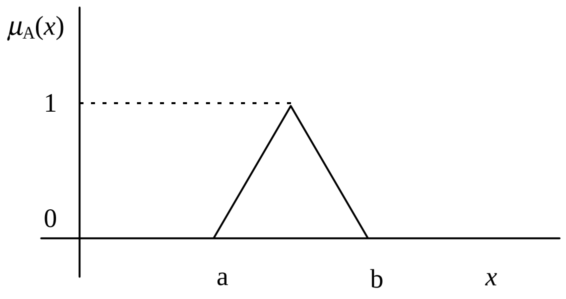


Figure 5.3: Triangular shaped fuzzy set

Usually the fuzzy sets contain a maximum membership $\mu_A(x)$ of one and a minimum of zero to be consistent with the limits of crisp sets. A triangular membership function is shown in Figure 5.3, although several shapes have been proposed by researchers including piecewise linear, Gaussian, and trapezoids.

5.1.2 Decision Making Logic Connectives

After Zadeh's introduction of fuzzy logic in [26], several researchers have proposed operations to calculate and enhance fuzzy logic values. A fuzzy system typically has two or more inputs, and produces one or more output. There are various methods for processing data, but the three main categories of fuzzy set operations are fuzzy conjunction, fuzzy disjunction, and fuzzy implication. Fuzzy conjunction is equivalent to the intersection of sets and is used when there is more than one input or output that require an "and" connective. An example of a linguistic rule that includes conjunction is

$$\text{IF } x_1 \text{ is } \mathbf{A_1} \text{ AND } x_2 \text{ is } \mathbf{A_2} \text{ THEN } y \text{ is } \mathbf{B} \tag{5.5}$$

The conjunction operator is designated by a '$\wedge$' making the notation from Equation 5.5

$$\mu_{A_1}(x_1) \wedge \mu_{A_2}(x_2) = \mu_B(y) \tag{5.6}$$

Common mathematical methods of applying the conjunction operator use the "minimum"

$$\mu_B(y) = \min\left\{\mu_{A_1}(x_1),\ \mu_{A_2}(x_2)\right\} \tag{5.7}$$

and the "product"

$$\mu_B(y) = \mu_{A_1}(x_1) \cdot \mu_{A_2}(x_2) \tag{5.8}$$

The disjunction operator is characterized by the '$\vee$' symbol which is used when a logical "or" is required. An example where the disjunction operator would be used is

$$\text{IF } x_1 \text{ is } \mathbf{A_1} \text{ OR } x_2 \text{ is } \mathbf{A_2} \text{ THEN } y \text{ is } \mathbf{B} \tag{5.9}$$

or equivalently,

$$\mu_{\mathrm{B}}(y) = \mu_{\mathrm{A}_1}(x_1) \vee \mu_{\mathrm{A}_2}(x_2) \tag{5.10}$$

The most common decision strategies for disjunction are the "maximum"

$$\mu_{\mathrm{B}}(y) = \max\left\{\mu_{\mathrm{A}_1}(x_1),\ \mu_{\mathrm{A}_2}(x_2)\right\} \tag{5.11}$$

and the "algebraic sum"

$$\mu_{\mathrm{B}}(y) = \mu_{\mathrm{A}_1}(x_1) + \mu_{\mathrm{A}_2}(x_2) \tag{5.12}$$

The disjunction operator is used when the system requires a union of sets.

The inputs and outputs can be mapped into a rule table. The rule table shows all of the possible configurations within the universe of discourse. A typical rule table is shown below for a two input, single output rule base. Two of the rules that can be inferred from the table are

$$\text{IF } x_1 \text{ is } \mathbf{A_{11}} \text{ AND } x_2 \text{ is } \mathbf{A_{12}} \text{ THEN } y \text{ is } \mathbf{B_1} \tag{5.13}$$

$$\text{IF } x_1 \text{ is } \mathbf{A_{31}} \text{ AND } x_2 \text{ is } \mathbf{A_{22}} \text{ THEN } y \text{ is } \mathbf{B_6} \tag{5.14}$$

Table 5-1: Typical Rule Table for Linguistic Variables

		x_1		
		A_{11}	**A_{21}**	**A_{31}**
x_2	**A_{12}**	**B_1**	**B_2**	**B_3**
	A_{22}	**B_4**	**B_5**	**B_6**
	A_{32}	**B_7**	**B_8**	**B_9**

Since most rule-based systems involve more than one rule as shown in Table 5-1, the process of obtaining the overall consequent requires aggregation of the rules. Usually the output rules are connected by "and" connectives [21]. This conjunctive system of rules and its corresponding output for *r* rules is typically determined by combining (aggregating) the individual outputs.

$$\mu_{\mathrm{B}}(y) = \min\left\{\mu_{\mathrm{B}_1}(y),\ \mu_{\mathrm{B}_2}(y), \ldots,\ \mu_{\mathrm{B}_r}(y)\right\} \tag{5.15}$$

Occasionally, the output will be a disjunctive set, where only one of the *r* rules of a set must be satisfied. When this occurs the final output membership function is typically determined by

$$\mu_{\mathrm{B}}(y) = \max\left\{\mu_{\mathrm{B}_1}(y),\ \mu_{\mathrm{B}_2}(y), \ldots,\ \mu_{\mathrm{B}_r}(y)\right\} \tag{5.16}$$

Several implication rules have been developed using fuzzy relations from the fundamental conditions of Equation 5.1. The number and type of rules to implement is problem-dependent although application of the rules is similar in all cases.

5.1.3 Fuzzy Relation Operator

The fuzzy relation depicts the strength of association between the input fuzzy set to the output fuzzy set. The basic fuzzy relation matrix **R** is determined by taking the cartesian product of an input and output set. There are several methods of determining the cartesian product, but the most common are the Mandami rule of fuzzy implication defined by

$$\mu_{\mathrm{R}}(x, y) = \min\left\{\mu_{\mathrm{A}}(x),\ \mu_{\mathrm{B}}(y)\right\} \tag{5.17}$$

and the product rule defined by

$$\mu_{\mathrm{R}}(x, y) = \mu_{\mathrm{A}}(x) \cdot \mu_{\mathrm{B}}(y) \tag{5.18}$$

These two rules can be applied directly if the system is in the form of Equation 5.1. Given input fuzzy values, $\mu_{\mathrm{A}}(x)$, and the rule base from $\mu_{\mathrm{R}}(x,y)$, the output set, $\mu_{\mathrm{B}}(y)$, can be calculated using the sup-star composition for binary fuzzy relations

$$\mathbf{B} = \mathbf{A} \circ \mathbf{R} \tag{5.19}$$

The forms to evaluate the composition expression from Equations 5.17 and 5.18 are the Mandami (max-min) composition of

$$\mu_{\mathrm{B}}(y) = \sup_{x \in X}\ \min\left\{\mu_{\mathrm{A}}(x), \mu_{\mathrm{R}}(x, y)\right\} \tag{5.20}$$

and the max-product method of composition

$$\mu_{\mathrm{B}}(y) = \sup_{x \in X} \; \mu_{\mathrm{A}}(x) \cdot \mu_{\mathrm{R}}(x, y) \tag{5.21}$$

The relational matrix **R** can also be viewed as a graphical inference. An example from [21] is given below in Figures 5.4 and 5.5 showing the results using the Mandami and max-product implication techniques. The two rules of this system are

Rule 1: IF x_1 is $\mathbf{A_{11}}$ AND x_2 is $\mathbf{A_{12}}$ THEN y is $\mathbf{B_1}$ (5.22)

Rule 2: IF x_1 is $\mathbf{A_{21}}$ AND x_2 is $\mathbf{A_{22}}$ THEN y is $\mathbf{B_2}$ (5.23)

In Figures 5.4 and 5.5, $\mu_{\mathrm{R}}(x,y)$ converts the input fuzzy sets to the output fuzzy sets. Crisp Input 1 and Input 2 determine which fuzzy input set is dominant and to what degree. For example, Input 1 is the dominant input for the first rule, therefore $\mathbf{A_{11}}$ controls $\mathbf{B_1}$. The max-product conclusions of Figure 5.5 are displayed as scaled triangles, although would not be implemented using this method.

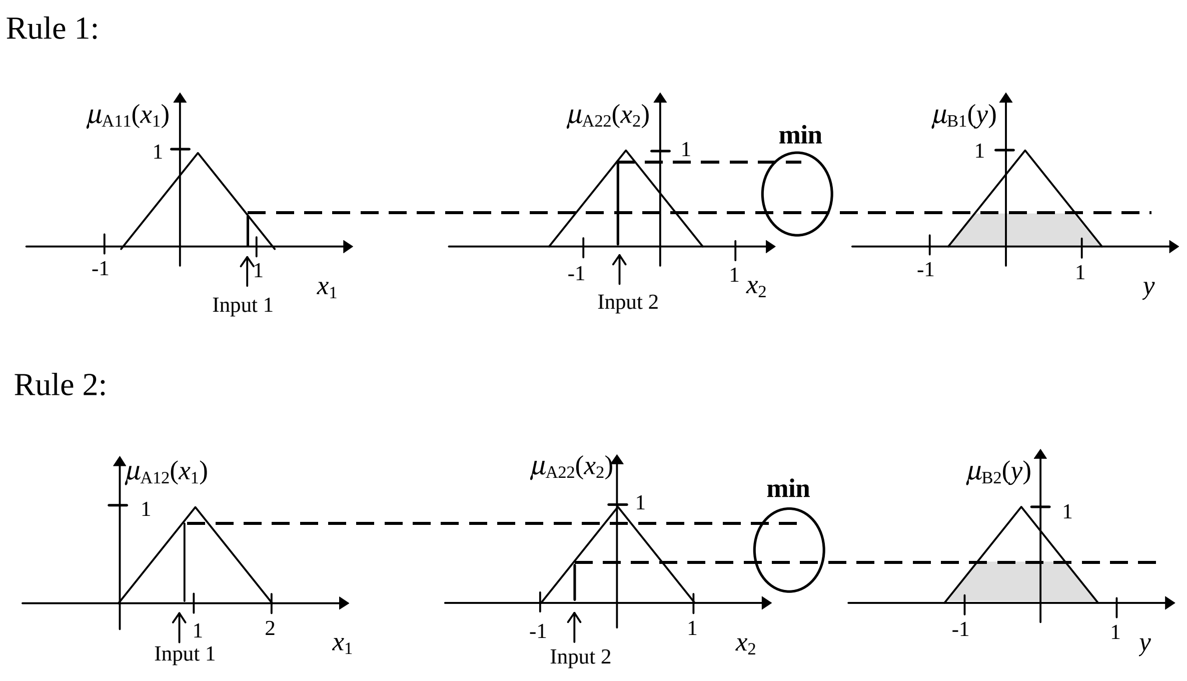


Figure 5.4: Graphical interpretation of Mandami inference

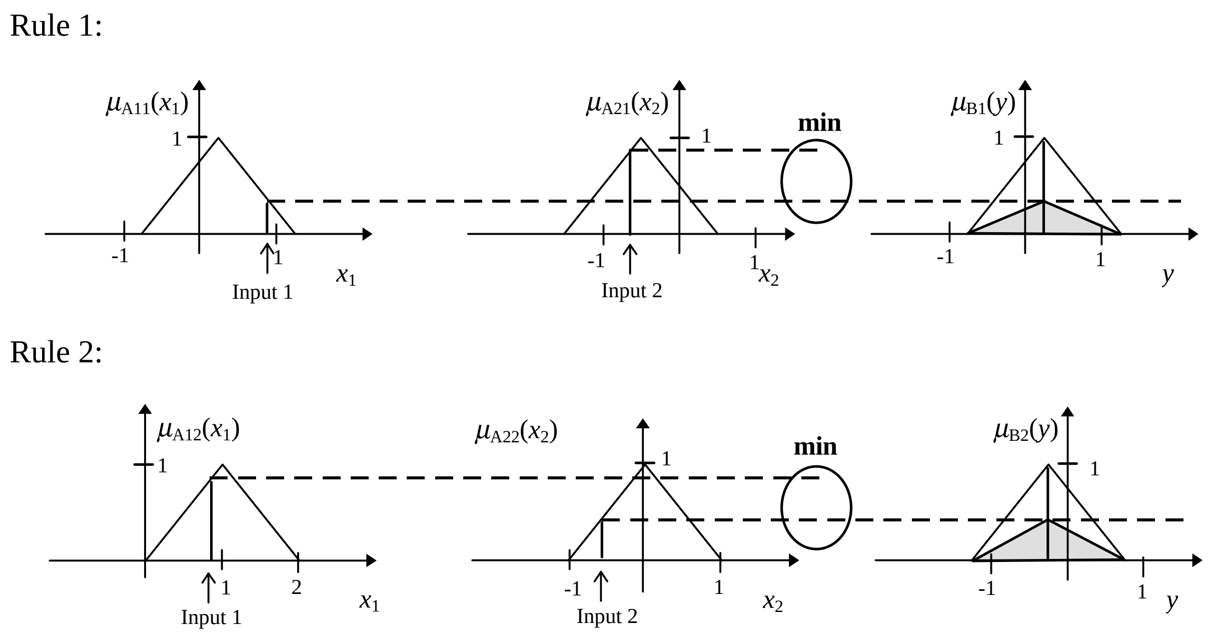


Figure 5.5: Graphical interpretation of max-product rules

The final fuzzy membership function aggregates the two consequences into one membership function. The resulting output membership functions for the above example are displayed below after applying the supremum functions of Equations 5.20 and 5.21.

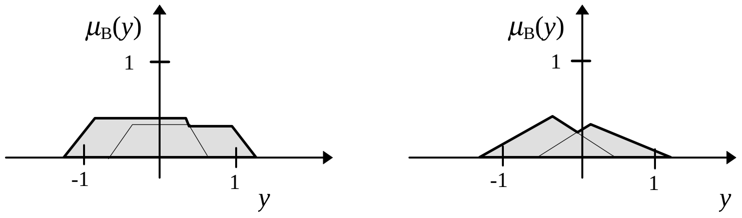


Figure 5.6: Output membership functions for Mandami and max-product inferences

5.1.4 Defuzzification Techniques

Once the fuzzy inference rules are applied to the fuzzy input set, a fuzzy output set can be determined. Typically, the designer requires a crisp output from the fuzzy logic system for further processing. Several defuzzification methods have been developed to convert the fuzzy output membership functions into crisp variables. The most commonly used strategies for most applications are the max criterion, mean of the maxim, and center of area [10]. Of these three, the center of area (COA) method is used most frequently, which is also referred to as the center of gravity, or centroid method. The COA method determines the centroid of the fuzzy set by the equation

$$y^* = \frac{\int \mu_B(y) \cdot y \, dy}{\int \mu_B(y) \, dy} \tag{5.24}$$

where $\mu_B(y)$ is the output membership function. From the Mandami inference output set of Figure 5.6, the crisp output using the COA method leads to:

$$
\begin{aligned}
y^* &= \frac{\int \mu_B(y)\cdot y\,dy}{\int \mu_B(y)\,dy} \\
&= \frac{\left\{\int_{-1.25}^{-0.85}(y+1.25)ydy+\int_{-0.85}^{0.35}0.4ydy+\int_{0.35}^{0.45}(0.75-y)ydy+\int_{0.45}^{0.95}0.3ydy+\int_{0.95}^{1.25}(1.25-y)ydy\right\}}{\left\{\int_{-1.25}^{-0.85}(y+1.25)dy+\int_{-0.85}^{0.35}0.4dy+\int_{0.35}^{0.45}(0.75-y)dy+\int_{0.45}^{0.95}0.3dy+\int_{0.95}^{1.25}(1.25-y)dy\right\}} \\
&= \frac{-0.0325}{0.7900} = -0.0411
\end{aligned}
\tag{5.25}
$$

Evaluating the integral of Equation 5.25 is computationally intensive for even two rules, therefore several strategies to approximate the centroid have cultivated. A method frequently implemented is the weighted (center) average. A crisp output from the weighted average method applies the equation

$$
y^* = \frac{\sum_{i=1}^{r}\mu_{Bi}(\bar{y}_i)\cdot\bar{y}_i}{\sum_{i=1}^{r}\mu_{Bi}(\bar{y}_i)} \tag{5.26}
$$

where $\bar{y}_i$ is the mean of each membership function. The one major drawback of the weighted average method is the output fuzzy sets must be symmetrical. From the Mandami inference output set of Figure 5.6, the weighted average is calculated by

$$
y^* = \frac{h_1 y_1 + h_2 y_2}{h_1 + h_2} \tag{5.27}
$$

where h_1 and h_2 are the heights of each of the trapezoids, and y_1 and y_2 are the centers of the fuzzy output sets $\mathbf{B_1}$ and $\mathbf{B_2}$. For the example above, the crisp output using Equation 5.27 is

$$
y^* = \frac{(0.3)(0.25)+(0.4)(-0.25)}{0.3+0.4} = -0.0357 \tag{5.28}
$$

Comparing the result of Equation 5.28 to that of 5.25 verifies the two methods provide similar results for this example. In fact, for most applications, the weighted average method provides a close approximation of the centroid.

5.2 Fuzzy Logic Applied to the Tennessee Eastman Problem

Now that the basics of fuzzy logic have been established, the question remains of how to exploit the benefits of fuzzy logic in the TE process problem. In genetic algorithm research, there have been studies such as those found in [4], [8], and [23] to decrease the convergence time of the GA, and still ensure the system does not converge and remain at local minimums. Determining optimal genetic algorithm parameters is a complex, problem-dependent task. Recently, Azam and VanLandingham [1] proposed a method where the probabilities of crossover and mutation are adjusted by a supervisory fuzzy controller. A fuzzy supervisory controller allows the parameters to be adjusted based on a set of approximating rules. The results from [1] show quicker convergence to lower minimums for the five problems posed by De Jong [5]. A similar procedure to that of [1] will be applied here to aid in convergence to the minimum-cost solution. In off-line simulations, such as the TE process problem, the largest concern is to avoid remaining at a local minimum. Reaching a minimum within a fixed amount of time is important, but convergence to the global or near-global minimum should have the highest precedence.

5.2.1 Rule Base for the Fuzzy Logic Controller

A block diagram of the system setup from [1] is shown in Figure 5.7. The fuzzy logic controller adjusts control parameters of the GA based on performance measurements and/or the existing control parameters of the GA. Two important control parameters of the GA are the probabilities of crossover and mutation, therefore these two variables make logical sense to be the outputs from the fuzzy logic controller. There are several possible input parameters of the fuzzy logic controller including the past and current crossover and mutation probabilities, the generation gap, the diversity of the population, the average fitness of the population, and the best fitness of the population.

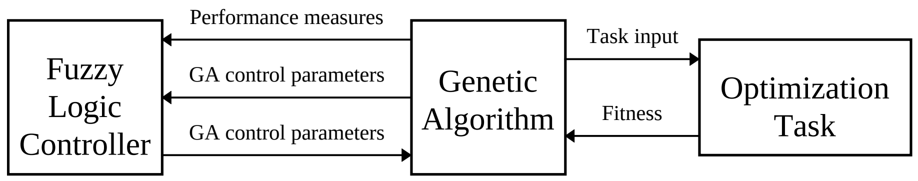


Figure 5.7: Adaptive fuzzy controlled genetic algorithm system

After selecting the inputs and outputs of the controller, the rules and associated fuzzy memberships must be generated. For this study, two fuzzy logic controllers are developed — one for each of the crossover and mutation operators. The inputs selected for the crossover fuzzy logic controller are the average fitness of the population, $\overline{f}$, the most fit individual, f_{max}, and the fitness of the parents, f_1 and f_2. The parent with the most fit value is assigned to the variable f'. The intuitive logic built into the controller is that if the parent associated with fitness f' is not "very fit" in comparison to the most fit individual, f_{max}, the parents should crossover their genetic makeup to produce more fit children. If one of the parents is already fit, the system should avoid crossover to keep the genetic makeups intact. Based on these rules, the fuzzy rule base for crossover using three input membership sets and five output membership sets is displayed in Table 5-2. The linguistic variables are **L** for low, **ML** for medium-low, **M** for medium, etc..

Table 5-2: Fuzzy Rule Base for Crossover

		$\overline{f}$ / f_{max}		
		L	**M**	**H**
$\frac{f'}{f_{max}}$	**L**	**M**	**MH**	**H**
	M	**ML**	**M**	**MH**
	H	**L**	**ML**	**M**

A similar fuzzy rule base can be established for the mutation operator. If the selected individual from the mating pool is considered "fit" in comparison to the most fit individual of the species, the new individual should have a low probability of mutation to protect the "fit" genes. If the selected individual is "unfit", the mutation rate should increase the chance that the new individual will mutate. The rule base for the mutation operator is displayed in Table 5-3.

Table 5-3: Fuzzy Rule Base for Mutation

		$\overline{f}\,/\,f_{max}$		
		L	**M**	**H**
$\dfrac{f}{f_{max}}$	**L**	**M**	**MH**	**H**
	M	**ML**	**M**	**MH**
	H	**L**	**ML**	**M**

Once this rule base has been established, the quantitative membership functions associated with **L**, **ML**, **M**, **MH**, and **H** must be designated.

During the testing phase, it was determined that calculating the average based on the best 20 individuals instead of the entire population provides better results. The 20 best individuals is a more realistic measure of the fitness of the entire population because an individual that causes plant shutdown largely affects the average.

A performance enhancement of the fuzzy GA is to use scaling to change the degree of fitness of the population. Scaling enhances the pressure of selection in the later stages of the search, thereby reducing the possibility of remaining at a local minimum cost. In the fuzzy controller of the TE process, scaling is used to account for the inherent costs of operating the plant. For example, if the current maximum fitness has an associated cost of \$101 and the average fitness is \$102, the differences seen by the fuzzy controller are minimal. But, if scaling is incorporated into the controller by creating an offset of \$100 from the input values, the two fitnesses appear drastically different. In the crossover and mutation fuzzy controllers of the TE process, an offset of \$90 is added, which is a value well below the expected minimum operating costs of the base-case mode. The new crisp input for the average changes from

$$x_1 = \frac{\overline{f}}{f_{max}} \tag{5.29}$$

to

$$x_1 = \frac{\overline{f} - 90}{f_{max} - 90} \tag{5.30}$$

With scaling included, x_1 still varies between 0 and 1. It has been empirically determined that scaling in the TE process decreases the potential for converging and remaining at local minima. The scaling technique outlined above is also applied to the inputs generated from f and f'.

5.2.2 Implementing the Fuzzy Logic Rules

Fuzzy membership functions must be assigned to each of the linguistic variable of Tables 5-2 and 5-3. The rule base and its associated membership functions are developed from expert knowledge of the system gained from several simulations. It has been shown in [7] and [4] that different applications involving genetic algorithms converge with almost any set of parameter values, but the optimal parameters can vary drastically depending on the application.

The MATLAB code that implements the fuzzy controllers are displayed in Appendix A as the files "fuzzpc.m" and "fuzzpm.m". Each of the functions are performed within the genetic algorithm loop. The output fuzzy sets are determined using the Mandami equation (max-min rules). The membership functions are symmetrical and triangular shaped, therefore the weighted average defuzzification method gives apt results. Originally, the problem was posed where crossover is determined using a set of linearly distributed membership functions from 0 to 100%, and the mutation operator uses membership functions that vary linearly from 0 to 0.1%. The system responses with these parameters and those from the base case genetic algorithm are shown in Figure 5.8 for typical simulations. At 100 generations, the fuzzy genetic system had a minimum cost of \$101.94, while the base genetic algorithm had a minimum cost of \$101.33. The main reason the fuzzy GA system converged and stayed at a local minimum is the mutation operator is small enough that the system is incapable of jumping to other local minimums in a reasonable amount of time. Examining the population at the final generation revealed that there was little diversity in the population, essentially due to the small probability of crossover. The overall results of the above fuzzy GA system have been observed through this and similar simulations to be approximately equivalent (if not worse) than those found using the base genetic algorithm with fixed parameters.

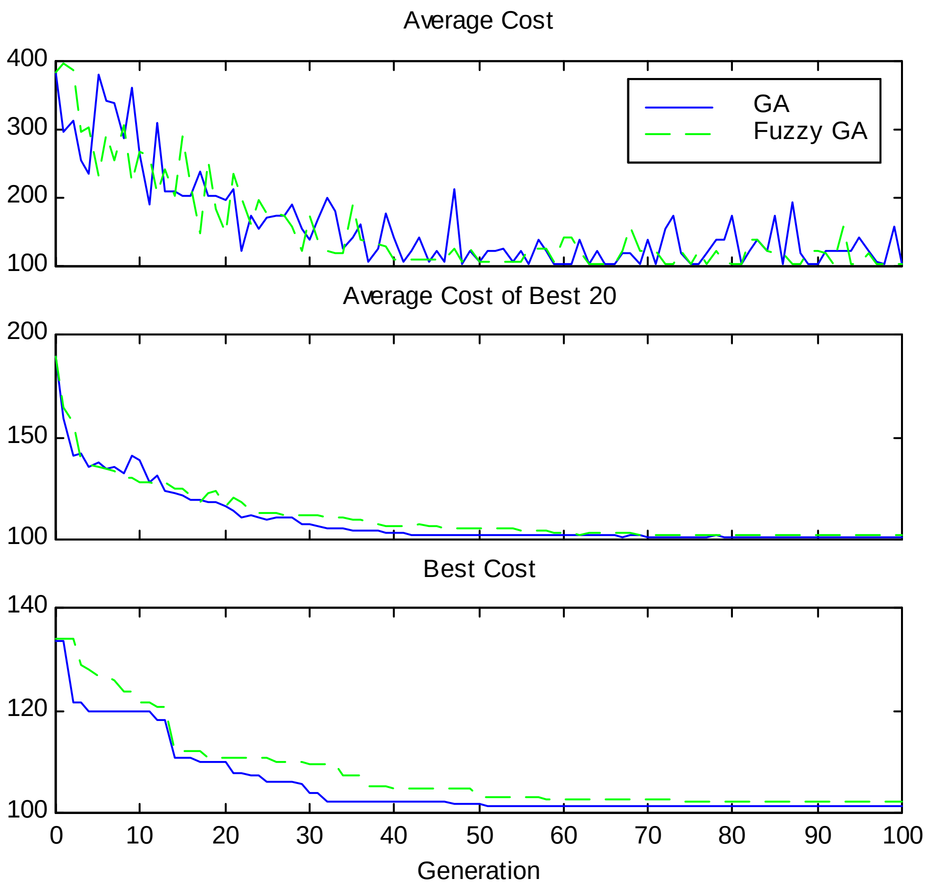


Figure 5.8: Results from simple fuzzy controller

The fuzzy controllers were reposed to incorporate the knowledge gained from the above simulation. After several simulations to provide more expert knowledge, the final input and output fuzzy sets are displayed in Figures 5.9 and 5.10. The input sets have symmetrical distributions, but the output sets reflect the notion that an "unfit" individual should have a much higher probability of crossover and mutation than a "near fit" individual.

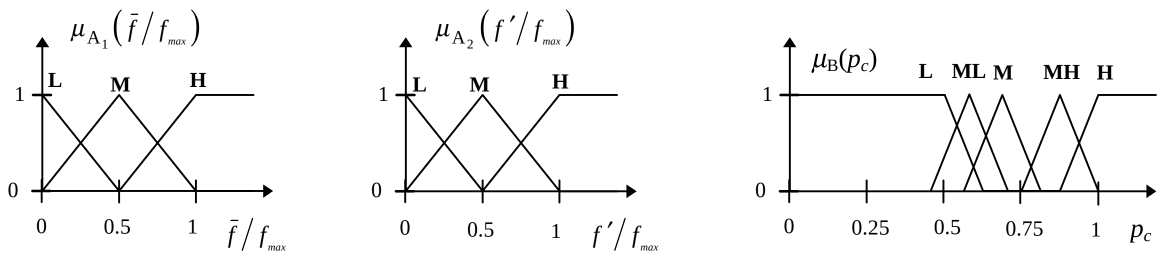


Figure 5.9: Input and output fuzzy sets for crossover operator

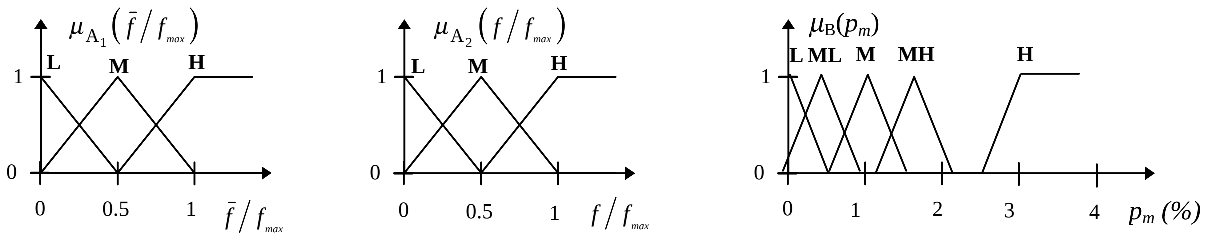


Figure 5.10: Input and output fuzzy sets for mutation operator

5.3 Results and Conclusions

Although the genetic algorithm produced results for the optimal cost problem, its convergence parameters were selected somewhat arbitrarily, and therefore could lead to sub-optimal results. A fuzzy genetic algorithm was presented that results in better performance than the base genetic algorithm. The disadvantage of using the fuzzy supervisory controller is the additional processing time required for implementation; but in this case, the processing time is negligible in comparison to the time requirement of simulating the process. Although tuning of the fuzzy rules improves the fuzzy controller performance, the fuzzy sets can be determined by a set of heuristic rules. For all initial populations tested, the fuzzy GA using the above rules performed better than the base genetic algorithm at converging to a minimum-cost solution. The minimum cost determined from several simulations was $98.19, which is only five cents more than the cost found in Chapter 2 without a controller. It is believed that this is the minimum attainable cost without more resolution, but this cannot be guaranteed. Several simulations were completed to verify this solution is the minimum, but for all the simulations involving the fuzzy GA, the minimum cost at 100 hours was within 3% of this “optimal” cost.

5.3.1 Comparing the Fuzzy Logic Controlled GA System to the Basic GA System

The plots of the system response using the basic GA and fuzzy adaptive GA for one particular simulation are displayed in Figure 5.11. Although only one simulation using the same initial population is displayed, the results of Figure 5.11 are representative of the results from other simulations. Typically, the best cost of the fuzzy GA converges at approximately the same rate as the base GA, but the minimum cost of the fuzzy GA at 100 generations is always smaller for several configurations tested. In both the GA and fuzzy GA, the overall average, average of the best 20 individuals, and the best cost, decrease in time at approximately the same rate. Although not displayed well in Figure 5.11, the fuzzy genetic algorithm also has superior performance for converging the population to a fit set of individuals.

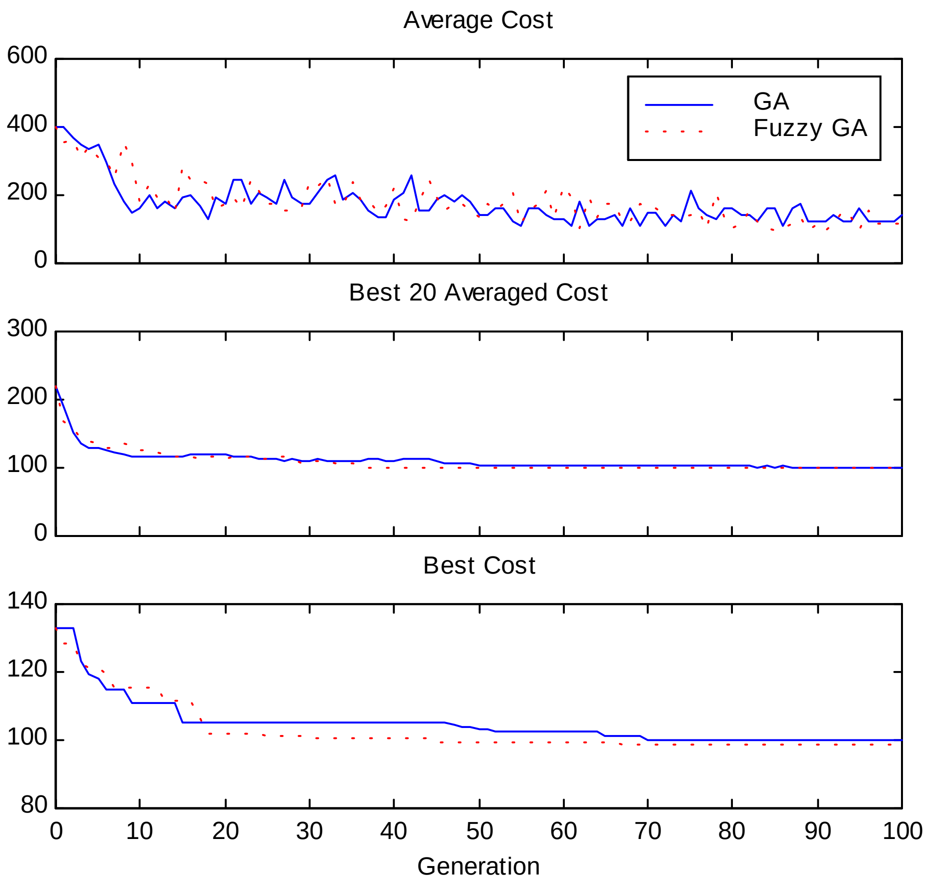


Figure 5.11: Genetic algorithm convergence to the minimum-cost solution

5.3.2 Additional Fuzzy Parameters

As mentioned in Section 5.2.1, there are other input variables that can be manipulated to improve the genetic algorithm performance. One important variable is the population diversity. After several generations, genetic algorithms tend to converge the majority of their population to one particular individual. A sufficient rule base for a fuzzy controller using the population diversity, average fitness, and most fit individual is displayed in Table 5-4. One of the controlling rules is that if the diversity is “low”, crossover and mutation should occur more frequently than normal to guarantee only a local minimum was not found. The other controlling rule is that if the average fitness is “far” from the maximum fitness, crossover and mutation should occur to attempt to bring the average closer to the maximum fitness.

Table 5-4: Rule Base Using Population Diversity

		$\overline{f} / f_{max}$		
		L	**M**	**H**
div	**L**	**H**	**MH**	**M**
	M	**MH**	**M**	**ML**
	H	**M**	**ML**	**L**

Bäck in [2] defines the bias of a population by examining the bits of each individual. For a population *a*, the diversity can be determined by the bias

$$b\big(P(t)\big) = \frac{1}{l\cdot\mu}\sum_{j=1}^{\mu}\max\left\{\sum_{i=1}^{\mu}\left(1-a_{i,j}\right), \sum_{i=1}^{\mu}a_{i,j}\right\} \tag{5.31}$$

where l is the individual length and μ is the size of the population. The result *b(P(t))* varies between 0.5 and 1.0. Values close to 1.0 show strong correlation between individuals, and very little population diversity; a value of *b(P(t))* close to 0.5 shows a large amount of population diversity. The rule base of Table 5-4 was tested on the TE controlled process, but the fuzzy logic controllers did not perform as well as those found above. The main problem was the average fluctuates too much in the first few generations, but more research and better tuned rules may develop better results, although was not pursued here.

Chapter 6

Results and Conclusions

In the base-case mode, the supervised genetic algorithm was capable of determining the settings for a near-minimum-cost solution to the TE process. With a decentralized controller in place, the algorithm converged to a set of plant operating conditions that forces the system to be stable and provide a low operating cost. Besides the results found here, there are improvements that make the search algorithm estimate the minimum cost in a shorter amount of time, and with less chance of remaining at a local minimum. Although the global minimum may not have been completely reached, the search method determined settings that other search methods were unable to find. Before discussing conclusions and possible future work, the current plant settings should be analyzed to determine the advantages and disadvantages of the controlled system, including parameter sensitivities for this solution.

6.1 Performance Evaluation of the Controlled System

The ten degrees of freedom associated with a cost of $98.19 per hour are shown in Table 6-1. The cost was determined from an average of the measurements over the 40th hour. The settings are for the system operating in the base-case mode with no active disturbances or measurement noise. Stability of the controller still must be examined when these settings are implemented and a disturbance is in progress. It is expected that the system will be less robust than using the base-case settings because some of the devices are operating close to their limits, however the system still should avoid plant shutdown.

Table 6-1: Optimal Setpoints and Agitation Rate for the Decentralized Controlled System

Manipulated Variable	Optimal Value
Recycle Flow Setpoint	34.6
Reactor Pressure Setpoint	2895
Reactor Level Setpoint	51.6
Reactor Temperature Setpoint	122.9
Separator Level Setpoint	95.5
Stripper Level Setpoint	95.5
Stripper Temperature Setpoint	59.5
Compressor Power Setpoint	266
Component B in Purge Setpoint	24.2
Agitation Rate	96.8

6.1.1 Parameters Adjusted to Conform to the Control Objectives

Several of the adjustable variables approach their minimum or maximum bounds from the genetic algorithm simulation. As reactor pressure increases, the operating costs always decreases. As a result, the reactor pressure setpoint should be fixed at its maximum allowable value. Similarly, the reactor level decreases close to its minimum value. As the liquid level decreases in the reactor, the gas residence time increases; therefore, reducing the reactor level to its minimum value has large economic costs. The separator and stripper levels approach their upper limits, but actually have little impact on costs. The corresponding level setpoints are not fixed to allow the system to respond to disturbances and transients. Operating outside of the normal operating limits was not tested as the severity of a plant shutdown or operating close to a plant shutdown condition is unknown; however, causing the plant to shut down is very detrimental for safety and environmental reasons. Logical reasoning reveals that operating slightly outside the normal operating limits should not cause significant alarm. For example, the low normal limit of the reactor level is 50%, but the reactor level needs only be an actual concern if the tank is close to running dry.

The dynamic responses of the system with the settings of Table 6-1 are shown in Figures 6.1 to 6.3. Figure 6.1 shows that the compressor recycle, purge and stripper steam valves saturate to their lower bounds. Since the compressor recycle and steam valves remain at their lower bounds, instead of placing controllers on these inputs, the valves should be completely closed. Closing the recycle valve maximizes the recycle rate, which reduces the compressor and purge costs. Closing the steam valve reduces both the steam costs and the stripper temperature, thereby creating more liquid product. The purge valve saturates and then increases back to a steady-state value, therefore the control loop may have to be modified. Some researchers advocate adding overrides when valve saturation occurs, while others neglect it if saturation occurs infrequently. Saturation of the purge valve in this case only occurs when changing from the base settings to the “optimal” settings. If valve saturation is a problem, the change to the new setpoint can be done gradually instead of a quick step change.

The agitation rate shown in Figure 6.1 approaches its upper bound limit. Although not determined from the steady-state analysis of Chapter 2, Ricker in [19] suggested that maximizing the agitation rate reduces the need for reactor coolant flow. The reasoning is that a faster agitation rate converts the gases to liquids more efficiently. As a result, the agitation rate should be fixed to its upper bound of 100.

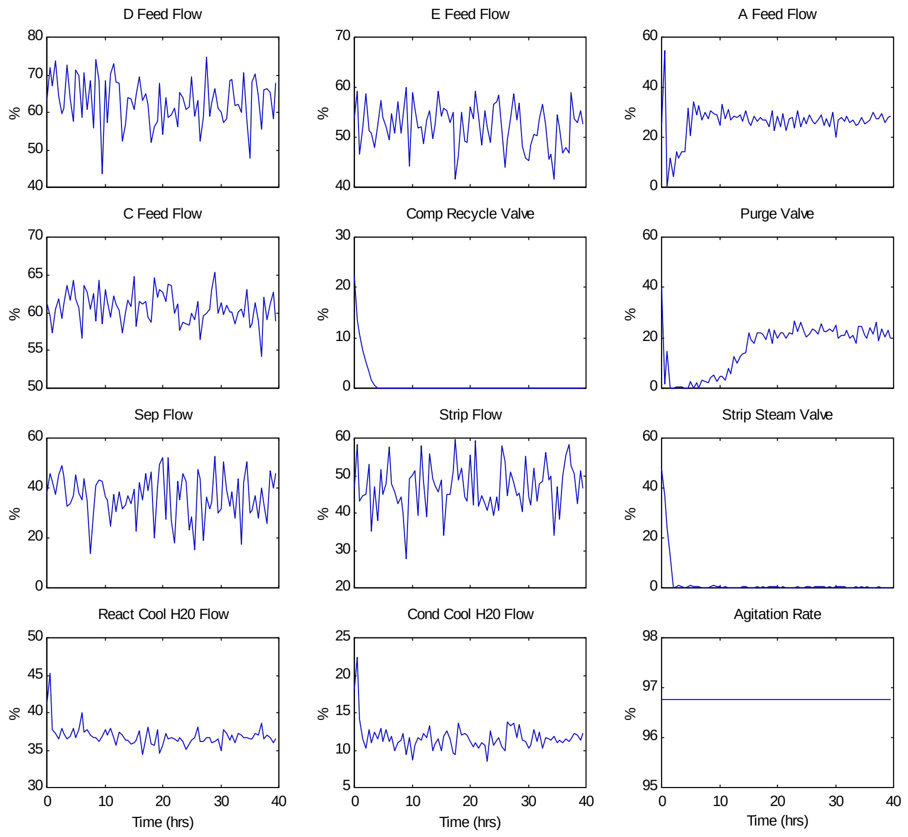


Figure 6.1: Inputs for the minimum-cost solution

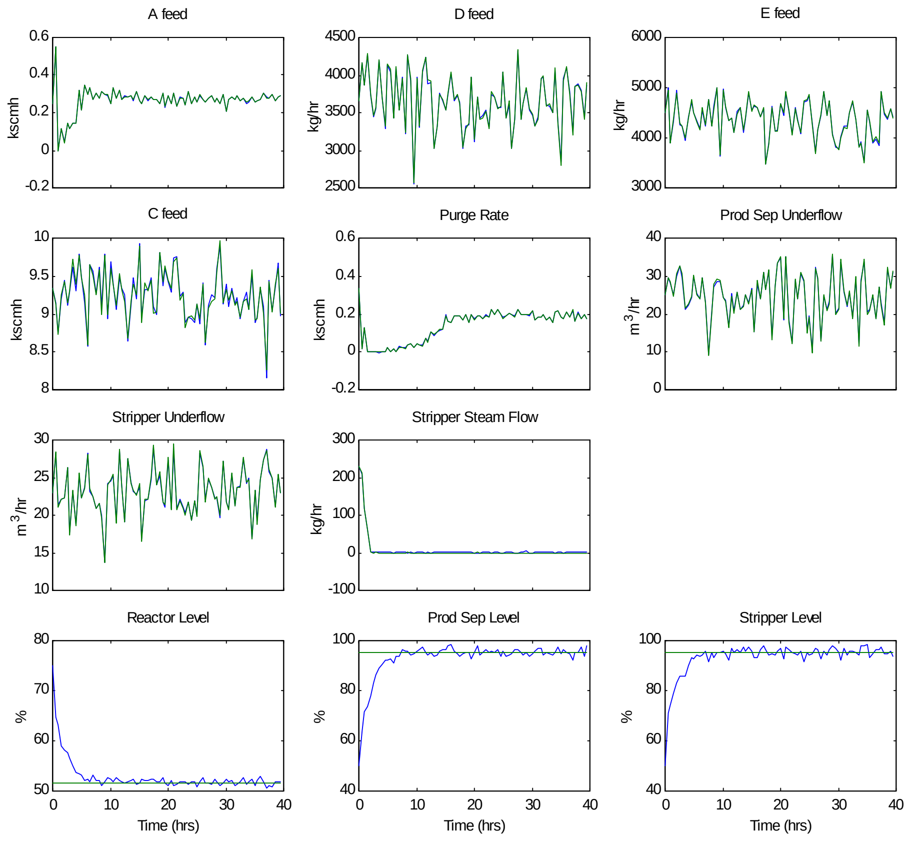


Figure 6.2: Selected outputs for the minimum-cost solution

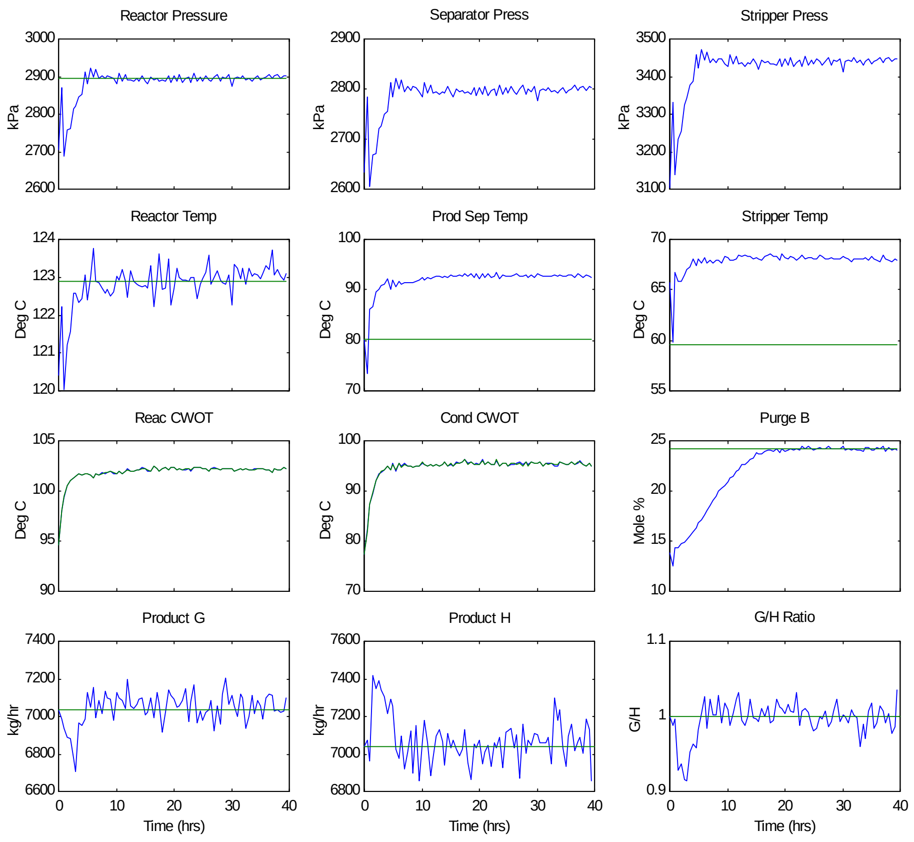


Figure 6.3: Selected outputs for the minimum-cost solution

An additional controller is implemented to handle the case when the reactor pressure increases beyond acceptable limits. When the reactor pressure increases above 2950 kPa due to a disturbance or setpoint change, the production rate setpoint is decreased by 25% until the reactor pressure decreases below 2895 kPa. The system reacts slow enough to this change that the production rate and product quality are not significantly affected. This override could have been implemented using a PI or similar type controller, but was found to be unnecessary.

Examining the simulation plots of Figures 6.1 to 6.3, the system briefly operates outside of the normal operating limits of Table1-3. The system operates outside the constraints due to the measurement noise and transients involved in changing setpoints, but in general does not approach the shutdown limits. There is still considerable margin from system shutdown, but if the system must operate within its normal operating limits, the boundaries of Table 4-1 can be adjusted to reflect the required margin, although at a monetary cost. In addition, the product quality is briefly disrupted when moving from the base-case setpoints to the near-optimal setpoints; both the G and H products show considerable fluctuations within the first 10 hours.

6.1.2 Results from Disturbances and Setpoint Changes

The results from disturbances 1, 4, 12, and 15 show almost perfect disturbance rejection due to the original design of the decentralized controller. Initially, disturbance 8 caused the plant to shutdown due to a build-up of reactor pressure. With the controller in place to reduce the production rate setpoint when the reactor pressure increases beyond acceptable limits, the system avoids shutdown. Selected inputs and outputs for this configuration are displayed in Figures 6.4 and 6.5. For this simulation, the compressor and stripper steam valves are both closed. The agitation rate is set to its maximum value. Both the purge and the A feed valves briefly saturate, but at steady-state are within acceptable limits.

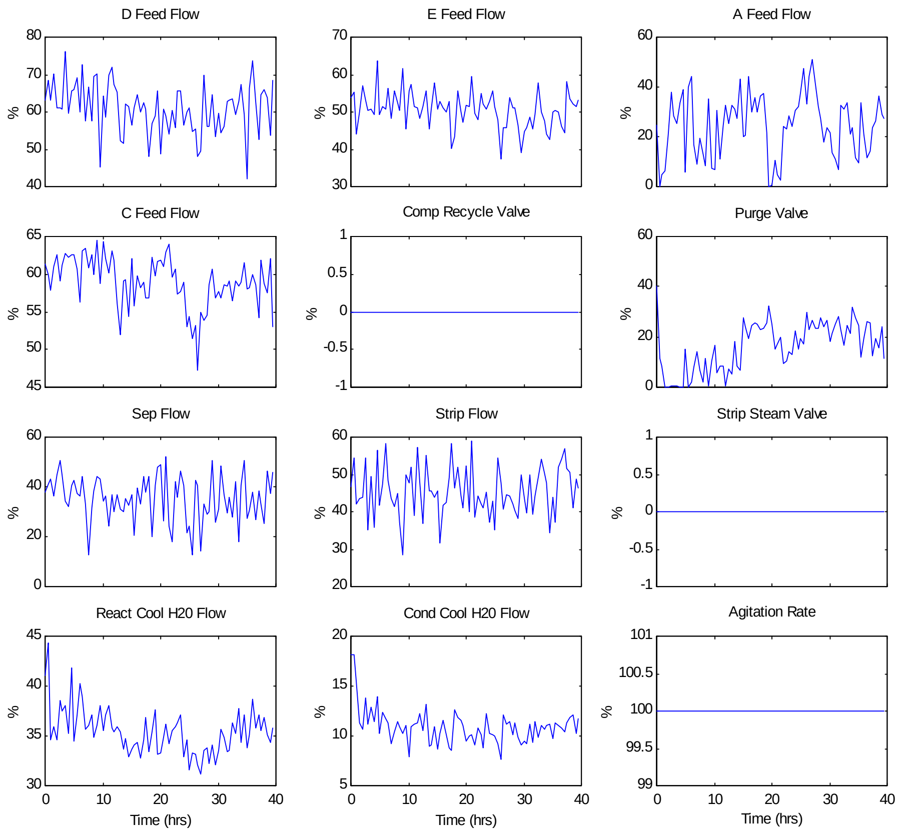


Figure 6.4: Input signals with IDV(8) active

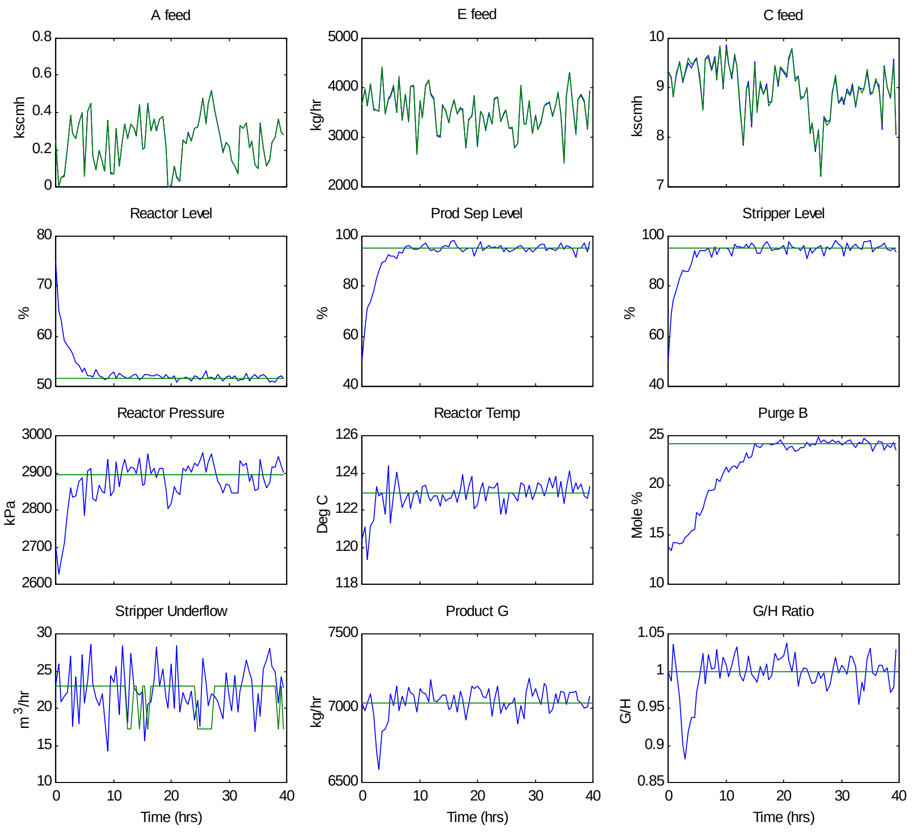


Figure 6.5: Selected outputs with IDV(8) active

Of all the disturbance scenarios, only IDV(6) could not be rejected. When the plant operates close to its limits and the A feed becomes uncontrollable, the system does not have enough time to react before the pressure builds up, and the plant shuts down. Several overrides can be used to temporarily prevent the process from shutting down when the A feed becomes uncontrolled. Decreasing the production rate to increase the amount of liquids in the system, and opening the purge valve to expel the excess gases are two potential overrides. Although it is desired to be able to control this disturbance for all time, it has yet to be completely controlled. IDV(6) was not one of the test disturbances recommended by the authors [6] because of its difficulty and infrequent occurrence.

Since the system has moved far from the base-case settings and the system is operating close to the normal limits, the recommended setpoint changes are not easily achieved. Setpoint changes are done by user interference, therefore before changing the setpoints, the system should be returned to the base-case setpoints. It was seen in Chapter 3 that the system with the base-case settings could move to the recommended setpoints with little disruption.

6.2 Conclusions

This study has proven several concepts that could be applied to most chemical processes. The first concept is that a genetic algorithm operating on a chemical process can determine parameters for near-optimal operating costs given cost information. Although the algorithm was used to minimize the operating costs, the optimization problem could be posed to determine optimal production, reduce variability, or optimize some other parameters. The problem was proposed as an off-line algorithm with a plant simulator, although the algorithm could be performed on-line for a system that does not have the stability and plant shutdown restrictions of the TE process. Although not implemented very often, on-line genetic algorithms have the advantage that if the system dynamics change with time, the controller can continually adjust its parameters to reflect the plant changes.

A second important concept is that a fuzzy controlled genetic algorithm can decrease the time to reach the minimum-cost solution. Reaching a significant minimum solution took approximately the same amount of time as the base genetic algorithm, although, the fuzzy genetic algorithm has superior performance at estimating the global minimum. Reducing the convergence time has large benefits in chemical processes which typically take a long time to reach steady-state values.

6.3 Future Work

The genetic algorithm has found an approximate minimum-cost solution around the bas-case mode. Since the base-case settings for this controlled system have been optimized for costs, a genetic algorithm, or other search algorithm, can use this data to focus on the variables that require higher resolution. Higher resolution may not be necessary as measurement noise and resolution of the valve controls may limit the significant digits. The genetic algorithm determined a “good” set of variables such as reactor pressure, reactor level, and the separator and stripper levels, therefore, their variations can be removed from future search algorithms that focus on the cost.

One crossover point per variable resulted in quicker convergence than using two randomly selected crossover points, but the idea of several multiple crossover points randomly distributed about the string was not pursued. Since two crossover points resulted in a much quicker convergence than a single crossover point, a third crossover point would be expected to converge even faster. One benefit of using randomly selected crossover points is that it allows interaction between variables, which in chemical processes, “is not necessarily a bad thing” [11]. A single crossover point per variable tends to isolate each of the variables, however, using three general crossover points would probably result in more configurations that cause plant shutdown.

The final fuzzy controller did not improve the genetic algorithm performance as much as desired. Previous research in optimizing GA parameters have only shown slight improvement, as well. The proposed fuzzy controller was unable to converge in significantly shorter time periods. The amount of time for tuning the fuzzy rules was reduced because simulating the plant takes so long. Although the problem is a realistic system, the amount of time required for a single simulation of 40 plant operating hours is very long for testing complex theories. Experimentation with a reduced-order system may provide a larger database of expert knowledge for the fuzzy controller, which can later be applied to the full-order process. As faster computers become available, the plant simulation time may become a moot point.

Chapter 5 briefly explored incorporating the population diversity into a fuzzy controller. The population diversity of a genetic algorithm becomes important as the individuals converge to local minimums, instead of the global minimum. If the individuals converge to a single value, the fuzzy controller should make random or directed changes to avoid depletion of the population diversity. Other than using the two-dimensional diversity fuzzy controller explained in Chapter 5, a multi-dimensional fuzzy controller could be created that includes the diversity, average fitness, maximum fitness, and individual fitnesses as inputs. The key control parameters to adjust are still the probabilities of crossover and mutation, although other parameters described in [8] could also be varied.

A major factor in the amount of research completed was the amount of time required to simulate the plant. The Euler integration algorithm was used to integrate the states forward in time because the routine is computationally fast. Most chemical processes are considered stiff systems because of the combination of the very fast and slow dynamics. Although computationally more intensive, routines that solve stiff differential equations, such as the Gear or Adams-Moulton routines, may simulate the system in shorter time periods and with more accuracy. In these routines, as the system approaches steady-state, the sample time increases. This could lead to quicker results allowing the researcher more time for experimentation.

Finally, the controller of the system can be redesigned to reflect the ability to operate over all the modes and meet all the disturbance scenarios. The system was originally designed to be comparable to previous designs which were operated exclusively in the first mode. To date, only Ricker in [18] has been able to operate over all modes using a decentralized controller. Although development of the controller is an important point, the main focus of this research was to determine if a genetic algorithm, or a fuzzy controlled genetic algorithm could determine improved settings for minimum-cost operation. Based on the findings from this research, if the controller had been designed to operate over all modes, the fuzzy based genetic algorithm could find the minimum operating cost of each mode. Besides operating over the other modes of the TE process, this procedure could be followed on similar processes where a realistic model of the plant is available and well-defined.

REFERENCES

1. Azam, F. and VanLandingham, H. F., “Fuzzy adaptive genetic algorithms,” *Advances in Systems, Signals, Control and Computers*. **2**, 231-235, 1998.

2. Bäck, T., *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

3. Banerjee, A. and Arkun, Y., “Control configuration-design applied to the Tennessee Eastman plant-wide control problem,” *Computers and Chemical Engineering*. **19**, 453-480, 1995.

4. Davis, L., *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

5. De Jong, K. A., “An analysis of the behavior of a class of genetic adapted systems (Doctoral Dissertation),” *Dissertation Abstracts International*. **36**, 5140B, 1975.

6. Downs, J. J. and Vogel E. F., “A plant-wide industrial process control problem,” *Computers and Chemical Engineering*. **17**, 245-255, 1993.

7. Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Publishing Company, 1989.

8. Grefenstette, J.J., “Optimization of control parameters of genetic algorithms,” *IEEE Transactions on Systems, Man, and Cybernetics*. **16**, 1986.

9. Holland, J., *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

10. Lee, C. C., “Fuzzy logic in control systems: fuzzy logic controller I and II,” *IEEE Transactions on Systems, Man, and Cybernetics*. **20**, 404-419, 1990.

11. Luyben, W. L., *Process Modeling, Simulation and Control for Chemical Engineers*. McGraw-Hill, 1990.

12. Luyben, W.L., “Simple regulatory control of the Eastman process,” *Industrial and Engineering Chemistry Research*. **35,** 3280-3289, 1996.

13. Lyman, P.R. and Georgakis, C., “Plant-wide control of the Tennessee Eastman problem,” *Computers and Chemical Engineering*. **19**, 321-331, 1995.

14. *MATLAB 5.0*. Computer software. The MathWorks, Inc., 1997.

15. McAvoy, T. J. and Ye, N., “Base control for the Tennessee Eastman problem,” *Computers and Chemical Engineering*. **18**, 383-413, 1994.

16. *MINOS*. Computer software. Stanford Business Software, Inc., 1985.

17. Prett, D. M. and Morari, M., *Shell Process Control Workshop*. Butterworth Publishers, 1986.

18. Ricker, N.L., “Decentralized control of the Tennessee Eastman challenge process,” *Journal of Process Control*. **6**, 205-221, 1996.

19. Ricker, N.L., “Optimal steady-state operation of the Tennessee-Eastman challenge process,” *Computers and Chemical Engineering*. **19**, 949-959, 1995.

20. Ricker, N.L. and Lee, J.H., “Nonlinear model-predictive control of the Tennessee-Eastman challenge process,” *Computers and Chemical Engineering*. **19**, 961-981, 1995.

21. Ross, T. J., *Fuzzy Logic with Engineering Applications*. McGraw-Hill, 1995.

22. *SOLNP*. Computer software. University of Iowa: College of Business Administration, 1989.

23. Srinivas, M. and Patnaik, L. M., “Adaptive probablilities of crossover and mutation in genetic algorithms,” *IEEE Transactions on Systems, Man, and Cybernetics*. **4**, 656-667, 1994.

24. Srinivas, Palavajjhala. Washington University, 5 May 1999. <http://www.che.wustl.edu/pub/pcsl/index.html>.

25. Zadeh, L. A. , “Fuzzy sets,” *Information and Control*. **8**, 338-353, 1965.

26. Zadeh, L. A., “Outline of a new approach to the analysis of complex systems and decision processes,” *IEEE Transactions on Systems, Man, and Cybernetics*. **3**, 28-44, 1973.

Appendix A

Computer Code

This appendix contains the computer code that was created to simulate and control the Tennessee Eastman process. First a brief overview of the code is described, followed by the actual code at the end of the appendix.

opt2_n.m - File to try to find steady-state base-case costs using constraints posed by Ricker. This file uses the 50 states as inputs to the program SOLNP. The file cost.m is being optimized.

cost.m - File that is optimized by the opt2_n.m program. This file is required as part of the SOLNP program.

ga99_15_n.m - This file does the brute work of the genetic algorithm. It creates a random initial population, encodes the population, calls the plant simulator and inputs the individual fitnesses. Once all of the individuals have been evaluated, the program performs the reproduction, crossover, and mutation operators. A call to the fuzzy controllers is included which may be commented out when not in use. The program also outputs to the terminal information regarding the costs and most fit individual.

TEMAIN30.FOR - Interface routine that requires sets.dat as input. This file outputs several variables including input values, setpoints, outputs and costs. This file also includes my controller written in FORTRAN

minplant.m - File used to see if MATLAB fmins (simplex method) could determine the optimal parameters for setpoints and agitation rate. The file it minimizes is minyp.m.

minyp.m - MATLAB function that tries to minimize the cost based on 10 inputs (setpoints and agitation rate). Requires call from minplant.m

fuzzpc.m - Final fuzzy logic controller to determine probability of crossover based on fitness values. The controller is implemented as a MATLAB function call.

fuzzpm.m - Final fuzzy logic controller to determine probability of mutation based on fitness values. The controller is implemented as a MATLAB function call.

```
% opt2_n.m
% 3/31/99
%
% This file replaces the MINOS program to replicate the optimal steady state
% soln using the SOLNP program.
%
clear all
home;
format short compact
t0=clock;
warning off;
nn=50; deltasec=1;
[time,yy,yp,xm,xmv,wspace]=teinit_n(nn);  % Needed wspace
% Initialize plant and variables

lx=zeros(50,1);                           % Lower bound of 0 on states;
ux=[10^8*ones(38,1); 100*ones(12,1)];     % Upper bound on states and XMV's
xb=[lx, ux];
i0=[xm(12)-.001; xm(15)-.001; xm(7)-.001; xm(8)+.001];% 0 150];
lh=[30; 30; 0; 50];% 0];
uh=[100; 100; 2895; 100];%150];
ib=[i0,lh,uh];
op=[0,100,4,10^-5,10^-4];
[vectopt,oh,y1,h1,ic]=solnp([yy,xb],ib, op);
yy=vectopt(1:50); xmv=vectopt(39:50);
ib=[ic,lh,uh];
op=[0,100,4,10^-6,10^-5];
[vectopt,oh,y1,h1,ic]=solnp([yy,xb],ib, op);
yy=vectopt(1:50); xmv=vectopt(39:50);
done=2; disp('Done 2')
ib=[ic,lh,uh];
op=[0,100,4,10^-7,10^-6];
[vectopt,oh,y1,h1,ic]=solnp([yy,xb],ib, op);
yy=vectopt(1:50); xmv=vectopt(39:50);
done=3; disp('Done 3')
ib=[ic,lh,uh];
op=[0,100,4,10^-8,10^-7];
[vectopt,oh,y1,h1,ic]=solnp([yy,xb],ib, op);
yys=vectopt(1:50); xmvs=vectopt(39:50);
disp('Done 4')
disp(['Total time is ' num2str(etime(clock,t0)/60) ' minutes.']);
disp(char(7));                            % Beep when done
%
```

```
function [f]=cost(yy,par);
% cost.m
%
% This file  is a function file to do the optimal steady-state values
% for the TE plant using the SOLNP program
clear mex;

nn=50; xmv=yy(39:50,1); deltasec=1;
[time,yy2,yp2,xm,xmv2,wspace]=teinit_n(nn);     % Needed wspace

[yy,yp,xm,xmv,wspace]=tefunc_n(nn,time,yy,xm,xmv,deltasec,wspace);
```

```
f(1) = 0.0536*xm(20) + 0.0318*xm(19) + 0.44791*xm(10) *
(2.206*xm(29)+6.177*xm(31)+...
   22.06*xm(32)+14.56*xm(33)+17.89*xm(34)+30.44*xm(35)+22.94*xm(36))+...
   4.541*yy(46) * (0.2206*xm(37)+0.1456*xm(38)+0.1789*xm(39));
for i=1:50
   f(i+1)=yp(i);
end
f(52)=1-2.8153*yy(46)*xm(40)/7038;              % Product G
f(53)=1-3.4510*yy(46)*xm(41)/7038;              % Product H
f(54)=xm(12);                                   % Separator Level
f(55)=xm(15);                                   % Stripper Level
f(56)=xm(7);                                    % Reactor Pressure
f(57)=xm(8);                                    % Reactor Level
%f(58)=xm(9);                                   % Reactor Temp -- not used
f=f';
```

```
% ga99_15_n.m
% 5/26/99
%
% This file will do the genetic algorithm for the TE chemical
% process adjusting the setpoints for configuration 3.  It
% currently uses my inner loop PI controller and adjusts
% the setpoints on the outer loop controllers and agitation rate.
% This version has 5 bits of accuracy instead of 4.
% I am changing things around again -- In particular I am using
% one crossover point per setpoint.
% Mutation changed to have a probability of 1% per setpoint.
% This file also has a preprocessing step to make sure have
% a good initial pop.
% This one uses roulette wheel.  Use sets3.m for plotting later.
% This one can use a fuzzy controller to control rates of
% crossover and mutation.
%
% Right now -- one crossover pt per setpt, pc=0.7, pm=.01
% Use roulette wheel
clear all; close all;
clc;                    % Clear the screen
t0=clock;               % To see how long it takes
disp('ga99_15_n in Progress!!');
disp(['Simulation started on ',int2str(t0(2)),'/',int2str(t0(3)),'/',...
   int2str(t0(1)), ' at ', int2str(t0(4)), ':', int2str(t0(5)), '.']);
      disp('Setpoints: 5, 7, 8, 9, 12, 15, 20, 30, 18, XMV12');
format compact; format short;

%Initialize variables
iter=0;
Npts=50; Nbits=50; Nbits_set=5;
elite=2;                % Make this an even number
pc=0.7; pm=0.01;        % Initial probabilites of crossover and mutation
tour_num=3;             % Number of individuals in each tournament selection
eps=10^-8;              % For random number generators (e.g. want 0 <= # < 1)


% Initialization for tefunc program -- reduces processing time.
bestind=[]; smallcost=[];          % For saving data
nn=50;                                               % Use 50 states
deltasec=1; deltat=1/3600;         % deltasec is integer seconds
hrs=40;     % Assuming transients died by this point -- maybe bad assumption
```

```
indb4=[]; costb4=[];                      % To avoid repeat evaluation

% Initialization for outputting costs to a data file
for_out='%4.0f';                          % Format for outputting costs
for j=1:Npts                              % Extra one for iteration
   for_out=[for_out '%10.4f'];
end

% Initialize random set
rand('state',sum(100*clock));       % Fairly random set of values
randstate=rand('state');            % Seed for if want to repeat simulation
ind=round(rand(Npts,Nbits));

r=0;
if r>0                              % Use this when have good init pop

% Done initialization
% This preprocessing part determines if range of population is good
% to make sure have good initial population

hrs=20;
i=1; num_mist=0;
while i<=Npts
   setoffset=16*ind(i,1)+8*ind(i,2)+4*ind(i,3)+2*ind(i,4)+1*ind(i,5);
   setpt(5)=22+setoffset*15/31;           % Between 22-37
   setoffset=16*ind(i,6)+8*ind(i,7)+4*ind(i,8)+2*ind(i,9)+1*ind(i,10);
   setpt(7)=2600+setoffset*295/31;        % Between 2600-2895
   setoffset=16*ind(i,11)+8*ind(i,12)+4*ind(i,13)+2*ind(i,14)+1*ind(i,15);
   setpt(8)=50+setoffset*50/31;           % Between 50-100
   setoffset=16*ind(i,16)+8*ind(i,17)+4*ind(i,18)+2*ind(i,19)+1*ind(i,20);
   setpt(9)=120+setoffset*30/31;          % Between 120-150
   setoffset=16*ind(i,21)+8*ind(i,22)+4*ind(i,23)+2*ind(i,24)+1*ind(i,25);
   setpt(12)=30+setoffset*70/31;          % Between 30-100
   setoffset=16*ind(i,26)+8*ind(i,27)+4*ind(i,28)+2*ind(i,29)+1*ind(i,30);
   setpt(15)=30+setoffset*70/31;          % Between 30-100
   setoffset=16*ind(i,31)+8*ind(i,32)+4*ind(i,33)+2*ind(i,34)+1*ind(i,35);
   setpt(20)=220+setoffset*180/31;  % Between 220-400
   setoffset=16*ind(i,36)+8*ind(i,37)+4*ind(i,38)+2*ind(i,39)+1*ind(i,40);
   setpt(30)=setoffset*50/31;             % Between 0-50
   setoffset=16*ind(i,41)+8*ind(i,42)+4*ind(i,43)+2*ind(i,44)+1*ind(i,45);
   setpt(18)=55+setoffset*20/31;          % Between 55-75
   setoffset=16*ind(i,46)+8*ind(i,47)+4*ind(i,48)+2*ind(i,49)+1*ind(i,50);
   xmv12=setoffset*100/31;                % Between 0-100

drawnow;
%geo7_n;          % Run thru fortran code
   set=[setpt(5), setpt(7), setpt(8), setpt(9), setpt(12), setpt(15),...
         setpt(20), setpt(30), setpt(18), xmv12];
      fid=fopen('d:\users\jsozio\mfiles\sets.dat','w');
      fprintf(fid,'%15.12f %15.10f %15.11f %15.11f %15.11f %15.11f %15.11f
%15.12f %15.12f %15.11f \n', set);
      fclose(fid);
      !d:\users\jsozio\mfiles\teout20
   load d:\users\jsozio\mfiles\cst.dat
   cst=cst(5);                            % Just want total cost
      cost(i)=cst;
if cst>500
      ind(i,:)=round(rand(1,Nbits));      % Try a different random individual
       num_mist=num_mist+1;
else
   cost(i)=cst;
   i=i+1;
   save starti i num_mist randstate       % Can look at to determine where at
end
end
disp(['Done beginning in ' num2str(etime(clock,t0)/3600) ' hours with '...
      int2str(num_mist) ' initial errors.' ]);
save goodind4 ind % save initial population
end                          % Assume have a good initial pop
load goodind5
%hrs=40;                     % Preprocessing used time of 20 hrs.

% Determine offsets -- uses 5 bits per setpoint
% This will also do the evaluation of the plant
% This part takes a long time
while iter<=150
    t1=cputime;
for i=1:Npts
   setoffset=16*ind(i,1)+8*ind(i,2)+4*ind(i,3)+2*ind(i,4)+1*ind(i,5);
   setpt(5)=22+setoffset*15/31;           % Between 22-37
   setoffset=16*ind(i,6)+8*ind(i,7)+4*ind(i,8)+2*ind(i,9)+1*ind(i,10);
   setpt(7)=2600+setoffset*295/31;        % Between 2600-2895
   setoffset=16*ind(i,11)+8*ind(i,12)+4*ind(i,13)+2*ind(i,14)+1*ind(i,15);
   setpt(8)=50+setoffset*50/31;           % Between 50-100
   setoffset=16*ind(i,16)+8*ind(i,17)+4*ind(i,18)+2*ind(i,19)+1*ind(i,20);
   setpt(9)=120+setoffset*30/31;          % Between 120-150
   setoffset=16*ind(i,21)+8*ind(i,22)+4*ind(i,23)+2*ind(i,24)+1*ind(i,25);
   setpt(12)=30+setoffset*70/31;          % Between 30-100
   setoffset=16*ind(i,26)+8*ind(i,27)+4*ind(i,28)+2*ind(i,29)+1*ind(i,30);
   setpt(15)=30+setoffset*70/31;          % Between 30-100
   setoffset=16*ind(i,31)+8*ind(i,32)+4*ind(i,33)+2*ind(i,34)+1*ind(i,35);
   setpt(20)=220+setoffset*180/31;        % Between 220-400
   setoffset=16*ind(i,36)+8*ind(i,37)+4*ind(i,38)+2*ind(i,39)+1*ind(i,40);
   setpt(30)=setoffset*50/31;             % Between 0-50
   setoffset=16*ind(i,41)+8*ind(i,42)+4*ind(i,43)+2*ind(i,44)+1*ind(i,45);
   setpt(18)=55+setoffset*20/31;          % Between 55-75
   setoffset=16*ind(i,46)+8*ind(i,47)+4*ind(i,48)+2*ind(i,49)+1*ind(i,50);
   xmv12=setoffset*100/31;                % Between 0-100

   if i==1
      set=[setpt(5), setpt(7), setpt(8), setpt(9), setpt(12), setpt(15),...
         setpt(20), setpt(30), setpt(18), xmv12]
      end

breakflag=0;
drawnow;
% Following code so don't have to run thru if already calculated cost
for p=1:length(costb4)
   if ind(i,:)==indb4(p,:)
      cost(i)=costb4(p);
      breakflag=1;
      break;
   end
end

if breakflag==0
   % Run thru fortran code for plant simulation
   % Fortran code 2x faster than mex files
   set=[setpt(5), setpt(7), setpt(8), setpt(9), setpt(12), setpt(15),...
         setpt(20), setpt(30), setpt(18), xmv12];
      fid=fopen('d:\users\jsozio\mfiles\sets.dat','w');
      fprintf(fid,'%15.12f %15.10f %15.11f %15.11f %15.11f %15.11f %15.11f
%15.12f %15.12f %15.11f \n', set);
      fclose(fid);
      !d:\users\jsozio\mfiles\teoutb
   load d:\users\jsozio\mfiles\cst.dat
   cst=cst(5);          % Just want total cost
      cost(i)=cst;
   if cst<500           % Only save best costs to avoid indb4 too big
      indb4=[ind(i,:); indb4];
      costb4=[cst; costb4];
            [costb4new, posb4]=sort(costb4);
            % Move individuals in ascending order (best->worst)
            for p=1:length(costb4)
               indb4new(p,:)=indb4(posb4(p),:);
            end
        costb4=costb4new;     % Sorted to try to keep the smallest costs
        indb4=indb4new;
      if length(costb4)>300   % clear values if getting too big
         costb4(301)=[];
         indb4(301,:)=[];
      end
   end
end
end

t2=cputime; % Time for passing individuals thru plant
disp(['Done eval for iter ', int2str(iter), ' in ' num2str((t2-t1)/60) '
minutes.']);


% Gone thru all the points -- Now do GA
indnew=[]; costsort=[]; costsort2=[]; roulette=[];    % Clear variables
totcst=0; bestcst=0; bestcst20=0;
[costsort, position]=sort(cost);
for i=1:Npts      % Move individuals in ascending order (best->worst)
   indsort(i,:)=ind(position(i),:);
end
% Elitism -- get best = elite
for i=1:elite
   indnew(i,:)=indsort(i,:);
end

% Set up roulette wheel
costsort2b=1./costsort;       % Want the minimum cost a maximum profit
for i=1:Npts
   totcst=totcst+costsort2b(i);
   roulette(i)=totcst;        % Summed cost of individuals
end
roulette=roulette/totcst;     % Normalize roulette to 1

% Crossover, mutation and reset of ind
for i=1:(Npts-elite)/2        % Fill up indnew

% Crossover and Mutation
% Get individuals using roulette wheel
for j=1:2
   randcst1=rand(1);
   for i2=1:Npts
      if randcst1<roulette(i2)
         firstind(j,:)=indsort(i2,:);
         mintourcst(j)=costsort(i2);
         break;
      end
      end
end

% Fuzzy controller for crossover goes here
fmax=costsort(1);                    % Best cost
fprime=min(mintourcst);              % Max cost of parents
favg=sum(costsort(1:20))/20;         % Average cost of best 20 individuals
pc=fuzzpc(fprime,favg,fmax);         % Between 50 and 100%
if rand(1)<pc                        % Decide to do crossover or not
for p=0:Nbits_set:(Nbits-Nbits_set)
crosspt=floor((Nbits_set-1-eps)*rand(1))+1;  % Crossover at 1 to Nbits_set-1
for t=1:crosspt
   indnew(2*i-1+elite,t+p)=firstind(1,t+p);
   indnew(2*i+elite,t+p)=firstind(2,t+p);
end
for t=crosspt+1:Nbits_set
   indnew(2*i-1+elite,t+p)=firstind(2,t+p);
   indnew(2*i+elite,t+p)=firstind(1,t+p);
end
end

else                                 % Use straight reproduction
   indnew(2*i-1+elite,:)=firstind(1,:);
   indnew(2*i+elite,:)=firstind(2,:);
end

% Mutation on each setpoint
for j=0:1                                  % 1st or 2nd new individual
pm=fuzzpm(mintourcst(j+1),favg,fmax);      % Between 0 and 3%
for j1=0:round(Nbits/Nbits_set-1)          % Beginning of setpoint
   if rand(1)<pm
      mutpt=floor((Nbits_set-eps)*rand(1))+1;
      % Random mutation point 1->Nbits_set
      indnew(2*i-j+elite,mutpt+j1*Nbits_set)=xor(1,indnew(2*i-
j+elite,mutpt+j1*Nbits_set));
   end
   end
end

end   % Done next generation of individuals (indnew filled)

% Update for Next generation
iter=iter+1;
ind=indnew;
bestind=[bestind; indnew(1,:)];            % Best individual at iteration i
smallcost=[smallcost, costsort(1)];        % Cheapest individual soln.
for i=1:Npts                               % Calculate average cost
   bestcst=costsort(i)+bestcst;
end
csttrk(iter)=bestcst/i;

for i=1:20                                 % Calculate average cost of best 20
   bestcst20=costsort(i)+bestcst20;
end
cst20(iter)=bestcst20/i;

totclk=etime(clock,t0)/3600;
titer=etime(clock,t1)/60;
currtime=clock;
```

```
disp(['Total time so far is ', num2str(totclk), ' hours at ',...
      int2str(currtime(4)), ':', int2str(currtime(5)), '.']);
%disp(['Time for GA processing is ' num2str(cputime-t2) ' seconds.']);
numiter(iter)=iter-1;
subplot(311); plot(numiter, csttrk); title('Average Cost');
subplot(312); plot(numiter,cst20); title('Best 20 Average Cost');
subplot(313); plot(numiter,smallcost); title('Best Cost');
xlabel('Generation')
disp(['Best Cost is ', num2str(costsort(1))]);
save cross5st indsort costsort csttrk cst20 bestind smallcost randstate

% Track Costs -- outputs costs of all 50 individuals at each iteration
fid = fopen('cross5st.dat','a');    % Need next 3 lines every iteration
fprintf(fid,[for_out '\n'], [iter costsort]);
fclose(fid);

clear indnew costsort2 roulette     % Clear variables to free memory
end
disp(char(7));                      % Beep when done
```

```
C  TEMAIN30.FOR
C
C  This file requires sets.dat in the right format shown below
C  It outputs the steady-state cost;
C  It also outputs the setpoints, manipulated and measurement
C  variables at the user specified intervals
C
C  Use tedat1.m to plot -- this one similar to geo5b_n
C
C  Main program for demonstrating application of the Tennessee Eastman Process
Control Test Problem
C=================================================================
C  Measurement and valve common block; Disturbance vector common block;
C  Controller common block; Local Variables.

      DOUBLE PRECISION XMEAS, XMV
      COMMON/PV/ XMEAS(41), XMV(12)
      DOUBLE PRECISION CST1,CST2,CST3,CST4,CST,XM,LNGTH
      COMMON/OUT1/ CST1,CST2,CST3,CST4,CST,XM(41),LNGTH
      INTEGER IPTS, CNT
      COMMON/OUT/ IPTS, CNT
      INTEGER IDV
      COMMON/DVEC/ IDV(20)
      DOUBLE PRECISION ERROLD, DELTAT, SETPT
      COMMON/CTRL/ ERROLD(41), DELTAT, SETPT(41)
      DOUBLE PRECISION TIME, YY(50), YP(50)
      INTEGER I, NN, NPTS, DATAOUT, DATAOUT2, J, HRS
      OPEN (50,FILE='d:\users\jsozio\mfiles\sets.dat')
      OPEN (51,FILE='d:\users\jsozio\mfiles\cst.dat')
      OPEN (52,FILE='d:\users\jsozio\mfiles\teoutb.dat')
      OPEN (53,FILE='d:\users\jsozio\mfiles\setpts.dat')

C  Set the number of differential equations (states). The process has 50
states.
c  If the user wishes to integrate additional states for CT controllers, NN
must
C  be increased by the number of additional differential equations.
      NN = 50
      HRS = 40
C  Set the number of points to simulate
      NPTS = 3600*HRS-5
C  Integrator Step Size:  1 Second Converted to Hours
      DELTAT = 1./3600.
C  Initialize Process (Sets TIME to zero)
      DATAOUT = 0
      DATAOUT2 = 0
      CNT = 1000
      LNGTH=0.0
C  Initialize errors to 0.0
      DO 101 I = 1, 41
      ERROLD(I) = 0.0
 101  CONTINUE
C  Set all Disturbance Flags to OFF
      DO 102 I = 1, 20
      IDV(I) = 0
 102  CONTINUE
      IPTS=1
C  Clear cost variables
      DO 103 I = 1, 41
      XM(I) = 0.0
 103  CONTINUE
C  Call the initialization routine
      CALL TEINIT(NN,TIME,YY,YP)
C  Create setpoints
      DO 222 I= 1, 41
      SETPT(I) =  XMEAS(I)-0.00*XMEAS(I)
 222  CONTINUE
C  G/H RATIO AND PRODUCTION RATIO
      SETPT(40) = (0.5*XMEAS(40)) / (0.5*XMEAS(41))
      SETPT(41) = XMEAS(2) / XMEAS(3)
      SETPT(27) = SETPT(17)
      CALL INPUT
      CALL OUTPUT2
C Disturbance

C  Simulation Loop;
      DO 1000 I = 2, NPTS
      IPTS=I
      CALL CONTRL3(NN,YY,YP)
      DO 223 J = 1,12
      IF(XMV(J).LT.0.0)THEN
      XMV(J) = 0.0
      ELSEIF(XMV(J).GT.100.0)THEN
      XMV(J) = 100.0
      ENDIF
 223  CONTINUE
      DO 224 J = 1,41
      IF(SETPT(J).LT.0.0)THEN
      SETPT(J) = 0.0
      ENDIF
 224  CONTINUE
      CALL INTGTR(NN,TIME,DELTAT,YY,YP)
      DATAOUT2 = DATAOUT2 + 1
C     OUTPUT ONCE EVERY 10 MINUTES
      IF(DATAOUT2.GE.(60*10))THEN
      DATAOUT2 = 0
      CALL OUTPUT2
      ENDIF
C     SAVE LAST DATA FOR COST
```

```
      DATAOUT = DATAOUT+1
      IF(DATAOUT.GE.(HRS*3600-3604))THEN
      LNGTH=LNGTH+1.
      XM(10)=XMEAS(10)+XM(10)
      XM(19)=XMEAS(19)+XM(19)
      XM(20)=XMEAS(20)+XM(20)
      XM(29)=XMEAS(29)+XM(29)
      XM(31)=XMEAS(31)+XM(31)
      XM(32)=XMEAS(32)+XM(32)
      XM(33)=XMEAS(33)+XM(33)
      XM(34)=XMEAS(34)+XM(34)
      XM(35)=XMEAS(35)+XM(35)
      XM(36)=XMEAS(36)+XM(36)
      XM(37)=XMEAS(37)+XM(37)
      XM(38)=XMEAS(38)+XM(38)
      XM(39)=XMEAS(39)+XM(39)
      XM(1)=YY(46)+XM(1)
      ENDIF
 1000 CONTINUE
      CALL OUTPUT
      CLOSE (50,STATUS='KEEP')
      CLOSE (51,STATUS='KEEP')
      CLOSE (52,STATUS='KEEP')
      CLOSE (53,STATUS='KEEP')
c     WRITE(6,203)
c 203 FORMAT(1X, 'Done Output File')
      STOP
      END
C
C=================================================================
C   Measurement and valve common block

      SUBROUTINE INPUT
      DOUBLE PRECISION XMEAS, XMV
      COMMON/PV/ XMEAS(41), XMV(12)
      DOUBLE PRECISION ERROLD, DELTAT, SETPT
      COMMON/CTRL/ ERROLD(41), DELTAT, SETPT(41)
      DOUBLE PRECISION SETPTO(10)

      READ(50,207)SETPTO(1),SETPTO(2),SETPTO(3),SETPTO(4),SETPTO(5),
     .SETPTO(6),SETPTO(7),SETPTO(8),SETPTO(9),SETPTO(10)
 207  FORMAT(1F16.12,1F16.10,1F16.11,1F16.11,1F16.11,1F16.11,
     .1F16.11,1F16.12,1F16.12,1F16.11)
      SETPT(5)=SETPTO(1)
      SETPT(7)=SETPTO(2)
      SETPT(8)=SETPTO(3)
      SETPT(9)=SETPTO(4)
      SETPT(12)=SETPTO(5)
      SETPT(15)=SETPTO(6)
      SETPT(20)=SETPTO(7)
      SETPT(30)=SETPTO(8)
      SETPT(18)=SETPTO(9)
      XMV(12)=SETPTO(10)

      RETURN
      END
C=================================================================
C   Measurement and valve common block

      SUBROUTINE OUTPUT2
      DOUBLE PRECISION XMEAS, XMV
      COMMON/PV/ XMEAS(41), XMV(12)
      DOUBLE PRECISION ERROLD, DELTAT, SETPT
      COMMON/CTRL/ ERROLD(41), DELTAT, SETPT(41)
      INTEGER IPTS, CNT
      COMMON/OUT/ IPTS, CNT

      WRITE(52,217)IPTS,XMEAS(1),XMEAS(2),XMEAS(3),XMEAS(4),
     .XMEAS(5),
     .XMEAS(6),XMEAS(7),XMEAS(8),XMEAS(9),XMEAS(10),
     .XMEAS(11),XMEAS(12),XMEAS(13),XMEAS(14),XMEAS(15),
     .XMEAS(16),XMEAS(17),XMEAS(18),XMEAS(19),XMEAS(20),
     .XMEAS(21),XMEAS(22),XMEAS(23),XMEAS(24),XMEAS(25),
     .XMEAS(26),XMEAS(27),XMEAS(28),XMEAS(29),XMEAS(30),
     .XMEAS(31),XMEAS(32),XMEAS(33),XMEAS(34),XMEAS(35),
     .XMEAS(36),XMEAS(37),XMEAS(38),XMEAS(39),XMEAS(40),XMEAS(41),
     .XMV(1),XMV(2),XMV(3),XMV(4),XMV(5),XMV(6),
     .XMV(7),XMV(8),XMV(9),XMV(10),XMV(11),XMV(12)
 217  FORMAT(1X,I6,3X,F9.5,3X,F9.1,3X,F9.1,3X,F9.4,3X,F9.3,
     .3X,F9.3,3X,F9.1,3X,F9.3,3X,F9.2,3X,F9.5,
     .3X,F9.5,3X,F9.3,3X,F9.1,3X,F9.3,3X,F9.3,
     .3X,F9.1,3X,F9.3,3X,F9.3,3X,F9.2,3X,F9.2,
     .3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,
     .3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,
     .3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,
     .3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,
     .3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,
     .3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3,3X,F9.3)
C
      WRITE(53,218)IPTS,SETPT(1),SETPT(2),SETPT(3),SETPT(4),
     .SETPT(5),
     .SETPT(6),SETPT(7),SETPT(8),SETPT(9),SETPT(10),
     .SETPT(11),SETPT(12),SETPT(13),SETPT(14),SETPT(15),
     .SETPT(16),SETPT(17),SETPT(18),SETPT(19),SETPT(20),
     .SETPT(21),SETPT(22)
 218  FORMAT(1X,I6,3X,F9.5,3X,F9.1,3X,F9.1,3X,F9.4,3X,F9.3,
     .3X,F9.3,3X,F9.1,3X,F9.3,3X,F9.2,3X,F9.5,
     .3X,F9.5,3X,F9.3,3X,F9.1,3X,F9.3,3X,F9.3,
     .3X,F9.1,3X,F9.3,3X,F9.3,3X,F9.2,3X,F9.2,
     .3X,F9.3,3X,F9.3)
      RETURN
      END
C
C=================================================================
C   Measurement and valve common block

      SUBROUTINE OUTPUT
      DOUBLE PRECISION XMEAS, XMV
      COMMON/PV/ XMEAS(41), XMV(12)
      DOUBLE PRECISION CST1,CST2,CST3,CST4,CST,XM,LNGTH
      COMMON/OUT1/ CST1,CST2,CST3,CST4,CST,XM(41),LNGTH
      INTEGER IPTS, CNT
      COMMON/OUT/ IPTS, CNT
      INTEGER SHTDWN

      SHTDWN=0
      IF(XMEAS(7).GT.2900.)THEN
      SHTDWN=1
      ELSEIF(XMEAS(9).GT.150.)THEN
      SHTDWN=1
      ELSEIF(XMEAS(8).GT.100.)THEN
      SHTDWN=1
      ELSEIF(XMEAS(8).LT.50.)THEN
      SHTDWN=1
      ELSEIF(XMEAS(12).GT.100.)THEN
      SHTDWN=1
      ELSEIF(XMEAS(12).LT.30.)THEN
      SHTDWN=1
      ELSEIF(XMEAS(15).GT.100.)THEN
      SHTDWN=1
      ELSEIF(XMEAS(15).LT.30.)THEN
      SHTDWN=1
      ENDIF
      IF(SHTDWN.EQ.1)THEN
      CST=999.
      ELSE
      XM(10)=XM(10)/LNGTH
      XM(19)=XM(19)/LNGTH
      XM(20)=XM(20)/LNGTH
      XM(29)=XM(29)/LNGTH
      XM(31)=XM(31)/LNGTH
      XM(32)=XM(32)/LNGTH
      XM(33)=XM(33)/LNGTH
      XM(34)=XM(34)/LNGTH
      XM(35)=XM(35)/LNGTH
      XM(36)=XM(36)/LNGTH
      XM(37)=XM(37)/LNGTH
      XM(38)=XM(38)/LNGTH
      XM(39)=XM(39)/LNGTH
      XM(1)=XM(1)/LNGTH

      CST1=0.0536*XM(20)
      CST2=0.0318*XM(19)
      CST3=44.791*XM(10)*(2.206*XM(29)+6.177*XM(31)+22.06*XM(32)+
     .(14.56)*XM(33)+17.89*XM(34)+30.44*XM(35)+22.94*XM(36))/100.
      CST4=4.541*XM(1)*(22.06*XM(37)+14.56*XM(38)+17.89*XM(39))/100.
      CST=CST1+CST2+CST3+CST4
      IF(CST.GE.500.)THEN
      CST=500.
      ENDIF
      ENDIF
      WRITE(51,237)CST1,CST2,CST3,CST4,CST
 237  FORMAT(F8.4,1X,F8.4,1X,F8.4,1X,F8.4,1X,F8.4)
C
      RETURN
      END
C================================================================
C  Euler Integration Algorithm

      SUBROUTINE INTGTR(NN,TIME,DELTAT,YY,YP)
      INTEGER I, NN
      DOUBLE PRECISION TIME, DELTAT, YY(NN), YP(NN)
      CALL TEFUNC(NN,TIME,YY,YP)
      TIME = TIME + DELTAT
      DO 100 I = 1, NN
          YY(I) = YY(I) + YP(I) * DELTAT
 100  CONTINUE
      RETURN
      END
C
C=================================================================
C    Digital control algorithms;
C    Measurement and valve common block; Controller common block;
```

```
C    Proportional-Integral Controller (Velocity Form)
      SUBROUTINE CONTRL3(NN,YY,YP)
C
      DOUBLE PRECISION XMEAS, XMV
      COMMON/PV/ XMEAS(41), XMV(12)
      DOUBLE PRECISION ERR1, DXMV
      DOUBLE PRECISION ERROLD, DELTAT, SETPT
      COMMON/CTRL/ ERROLD(41), DELTAT, SETPT(41)
      INTEGER IPTS, CNT
      COMMON/OUT/ IPTS, CNT

C    PI Controllers for for flowrate/manipulated valve combinations
C     Inner Loop Controllers
C    Control Loop 1 -- A feed                   A feed flow
      ERR1 = SETPT(1) - XMEAS(1)
      DXMV = 65. * ((ERR1 - ERROLD(1)) + ERR1 * DELTAT * 60./0.1)
      XMV(3) = XMV(3) + DXMV
      ERROLD(1) = ERR1
C    Control Loop 2 -- D feed                   D feed flow
      ERR1 = SETPT(2) - XMEAS(2)
      DXMV = 0.0115 * ((ERR1 - ERROLD(2)) + ERR1 * DELTAT * 60./0.1)
      XMV(1) = XMV(1) + DXMV
      ERROLD(2) = ERR1
C    Control Loop 3 -- E feed                   E feed flow
      ERR1 = SETPT(3) - XMEAS(3)
      DXMV = 0.008 * ((ERR1 - ERROLD(3)) + ERR1 * DELTAT * 60./0.1)
      XMV(2) = XMV(2) + DXMV
      ERROLD(3) = ERR1
C    Control Loop 4 -- A&C feed                 A&C feed flow
      ERR1 = SETPT(4) - XMEAS(4)
      DXMV = 4.4 * ((ERR1 - ERROLD(4)) + ERR1 * DELTAT * 60./0.1)
      XMV(4) = XMV(4) + DXMV
      ERROLD(4) = ERR1
C    Control Loop 10 -- Purge Rate/Flow         Purge Valve
      ERR1 = SETPT(10) - XMEAS(10)
      DXMV = 79. * ((ERR1 - ERROLD(10)) + ERR1 * DELTAT * 60./0.1)
      XMV(6) = XMV(6) + DXMV
      ERROLD(10) = ERR1
C    Control Loop 14 -- Prod Sep Underflow      Separator Pot Liquid Flow
      ERR1 = SETPT(14) - XMEAS(14)
      DXMV = 1. * ((ERR1 - ERROLD(14)) + ERR1 * DELTAT * 60./0.1)
      XMV(7) = XMV(7) + DXMV
      ERROLD(14) = ERR1
C    Control Loop 17 -- Stripper Underflow      Stripper Product Flow
      ERR1 = SETPT(17) - XMEAS(17)
      DXMV = 1.35 * ((ERR1 - ERROLD(17)) + ERR1 * DELTAT * 60./0.1)
      XMV(8) = XMV(8) + DXMV
      ERROLD(17) = ERR1
C    Control Loop 19 -- Stripper Steam flow     Stripper Steam Valve
      ERR1 = SETPT(19) - XMEAS(19)
      DXMV = 0.16 * ((ERR1 - ERROLD(19)) + ERR1 * DELTAT * 60./0.1)
      XMV(9) = XMV(9) + DXMV
      ERROLD(19) = ERR1
C    Control Loop 21 -- Reactor Cooling Temp    Reactor Cooling Flow
      ERR1 = SETPT(21) - XMEAS(21)
      DXMV = -5. * ((ERR1 - ERROLD(21)) + ERR1 * DELTAT * 60./0.5)
      XMV(10) = XMV(10) + DXMV
      ERROLD(21) = ERR1
C    Control Loop 22 -- Separator Cooling Temp  Condenser Cooling Flow
      ERR1 = SETPT(22) - XMEAS(22)
      DXMV = -5. * ((ERR1 - ERROLD(22)) + ERR1 * DELTAT * 60./0.5)
      XMV(11) = XMV(11) + DXMV
      ERROLD(22) = ERR1
C
C     Tier II
      CNT = CNT+1
      IF(CNT.GE.30)THEN
      CNT = 0
C    Control Loop 15 -- Stripper level          Stripper Bottom Flow Setpt
      ERR1 = SETPT(15) - XMEAS(15)
      DXMV = -2 * ((ERR1 - ERROLD(15)) + ERR1 *30.* DELTAT*60./120.)
      SETPT(17)=SETPT(17) + DXMV
      ERROLD(15) = ERR1
C    Control Loop 12 -- Separator Level         Separator Underflow
      ERR1 = SETPT(12) - XMEAS(12)
      DXMV = -6. * ((ERR1 - ERROLD(12)) + ERR1 *30.* DELTAT*60./120.)
      SETPT(14)=SETPT(14) + DXMV
      ERROLD(12) = ERR1
C    Control Loop 8 -- Reactor Level            E-Feed Setpoint
      ERR1 = SETPT(8) - XMEAS(8)
      DXMV = 800. * ((ERR1 - ERROLD(8)) + ERR1 *30.* DELTAT*60./120.)
      SETPT(3)=SETPT(3) + DXMV
      ERROLD(8) = ERR1
C    Control Loop 40/41 -- G/H Ratio            D/E feed flow
      ERR1 = SETPT(40) - XMEAS(40)/XMEAS(41)
      DXMV = 0.1 * ((ERR1 - ERROLD(41)) + ERR1 *30.* DELTAT * 60./50.)
      SETPT(41) = SETPT(41) + DXMV
      SETPT(2)=SETPT(41)*XMEAS(3)
      ERROLD(41) = ERR1
C    Control Loop 17 -- Product Flow            A&C feed
      ERR1 = SETPT(27) - XMEAS(17)
      DXMV = 0.1 * ((ERR1 - ERROLD(27)) + ERR1 *30.* DELTAT * 60./50.)
      SETPT(4)=SETPT(4) + DXMV
      ERROLD(27) = ERR1
C     Control Loop 9 -- Reactor Temp            Reactor Cool Temp
      ERR1 = SETPT(9) - XMEAS(9)
      DXMV = 0.375 * ((ERR1 - ERROLD(9)) + ERR1 *30.* DELTAT * 60./20.)
      SETPT(21)=SETPT(21) + DXMV
      ERROLD(9) = ERR1
C    Control Loop 18 -- Stripper Temp           Stripper steam flow
      ERR1 = SETPT(18) - XMEAS(18)
      DXMV = 8. * ((ERR1 - ERROLD(18)) + ERR1 *30.* DELTAT * 60./25.)
      SETPT(19)=SETPT(19) + DXMV
      ERROLD(18) = ERR1
C    Control Loop 7 -- Reactor Pressure         A feed
      ERR1 = SETPT(7) - XMEAS(7)
      DXMV = -0.003 * ((ERR1 - ERROLD(7)) + ERR1 *30.* DELTAT*60./150.)
      SETPT(1)=SETPT(1) + DXMV
      ERROLD(7) = ERR1
C    Control Loop 30 -- Purge B composition     Purge Valve
      ERR1 = SETPT(30) - XMEAS(30)
      DXMV = -0.13 * ((ERR1 - ERROLD(30)) + ERR1 *30.* DELTAT*60./300.)
      SETPT(10)=SETPT(10) + DXMV
      ERROLD(30) = ERR1
C    Control Loop 20 -- Compressor Power        Compressor Recycle valve
      ERR1 = SETPT(20) - XMEAS(20)
      DXMV = 0.036 * ((ERR1 - ERROLD(20)) + ERR1 *30.* DELTAT * 60./20.)
      XMV(5) = XMV(5) + DXMV
      ERROLD(20) = ERR1
C    Control Loop 5 -- Recycle flow             Condenser Cooling Water Flow
      ERR1 = SETPT(5) - XMEAS(5)
      DXMV = 1. * ((ERR1 - ERROLD(5)) + ERR1 *30.* DELTAT * 60./25.)
      SETPT(22)=SETPT(22) + DXMV
      ERROLD(5) = ERR1
      ENDIF
      RETURN
      END
```

```
% This file will try to calculate the minimum cost using my controller
% and MATLAB's built in minimization routine fmins.  This file calls
% minyp.  The results were not nearly as close as the results found
% by a GA

clear all
format short compact
t0=clock;
nn=50;
deltasec=1; deltat=1/3600;
% Initialize plant and variables
[time,yy,yp,xm,xmv,wspace]=teinit_n(nn);  % Initial Guesses
time1=time;

options(2)=1e-4; options(3)=1e-4;
set=[xm(5), xm(7), xm(8), xm(9), xm(12), xm(15),...
     xm(20), xm(30), xm(18), xmv(12)];

setout=fmins('minyp', set, options)
t2=etime(clock,t0)/3600
```

```
function cst=minyp(set)
fid=fopen('d:\users\jsozio\mfiles\sets.dat','w');
fprintf(fid,'%15.12f %15.10f %15.11f %15.11f %15.11f %15.11f %15.11f %15.12f
%15.12f %15.11f \n', set);
fclose(fid);
!d:\users\jsozio\mfiles\teoutb
load d:\users\jsozio\mfiles\cst.dat
cst=cst(5)                    % Just want total cost
```

```
function defuzz3=fuzzpc(f,favg,fmax)

val1=(fmax-90)/(f-90);              % Varies between zero and one
val2=(fmax-90)/(favg-90);           % Varies between zero and one
% Flipped val1 and val2 because max fitness is minimum cost

% Fuzzification
if val1<0.5
      ant1(1)=1-2*val1; ant1(2)=2*val1; ant1(3)=0;
elseif val1>=0.5
      ant1(1)=0; ant1(2)=2-2*val1; ant1(3)=-1+2*val1;
end

if val2<0.5
      ant2(1)=1-2*val2; ant2(2)=2*val2; ant2(3)=0;
```

```
elseif val2>=0.5
      ant2(1)=0; ant2(2)=2-2*val2; ant2(3)=-1+2*val2;
end

% Defuzzification
for i=1:3
    for j=1:3
        A(i,j)=min(ant1(i), ant2(j));
    end
end
A;
Y1 =   [0.7  0.85 1;...
        0.6  0.7  0.85;...
        0.5  0.6  0.7];       % The rules

% weighted average
num3=sum(sum(Y1.*A));
den3=sum(sum(A));
defuzz3=num3/den3;
```

```
function defuzz3=fuzzpm(f,favg,fmax)

val1=(fmax-90)/(f-90);              % Varies between zero and one
val2=(fmax-90)/(favg-90);           % Varies between zero and one

% Fuzzification
if val1<0.5
      ant1(1)=1-2*val1; ant1(2)=2*val1; ant1(3)=0;
elseif val1>=0.5
      ant1(1)=0; ant1(2)=2-2*val1; ant1(3)=-1+2*val1;
end

if val2<0.5
      ant2(1)=1-2*val2; ant2(2)=2*val2; ant2(3)=0;
elseif val2>=0.5
      ant2(1)=0; ant2(2)=2-2*val2; ant2(3)=-1+2*val2;
end

% Defuzzification
for i=1:3
    for j=1:3
        A(i,j)=min(ant1(i), ant2(j));
    end
end
A;
Y1 =   [1.0 1.6 3.0;...
        0.4 1.0 1.6;...
        0.0 0.4 1.0]/100;           % The rules from percentages

% weighted average
num3=sum(sum(Y1.*A));
den3=sum(sum(A));
defuzz3=num3/den3;
```

VITA

John C. Sozio

John C. Sozio was born in Baltimore, Maryland on November 1, 1974. He spent the majority of his youth in the Maryland area where he received his high school diploma from Perry Hall High School in 1992. Once completed high school, he traveled to Virginia to pursue an undergraduate degree in electrical engineering. While working towards his undergraduate degree, he participated in the cooperative education program working three semesters at GE FANUC Automation. Once he received the Bachelor of Science degree from Virginia Polytechnic in May 1997, he had a brief stint working for Black & Decker on motor control projects. The same year he continued towards his Master of Science Degree in electrical engineering. Once completed his degree, John will begin working for Northrop Grumman Systems and Sensors Division in the Baltimore area on submarine control systems.