ARTIFICIAL NEURAL NETWORKS
IN THE SEARCH FOR
AN OPTIMAL CHEMICAL REACTOR SERIES

A Thesis Presented to
The Faculty of the
Fritz J. and Dolores H. Russ
College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

by
Trevor C. Howard
June, 1996

OHIO UNIVERSITY
LIBRARY
# TABLE OF CONTENTS

LIST OF TABLES .................................................................................... iv

LIST OF FIGURES .................................................................................. v

1. INTRODUCTION ................................................................................ 1
   1.1 Background ..................................................................................... 1
   1.2 Reactor Optimization ..................................................................... 3
   1.3 Problem Statement ........................................................................ 5
   1.4 Thesis Outline ............................................................................... 6

2. LITERATURE REVIEW ........................................................................ 7
   2.1 Chemical Reactor Design ............................................................... 7
   2.2 Non-Linear Optimization ............................................................... 9
   2.3 Neural Networks .......................................................................... 13
      2.3.1 Introduction ........................................................................... 13
      2.3.2 Backpropagation Networks .................................................. 15

3. NEURAL NETWORK CONCEPT ......................................................... 18
   3.1 Introduction ................................................................................... 18
   3.2 Backpropagation Neural Networks .............................................. 19
   3.3 Chemical Reactor Process Defined ............................................. 26
   3.4 Neural Networks and Reactor Optimization ............................... 27
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Reactor variable input analysis results</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>Network development results</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Two reactor test problem results</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>Three reactor test problem results</td>
<td>47</td>
</tr>
<tr>
<td>5.3</td>
<td>Five reactor test problem results</td>
<td>47</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1.1   Polyethylene Terephthalate production process ........ 2
Figure 3.1   Typical backpropagation artificial neural network .... 21
Figure 3.2   Reactor and neural network variable arrangements .... 29
Figure 4.1   Data histogram example ................................ 35
Figure 4.2   Flowchart of search heuristic ............................ 44
CHAPTER 1: INTRODUCTION

1.1 Background

Many unique types of plastics are fabricated in today’s modern manufacturing facilities, with much attention recently turning to the common, yet highly valued plastic, polyethylene terephthalate (PET). Plastics are made up of individual molecular units known as monomers, a repeating subunit composed of differing elements based on the type of plastic. The resulting chain, which may include thousands of monomers, is termed a polymer. The final average overall length of the polymer, i.e. the number of monomers forming the chain, is known as the degree of polymerization (DP). There is ordinarily a target degree of polymerization for a specific plastic based on designated product requirements.

In addition to the degree of polymerization, a diethylene glycol (DEG) content is calculated as a basis of determining the quality of the polyethylene terephthalate. Diethylene glycol is formed during the side reactions within the process and is the major impurity within polyethylene terephthalate, causing decreases in the melting temperature of the finished plastic. The DEG content is a value based on this unwanted chemical species mole percentage found in the plastic. The production goal is to keep the DEG content below a target level, with a low range of mole percentage values typically considered acceptable.
As shown in Figure 1.1, the overall PET production process begins when two ingredients are fed into an initial reactor of specific volume, temperature, and pressure. As these ingredients mix under the specific conditions, a partial reaction occurs between the constituents towards the final overall goal, i.e. the degree of polymerization is increased. From this reactor, the eight output ingredients then flow into a second similar reactor of specific volume, temperature, and pressure. Here, a similar process takes place in that the ingredients react further towards the goal state. This overall process continues typically for one to four more reactors until the point where the final product reaches the necessary predetermined degree of polymerization specifications, which in the test problems for this research were 80 and 120.

Figure 1.1: Polyethylene Terephthalate production process
1.2 Reactor Optimization

Traditional approaches to reactor design use handbooks, prepared tables of values, and design equations to determine the appropriate reactor parameters to achieve the desired material results. These approaches do not optimize; rather, they simply achieve the desired end results. When optimal reactor parameters have been investigated, the analysis has been for the most part restricted to single reactors within a series of reactors.

Chemical reactor optimization is a complex procedure, involving a wide variety of chemical species, their interactions, and the various means of evaluation. Typical models relating to this research area use high-level mathematics and take weeks to develop. Artificial neural networks provide an alternative approach to the usual reactor optimization procedures through their inherent learning abilities, bypassing the time-consuming mathematical formulations. This technology avoids repeating the complex calculations required in traditional non-linear optimization procedures. In this research, through modifications to an existing chemical reactor optimization program a new method was attempted to determine a minimum cost sequence of chemical reactors.

Eastman Chemical Company is an organization which has taken interest in optimizing the high volume production of polyesters, including PET. The Chemical Engineering Department at Ohio University recently undertook a project in cooperation with Eastman Chemical Company to determine the specifications of a series of reactors which would minimize the overall system cost [Pattalanchinti, 94]. The cost function included both the cost of installed capital and the operating costs for the reactor series.
A FORTRAN 77 program used detailed cost equations, reaction models, and a non-linear search routine to determine the minimum cost sequence of reactors within the physical constraints imposed by the system. The constraints consisted of the product characteristic requirements on the degree of polymerization and DEG content versus their user-specified target values along with seven material balance constraints per reactor. The program conducted the constrained optimization line search along a reduced gradient to find an optimal sequence of reactors and utilized the Newton-Raphson method for solving the non-linear equations. An initial solution guess to the search variables had to be given for each replication due to the non-linearities in the system, which introduced the possibility of local optimality. More recent research by Calmeyn [1995] reduced the number of constraints from nine to two and incorporated a more accurate cost function. The results achieved by this system appear to be relatively accurate and provide a generally good means of searching for an optimal chemical reactor series.

An alternative method of finding an optimal cost for a series of reactors is through the use of artificial neural networks. Through modifications to the existing FORTRAN program, data was collected relating to the chemical reaction processes within single reactors in series. This data was then used to develop and train backpropagation artificial neural networks to create a model capable of generalizing to new circumstances. Reactor series optimization utilizing artificial neural networks was a new approach that showed the potential for solving this large problem in a more direct and innovative manner. In this approach it was possible to arrange the input and output
variables in a manner that completely eliminated the constraints to the original chemical reactor optimization problem, thereby simplifying the overall non-linear optimization to an unconstrained search. Neural networks fast computational abilities permitted a large number of trials, and therefore a greater portion of the solution space, to be explored for a global optimum.

1.3 Problem Statement

The objective behind this research project is to utilize the unique attributes of artificial neural networks to provide an innovative and superior approach for determining a minimum cost chemical reactor series. Past research has focused on modeling entire reactor systems within a single FORTRAN program. This research differs in that it models a single reactor at a time using artificial neural networks, and then links the individual reactor models together into a single overall system using a shell program. The neural networks in this research are trained and tested using data collected from a modified chemical reactor optimization FORTRAN program. The input and output variables of the networks are arranged in a manner that eliminates the constraints to the original problem. An initial RMS error goal of 0.01 was set as a maximum network training error level to achieve reliable results. Once system development is complete, a search is performed to identify the solution to the optimal chemical reactor series. Finally, a comparison is made to results in previous research.
1.4 Thesis Outline

The first chapter of this thesis gives a brief introduction to chemical reactor optimization.

Chapter two reviews the pertinent literature to the subject of this research. The first section describes how chemical reactors are currently designed and analyzed in real-world applications. This chapter then describes chemical reactor optimization using constrained non-linear optimization search techniques. The final section discusses the general attributes of neural networks, the various types of neural networks, and the reasons for choosing the specific neural network used.

Chapter three details this project's artificial neural networks in the optimization search. The single reactor neural network and optimal reactor series search approach is described. The simplifications and improvements over previous approaches are identified.

The fourth chapter discusses the specific neural network training and testing. Modifications to the existing FORTRAN optimization program for data collection are outlined. The preliminary network testing, the neural network development, results, and the shell program development are detailed.

The test results of the neural network reactor optimization system are summarized and compared to previous results in chapter five.

Chapter six contains the general results, suggested future work in this research area, and an overall evaluation of this research project.
CHAPTER 2: LITERATURE REVIEW

2.1 Chemical Reactor Design

Reactor design is a scientific field that has been proposed as having an infinite number of physical situations [Fogler, 92]. The practitioners within this field require a very general set of tools which may be applied to the many unique and complex situations encountered.

To this degree, the traditional means of determining the reactor parameters in a chemical production process have been through previously tabulated data, existing equations, and experimentation. Data sets relating to various chemical constants and rates have been studied and documented for use by the chemical engineers. These sets of data include the rates at which chemical reactions take place, how specific materials behave together, the effects that the concentration of a species has on a reaction, etc. [Fogler, 92]. Many general chemical equation forms have also been developed and applied, including mole balance equations, stoichiometric equations, and design equations to determine the design variables of a reactor given the overall objective of the analysis. Pilot plant experiments define the interactions of different chemicals and verify existing assumptions.

These different forms of identifying the necessary reactor parameters must be adaptable to the various situations encountered. These differences may arise through distinct reactor types, different chemical species, and through changes in the objective
of the overall process. Therefore, traditional reactor design is well suited for achieving
the primary goal of meeting product specifications in the production process, but lacks
the ability to determine the parameter values which will optimize the overall process.

Recent approaches to optimizing reactor series have been attempted in the
works of Pattalachinti and Calmeyn. In Pattalachinti's research, a FORTRAN program
was developed which searched through the solution space using a generalized reduced
gradient method and used the Newton-Raphson method of solving the non-linear
constraint equations. However, a large set of variables and constraints, a program
execution time that may take up to several minutes, and discontinuities in the cost
function inhibited this solution approach from achieving accurate and reliable results.
In some instances, the volume and pressure variables' final values were out of the range
normally considered acceptable in reactor design. An additional issue was a problem of
inconsistency, where the degree of polymerization of the product and the specified
reactor type were not compatible. Calmeyn improved Pattalachinti's model by
smoothing the discontinuities in the cost function and by eliminating seven of the nine
constraints. The constraints were eliminated by presolving for the constrained variables
within the objective function. Program execution time was also decreased through
these changes and the results appeared reliable. The solution approach attempted using
the neural networks optimization system was a parallel approach to the work of
Calmeyn. This approach eliminates all of the constraints, provides an even faster
program execution time, and furthermore was thought to smooth the discontinuities in
the objective function.
2.2 Non-Linear Optimization

Non-linear systems' functions can be difficult, if not impossible, to solve for a global optimal solution. There are a variety of different mathematical formulations that require one of many different methods of solving the non-linear functions. A number of methods are discussed by Hillier and Lieberman [1990], Simmons [1975], Biles and Swain [1980], and Kuester [1973].

Nonlinear programming problems are typically solved either with specific numerical methods or through nonlinear search techniques. The numerical methods often fit a limited problem form, but provide a direct means of solving for the unknown search variables. The nonlinear search techniques typically are more general in form and fit a larger set of nonlinear problems.

The nonlinear problems may be grouped into one of a several different categories. The most simple form is an unconstrained nonlinear problem. This problem has a nonlinear objective function with no constraints and can be either a single or multi variable case. Common solution approaches to this type of problem include the bisection search algorithm [Ecker, 91] and the Fibonacci search algorithm [Cooper, 70] for single variable cases, the Hooke and Jeeves search algorithm, the Fletcher and Reeves search algorithm, and the gradient search algorithm for multi variable cases [Kuester, 73], or through direct numerical solution approaches of solving a system of equations obtained by setting the partial derivatives equal to zero [Hillier, 90].
Another form of the nonlinear problem is the constrained nonlinear problem. This problem may contain a nonlinear objective function with linear constraints, a linear objective function with nonlinear constraints, or both nonlinear objective function and constraints. Common search approaches to this form of the problem include the Fibonacci search algorithm for single variable cases and the Box search algorithm and the Constrained Fletcher-Powell search algorithm for multi variable cases [Kuester, 73]. A very common optimization algorithm within the chemical engineering field is the Successive Quadratic Programming algorithm [Horst, 1995]. Numerical methods include the specialized convex programming algorithm and separable programming algorithm [Hillier, 90]. For problems with both nonlinear constraints and objective function, global optimality can typically only be guaranteed in those cases where the objective function is being maximized and is concave or where the objective function is being minimized and is convex.

The Generalized Reduced Gradient method (GRG) method of solving non-linear optimization problems is a very popular iterative solution approach that was used by Pattalachinti. This algorithm takes a non-linearly constrained optimization problem and, through substitutions, changes the problem into a non-linear, unconstrained optimization problem. The algorithm approximates its non-linear constraints as linear constraints through Taylor series expansion, dropping the terms in the expansion beyond the linear terms.
Given a set of $n$ equations:

$$f_i(x) = 0$$

$$i = 1, \ldots, n$$

$$x = (x_1, \ldots, x_n)$$

The Taylor expansion, dropping nonlinear terms, is

$$f_1(x_0) + \left(\frac{df_1(x_0)}{dx_1}\right) Dx_1 + \left(\frac{df_1(x_0)}{dx_2}\right) Dx_2 + \ldots + \left(\frac{df_1(x_0)}{dx_n}\right) Dx_n = 0$$

$$f_2(x_0) + \left(\frac{df_2(x_0)}{dx_1}\right) Dx_1 + \left(\frac{df_2(x_0)}{dx_2}\right) Dx_2 + \ldots + \left(\frac{df_2(x_0)}{dx_n}\right) Dx_n = 0$$

$$\ldots$$

$$f_n(x_0) + \left(\frac{df_n(x_0)}{dx_1}\right) Dx_1 + \left(\frac{df_n(x_0)}{dx_2}\right) Dx_2 + \ldots + \left(\frac{df_n(x_0)}{dx_n}\right) Dx_n = 0$$

$$x_0 = \text{starting solution}$$

The number of variables in the problem is then reduced by solving for a subset of the variables in terms of the remaining variables. The constraints to the problem are eliminated by substituting the reduced constraint equations into the objective function. The new form of the objective function is then either solved analytically by solving the partial derivatives of the objective function in terms of the variables or through a numerical means such as gradient descent. In the analytical solution approach, the new solution is then used as the starting solution for the next iteration of the algorithm. The iterative search process continues until a defined convergence criteria is met, giving at least a local optimal solution.
A popular technique for solving simultaneous nonlinear equations is the Newton-Raphson method, in which the solution is determined through numerical iterations. A Taylor series expansion is performed on the partial derivatives of the objective function around a selected starting point. These equations are then simultaneously solved for the new solution $x_I$, which should represent a better solution set than the previous set. This process is continued until the difference between two successive solution sets converges to a reasonably small value. This final solution should be a fairly accurate estimate of the true solution.

This algorithm was used by Pattalachinti [94] to solve a set of nonlinear equations to reach a feasible solution. The generalized reduced gradient algorithm would search along linearized constraints until a minimum was found, which in all cases would be an infeasible solution. At that point, the Newton-Raphson method would determine a feasible solution within the vicinity of this minimum.

The selection of the method for solving a non-linear programming problem is almost always dictated by the general form of the objective function and constraints. The chemical reactor optimization problem studied within this research is a nonconvex programming problem with both nonlinear constraints and objective function, and represents the most difficult type of optimization problem. Although a global optimal solution cannot be determined in problems that have a concave objective function with concave constraints (or vice versa), a very good local optimal solution can nearly always be determined.
2.3 Neural Networks

2.3.1 Introduction

In their ability to mimic the human learning process, artificial neural networks have recently gained much attention in the research community. Never before has any form of artificial intelligence been able to learn from experience and adapt in a way that enables itself to generalize to new data. With this newfound ability to imitate the human neural process, an abundance of inspirations have come forth on how this technology may be applied. Examples of how neural networks have been applied to real-world situations include stock performance modeling [Refenes, 94], signal processing [NeuralWare, 93], handwritten character recognition [Gallant, 93], and others. In chemical processes, neural networks have in many instances modeled data to a lower error level than their regression model counterparts. [Refenes, 94] Neural networks have been utilized within the chemical industries for long-term predictions of chemical processes [Su, 92], PVT data analysis [Normandin, 93], and chemical pattern recognition [Borman, 93]. However, there is no published research on neural networks in the optimization of chemical reactors. The potential for benefits to the reactor design process through the use of neural networks creates the demand for such an application.

An advantage of artificial neural networks to this project is their fast computational abilities. Neural networks use a direct set of forward and backward pass numeric calculations which greatly reduces the amount of time required for program execution.
There exist two main types of training paradigms in neural networks: supervised and unsupervised [Gallant, 93]. Supervised learning has a desired set of outputs corresponding with each set of inputs shown to the network through which network adjustments may be determined and evaluated. Unsupervised learning, on the other hand, does not have any target outputs for the network; the most that can be expected of such a network is the grouping of similar data. The nature of this project mandated the use of supervised learning since the intent of the neural networks was to map a given set of inputs to the desired outputs. It has also been shown that supervised learning is sufficient for nearly all chemical engineering problems [Chitra, 93].

There are different types of supervised-learning artificial neural networks, such as the perceptron network and the backpropagation network. The most appropriate type of neural network for this chemical reactor design project is the backpropagation network; this type of network is often used in many chemical engineering applications [Chitra, 93]. The backpropagation network allowed multiple hidden layers within the network for more accurate higher order function input-to-output mapping. The multiple hidden layers turned out to be a requirement for this project to allow better network performance. Backpropagation networks were used to allow the non-linear chemical relationships within the complex reactor systems to be modeled to a lower error level than should be possible by other types of artificial neural networks due to the ability to have multiple hidden layers.
2.3.2 Backpropagation Networks

Backpropagation networks are currently the most common and renowned type of neural networks in use. They have attained high recognition for their ability to modify the weights associated with hidden layers in networks and their fast execution speeds, and “... quickly became the model of choice by many researchers” [Gallant, 93, p. 26]. In simplistic terms, a backpropagation network is an elaborate regression model. When given a set of inputs, a comparison is made between the network outputs and the desired outputs, and a corresponding change in the weight set is made to reduce this difference. The neural network cycles through an entire set of these input-output combinations until no further reduction in the error is possible. At this point, the network is tested against new data in the testing set and an overall determination of network ability is evaluated.

Backpropagation networks are feedforward networks which may have either discrete or continuous inputs or outputs. The continuous outputs to a backpropagation network must be scaled to a range of 0 to 1 or -1 to 1, depending on the network transfer function. Due to the chemical reactor's continuous inputs and outputs within this project, continuous variables were used within the network and were scaled between 0 and 1. In networks where the data is continuous, the data should be evenly distributed within the variable's range to ensure that the input variables accurately reflect the changes in the output variables [Niu, 91]. The learning rate, momentum, and weights are other network parameters which must have initial values assigned to them for fast and accurate network development.
The means of evaluating the various networks was the overall network Root-Mean-Square (RMS) error. This evaluation function is an objective measure of how much the actual network outputs varied from the desired network outputs, and therefore "... is a measure of correctness of prediction" [Refenes, 94, p. 379]. Therefore, as better network architectures were determined and as the networks trained to more accurate levels, the RMS error decreased.

Within neural networks there exist many different types of learning rules. These include the Pocket Algorithm, linear programming algorithms, and the General Delta Algorithm. The learning used by backpropagation networks is a gradient descent algorithm similar to the general delta rule algorithm, with the exception that the delta rule operates on a single-cell network and backpropagation networks most often have many processing elements within the network. The gradient descent algorithm looks at the derivative of the error within the output layer cells and then passes this error back through preceding layers of processing elements, making step-size changes to the weights to minimize the error. This algorithm is necessary for updating weights within multiple hidden layers and is a key element of the backpropagation neural network.

A requirement for the gradient descent learning algorithm is that the processing elements must have differentiable activation functions, and therefore must be continuous. Several activation functions meet this prerequisite, such as the hyperbolic tangent and the sigmoid functions. However, for most backpropagation networks, this function should be the sigmoid. [Chitra, 93] The sigmoid function is a continuous squashing function with a range between 0 and 1.
Backpropagation artificial neural networks have no requirements on the arrangement of the inputs and outputs to the network. The inputs and outputs to the network may be rearranged in a manner not possible with more traditional statistical regression models. The variable rearrangement proved to be of great significance since it eliminated the constraints to the chemical reactor problem. This result permitted a much more simple and direct non-linear search.

Based on the above, the backpropagation network was chosen as the type of neural network for this chemical reactor design project. This type of network is essentially a regression model capable of storing the complex non-linear chemical reactor process functions within the internal network weights. It provided execution speed, network accuracy, and the rearrangement of the inputs and outputs to the problem. These all contributed to a much more direct approach to finding a global minimum cost function to the reactor series problem. All of these ideas indicated that the backpropagation artificial neural network could provide many attractive features not possible through other more standardized approaches.
CHAPTER 3: NEURAL NETWORK CONCEPT

3.1 Introduction

Neural networks are a relatively recent technological innovation possessing properties which lend themselves to the unique manipulation of data for many applications. The ability to evolve a network to meet the needs of a particular situation made this technology an ideal tool for investigating chemical reactor processes. Neural networks are analytically flexible, computationally fast, and do not need a prespecified form to the equations. They are essentially a 'black box' regression model, where the mathematical relationships are stored within the network parameters. Due to a rather simplistic computational sequence, the execution time associated with any particular network is typically only a few seconds or less. The combination of these neural network properties, along with a vision of how they could interact with other programming tools, were the main determinants in utilizing this technology to search for an optimal chemical reactor series.

The initial idea behind using artificial neural networks in the search for a minimum-cost sequence of reactors was to simplify this whole reactor process by breaking the overall problem down into smaller components. By using the neural networks, it was possible to decompose this large optimization problem into single reactor sections. A program was then developed to combine these smaller subsections in a manner to solve the overall problem. Each reactor was evaluated separately by the
neural networks, with the overall system combined together to view the final results of
the system.

Neural networks' fast computational abilities also help in searching through the
solution space. Even the largest of neural networks typically take a second or less to
run when developed and trained properly. The FORTRAN program execution time
limited the number of reasonable trials that could be attempted, which in turn limited
the coverage of the solution space in trying to locate the global minimum. With the
short processing time of neural networks, this limiting factor was overcome.

The biggest advantage to using artificial neural networks in this project was in
the ability of the networks to allow the rearrangement of the variables associated with
the chemical reactor processes in such a manner as to eliminate the constraints to the
problem. Neural networks allow any input-output variable arrangement as long as the
solution is completely defined. Two output variables were moved to input variables
along with a single corresponding reactor variable, eliminating the constraints to the
problem and thereby simplifying the overall problem to an unconstrained search.

3.2 Backpropagation Neural Networks

Artificial neural networks have been developed from the manner in which the
human body synthesizes data. The basic element of neural networks is the processing
element, analogous to the neuron in the human body. Entering the processing element
on one side and exiting from the other are inter-connecting links containing weight
fields, representing the axons and dendrites of the human neural system. Information is presented to the network from input cells, is acted upon and is passed locally from one processing element to another through the weight fields; a final solution is determined at the output layer of processing elements.

In such a network, a training set of data containing input-output examples feeds the network the necessary information for the learning process. A given set of inputs from the training set is presented to the network. These values are modified in the layers of processing elements, passed along the links containing the weights, and ultimately produce a set of outputs. When the final outputs are produced, a learning rule then modifies the weights of the network in a backward pass in such a manner as to reduce the overall error, i.e., learning from the given data. The final result is a network that should be able to generalize to new data relating to the same process. A testing set of input-output examples is then used to determine the network’s ability to generalize to the new data. Each network, however, is application-specific; there is no guarantee that every application will be able to produce a network with a low enough error or that will be able to generalize well enough to model new data.

The processing elements in the network perform three very important functions. First, they collect the data coming from either the inputs or other processing elements. The processing elements are created in layers; each layer of processing elements typically feed the successive layer the required data. The first layer of processing elements is the input layer, the last layer of processing elements is the output layer, and any layer in between these are hidden layers. Once the data is collected by the
processing element, it is transformed based on an activation function that is common to all of the processing elements. The exception is the input layer of processing elements, whose main task is to distribute the inputs to the processing elements in the second layer. All of the inputs to a given processing element are summed together and passed to the activation function, which takes the sum and transforms it to a new value. Once the activation function transformation is complete, the processing element then takes the transformed value and passes it along each of the output links to other processing elements. Backpropagation is a feedforward network, so these links are to processing elements further towards the output layer. The processing elements then perform these same functions until the output layer of processing elements produces the final network output.

Figure 3.1: Typical backpropagation artificial neural network [Chitra, 93, p. 45]
As the outputs of the processing elements are passed along the links, the outputs are multiplied by the corresponding weights. The weights may be either positive or negative values, thereby either adding to a processing element's summation or taking away from it. When the entire network has been evaluated, a learning rule modifies the weights by an incremental amount in such a manner as to reduce the overall network error. Therefore, an entire single forward pass through the network is made to determine the network outputs and then a corresponding backward pass is made to adjust the weights within each layer.

The equations and variables required for the network gradient descent learning rule are as follows [Gallant, 93]. First, each processing element computes its activations in a forward pass through the network by summing weighted inputs and using an activation function to calculate the processing element output as follows:

\[ S_i = \sum_{j} w_{ij} u_j \]  

\[ i = 1 \ldots N_r \]

\[ j = 1 \ldots N_e \]
\[ u_i = f(S_i) = \frac{1}{1 + e^{-S_i}} \] \hspace{1cm} (3.2)

- \( S_i \) = summation into processing element \( i \)
- \( u_i \) = output of processing element \( i \) using sigmoid activation function
- \( w_{ij} \) = weight from element \( i \) to element \( j \)
- \( N_r \) = No. of processing elements in current row
- \( N_p \) = No. of processing elements in preceding row

The sigmoid function, a very common activation function, is defined as

\[ f(x) = \frac{1}{1 + e^{-x}} \] \hspace{1cm} (3.3)

- \( x \) = evaluation point for sigmoid function

The derivative of the sigmoid function is calculated as follows:

\[ f'(x) = \frac{d}{dx}(1 + e^{-x})^{-1} \]

\[ = (-1)(1 + e^{-x})^{-2}e^{-x}(-1) \]

\[ = \frac{1}{(1 + e^{-x})^2}(1 - \frac{1}{1 + e^{-x}}) \] \hspace{1cm} (3.4)
Through substitution, we have the following identity:

\[ f'(S_i) = u_i(1 - u_i) \quad (3.5) \]

\[ i = 1 \ldots N_e \]

Then, for determining the error associated with any processing element, we have

\[ \delta_i = (C_i - u_i)f'(S_i) \quad i = 1 \ldots N_r \quad (3.6) \]

for the output layer processing elements,

\[ \delta_i = (\Sigma_{m:m>i} w_{m,i} \delta_m)f'(S_i) \quad i = 1 \ldots N_r \quad (3.7) \]

for all other processing elements,

where: \( C_i \) = desired output for output cell \( i \)

\( \delta_i \) = error associated with cell \( i \)

The final step is to make the weight changes to the processing elements in the network to form the new weight set \( w^* \). The general delta rule has the following form:

\[ w^*_{ij} = w_{ij} + \rho \delta_i u_j \quad (3.8) \]

where: \( \rho \) = learning rate
This entire procedure is cycled through with each example within the training set until the point where the incremental changes to the weights ($\delta$) converge to a user-defined steady-state. At that point, the network has 'learned' as much as possible from the given training set. The network is evaluated at specified points in this cycle using the RMS error criterion. The RMS error compares the actual output to the desired output for each output processing element and is defined as follows:

$$\text{RMS Error} = \sqrt{\frac{1}{n} \sum (X_{i, \text{act}} - X_{i, \text{des}})^2}$$  \hspace{1cm} (3.9)

where: $X_{i, \text{act}} = \text{actual output for cell } i$

$X_{i, \text{des}} = \text{desired output for cell } i$

$n = \text{number of samples}$

It is also common to test a network against new data to check for its generalization abilities. The testing results will typically have a somewhat higher RMS error level than the training RMS error level, but there will typically not be more than a few percentage points difference in a properly trained network.

The development of a backpropagation network is a directed trial-and-error procedure, i.e., there are no specific rules for network architecture based on an application. The most common method of developing a network is to start with a small, three layer network (1 hidden layer), train this network until the weights converge, and
then decide if this network provides the desired results (low enough RMS error). If it
does not, increasing the size of the single hidden layer is the next step. If a large single
hidden layer network still does not provide good enough results, a second hidden layer
should be added to the network. Various size combinations between the two hidden
layers should be investigated until the best network architecture is achieved. Most
applications will only require a single hidden layer; however, two hidden layers may be
required for more intense regression applications

3.3 Chemical Reactor Process Defined

The intuitive way of arranging the network inputs and outputs is in the same
form in which the various variables occur within the natural chemical process, as was
shown in Figure 1.1. In this case, eight ingredients have specific flowrates into a
reactor along with a corresponding mixture temperature. The reactor itself has four
variables associated with it: the reactor volume, the reactor temperature, the reactor
pressure, and the overall reactor cost. The ingredients react within the vessel, creating a
specific degree of polymerization and DEG content based on the altered ingredient
flowrates. When the chemical processes are completed, the eight altered ingredient
flowrates exit the reactor at the temperature of the reactor. Many non-linear constraints
exist within the natural mathematical formulas following such a process. These include
seven material balance constraints, the degree of polymerization constraint, and the DEG content constraint for each reactor within the series.

3.4 Neural Networks and Reactor Optimization

Neural networks allowed the separation of these reactor variables into inputs and outputs in any arrangement, as long as the solution was completely defined in the inputs. Consequently, alternative variable arrangements dealing with the chemical processes were considered. One specific external network configuration eliminated the constraints associated with the chemical processes, shown in Figure 3.2. The inputs were the target degree of polymerization, target DEG content, the ingredient flowrates, and one of the three reactor variables (temperature, pressure, or volume). The corresponding outputs were the cost of the reactor, the two remaining reactor variables, and the output ingredient flowrates. By defining the degree of polymerization and DEG content as inputs, all of the constraints to the original problem are eliminated, thereby simplifying the overall search process. The search variables for the reformulated problem are the product degree of polymerization, the DEG content, and the reactor volume for each reactor. Since this reformulated problem was unconstrained, these variables were able to take on any reasonable values. The input variables for reactor 1
Variable Arrangements

reactor cost
= f(V, T, P)

8 input flowrates
input temp.

8 output flowrates

volume temp. pressure

DP, DEG = f(outputs)

Reactor Variables

inputs

8 input flowrates
input temp.
reactor volume
DP
DEG content

outsputs

8 output flowrates
reactor pressure
reactor temp.
cost

Neural Network Inputs and Outputs

Figure 3.2: Reactor and neural network variable arrangements
were read from an input file and the input variables for the other reactors are carried
over from the outputs of the previous reactors in the series.

The final step in using a neural network for reactor optimization was to
determine which of the three reactor variables was required to completely define the
solution. Through a process known as dithering discussed in section 4.4, initial
conclusions indicated that the reactor pressure was the final required variable; however,
through later experimentation, this was changed to the reactor volume.

3.5 Conclusion

In brief, neural networks enabled the larger problem of finding an optimal
chemical reactor series to be decomposed into single reactor sections. The extremely
fast computational abilities of neural networks permitted a search of a greater portion of
the solution space while allowing different arrangements of input and output variables.
The variable arrangement eliminated the non-linear constraints, which became the core
concept behind this project. The next necessary step was to find the best neural
network and architecture for PET production.
CHAPTER 4: NEURAL NETWORK DEVELOPMENT

The development of the neural networks for the purpose of finding a minimum-cost sequence of reactors included training and testing set development, data analysis, actual network development, and shell program development steps.

4.1 FORTRAN Program Modifications

Generating the training and testing sets for the neural networks was the first phase of this project. Data was collected by making modifications to the existing FORTRAN program in its single reactor state. With a few minor changes to the program code it was possible to collect the required training and testing data sets.

The original version of the program had an input file, ‘grgr.inp’, which contained all of the necessary variables and constants required by the solution procedure. Included in this input file was the target degree of polymerization and target DEG content for the final reactor, the volume of the reactors, and other required variables and constants. Program execution would generate a solution which consisted of the required reactor temperature and reactor pressure for each reactor in series. However, in order to efficiently develop training and testing sets that covered the range of all variables, it was necessary to directly specify the reactor volume, temperature, and pressure in the input file for a reactor. The FORTRAN program solution procedure used a series of alternating reduced gradient line searches and Newton-Raphson
searches to find an optimal solution. The initial basis included nine dependent variables consisting of the initial guesses of the reactor temperature and eight material flowrates. The ingredient 'FG' was the main factor in determining the pressure within a reactor, and thus for the purposes of this project both the temperature and pressure were considered dependent variables. The reactor volume, however, was an independent input variable. There were nine constraint equations consisting of seven material balance constraints, the target degree of polymerization, and target DEG content. The heart of this program was a loop in which the line search function was repeatedly called to find an optimal solution using the Newton-Raphson method. Once the output properly converged (if possible), the solution variables were written to the output file 'grgr.out' and the program was exited.

As stated, the goal of the FORTRAN program modifications was to allow the user to fix the volume, temperature, and pressure of a reactor and have as a portion of the solution set the degree of polymerization, the DEG content, the output flowrates for the ingredients, and the overall cost. The pressure was held constant fixing the flowrate of ingredient 'FG' as an input to the model. Two product characteristic constraints to the problem, the target degree of polymerization and the target DEG content, were removed from the system of equations. This allowed three of the dependent variables to be changed to independent variables, thus allowing their values to be fixed prior to program execution. These variables were the reactor volume, the reactor temperature, and the 'FG' material flowrate relating to pressure. With this altered mathematical formulation, the three reactor variables' values were able to be user chosen, and
allowed a wide variety of reactor conditions to be modeled. Due to the excellent design of the program, a minimal number of changes were required to alter the program for our purposes.

The first change to the program was to set the 'kflag' variable to '0' in the input file, thereby eliminating the call to the optimization portion of the FORTRAN program. The target degree of polymerization, the target DEG content, and the penalty factor inputs were still read in to the program through the ‘Input’ subroutine; however, they were not used since these constraints were removed from the solution procedure. The number of dependent variables (ND) was reduced by two since the temperature and 'FG' flowrate were no longer in the basis.

The 'Constraints' subroutine, responsible for calculating the error in the material balance constraints, was changed by removing the section that calculated the deviation from the desired degree of polymerization and DEG content of the last reactor. Once again, this change was necessary since the degree of polymerization and DEG content were no longer constraints in the solution procedure. Following this in the same manner were the modifications in the subroutine 'Jacobian, which is used to calculate the analytical derivatives for the Jacobian matrix used in the Newton-Raphson search'. Here, the two segments which included the degree of polymerization and DEG content in the search matrix were removed.

With the modifications complete, test runs were performed and compared to previous program outputs to validate the results. The inputs and outputs of the modified program appeared to parallel the results obtained in the original program,
validating the modified program. Appendix A shows the modified FORTRAN program.

4.2 Training Set Development

Single reactor model data collection for the training and testing of the neural networks was the next main step. The main variables of interest were the reactor volume, temperature, and pressure, since these variables had to be varied in such a way that as many of the feasible variable combinations were covered to provide for a robust model. Each variable range was defined, a discrete set of values were chosen for each variable, and the characteristic progression of these values from reactor to reactor was outlined. The reactor volumes used were 5, 10, 20, 30, and 40 cubic meters, and there was no set volume pattern from reactor to reactor. The reactor temperature was allowed to vary between 500 and 600 degrees Fahrenheit, and for the most part increased with successive reactors. Finally, the 'FG' flowrate, relating to the reactor pressure, was varied between 1.0 and .00001 kgmole/hr, and these values typically decreased an order of magnitude with successive reactors. Initial ingredient data was provided by an input file, and for successive reactors the inputs to the reactors were the outputs from the preceding reactors. The data collected covered enough reactors in series such that the goal degree of polymerization of 120 was attained.

With these general rules in mind, the modified FORTRAN program was run 504 times to collect samples with which to train the two neural networks. The first network
used one hundred data examples and the other network used the remaining data examples for training and testing. All of the results were sent to individual output files generated by the FORTRAN program.

4.3 Data Analysis

With all of the necessary data interspersed with other non-needed data in the output files, a C program (Appendix B) was created which separated the required data into another file in the proper format for analysis purposes. The data was analyzed for any characteristics that might have influenced the training of the neural networks. Histograms were chosen as the best means of displaying the data graphically. Each variable’s data was first separated into individual files by means of a C program, and the minimum and maximum values associated with each variable were determined. Each data set was plotted as a histogram with ten equal increment groups. The histogram for ingredient ‘ZG’ is shown in Figure 4.1 and the others are in Appendix C. It was obvious that the first reactor in the series had significantly different variable values than the rest of the reactors, leading to bimodal data patterns. Those data points that originated from the first reactor in the overall series most often had extremely low values, whereas the data points from any of the other reactors in series were likely to have higher values. There were very few, and most often no variables at all in between these high and low ranges. The values must be fairly evenly distributed throughout the
range of the variable in order for the neural networks to train properly. With this big division between the low values of the first reactor and the high values associated with the rest of the reactors, it was concluded that two networks would be most appropriate to model the reactors. Thus it was decided that the first reactor in series would have a neural network of its own, and the rest of the reactors in series would be represented by a second neural network.

Figure 4.1: Data histogram example
4.4 Network Investigation

NeuralWare [1993] was chosen as the neural network software used in this project. At this point, it was necessary to determine experimentally which of these three variables was the necessary final input to completely define the solution. Initially, all three of these variables were placed as inputs to a network along with all of the other inputs, creating a network with fourteen inputs and nine outputs. An initial network was trained using this external setup to an RMS error of 0.07. The neural network software then allowed a procedure known as dithering [NeuralWare, 93], where the inputs to a network were varied within a defined percentage range in order to determine the effect that each input has on the outputs. With this feature, each of the three reactor variables were dithered between 10% and 75% of their range. The Root-Mean-Square error was calculated for each of the three reactor variables and compared to determine which of the variables had the greatest affect on the outputs. Based on Table 4.1, the reactor pressure had the most influence on the outputs of the network, and this variable was therefore initially chosen as the final neural network input. However, later training of actual networks revealed through trial-and-error that the reactor volume was actually a superior input to the network, and this change was therefore incorporated into the project.
Table 4.1: Reactor variable input analysis results

<table>
<thead>
<tr>
<th>% DITHER</th>
<th>REACTOR VOL.</th>
<th>REACTOR PRESS.</th>
<th>REACTOR TEMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>3.4%</td>
<td>56.5%</td>
<td>8.0%</td>
</tr>
<tr>
<td>25%</td>
<td>9.8%</td>
<td>45.2%</td>
<td>5.4%</td>
</tr>
<tr>
<td>50%</td>
<td>4.2%</td>
<td>37.4%</td>
<td>3.7%</td>
</tr>
<tr>
<td>75%</td>
<td>3.5%</td>
<td>33.5%</td>
<td>4.8%</td>
</tr>
</tbody>
</table>

4.5 Network Development

With all of the preliminary information gathering and analysis completed, the actual neural network development was begun using the backpropagation paradigm with the sigmoid transfer function and the general delta learning rule. An initial training set size of 100 samples and a testing set size of 100 samples was used for initial network parameter estimation. A simple neural network was developed to investigate some of these network parameters.

There are various network parameters which influence the training procedure and ability of the backpropagation networks. The learning rates affects the step size of the network weight adjustments. There are no specific learning rates that are known to be best since "... the appropriate choice of p (learning rate) is problem-specific, ..." [Gallant, 93, p. 220] However, the learning rates should be large enough for fast error
reduction, but not large enough to cause network weight oscillations. The typical range for learning rates is between 0 and 1. For this project, the learning rates for each layer of connections were set as follows:

<table>
<thead>
<tr>
<th>Between...</th>
<th>Learning Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>input and 1st hidden layer</td>
<td>0.7</td>
</tr>
<tr>
<td>1st and 2nd hidden layer</td>
<td>0.5</td>
</tr>
<tr>
<td>2nd hidden layer and outputs</td>
<td>0.4</td>
</tr>
</tbody>
</table>

These initial learning rate values were chosen for their relatively fast reduction in RMS error at the start of network training, but these rates were reduced by one-half of their values at several stages within the training process where the training error leveled off. These changes were made to avoid the network weight oscillations typical of networks being trained at points where the weight adjustments become small. A parameter very similar to the learning rate, the momentum, is typically used to “... add speed while avoiding instability...” [Gallant, 93, p. 221]. The momentum was also experimentally chosen to be 0.4 throughout the network.

Initializing the network weights was also required before any network training session. It is typically best to have small, randomly assigned initial weights to a network, since too small or too large weight values may slow the learning process by making the transfer functions at any particular processing element initially too close to either 0 or 1, thereby slowing their adjustments to their optimal values. The initial
weight range was also chosen experimentally, and the network weights were randomly initialized with values between -0.7 and +0.7.

Since there is no set procedure for the actual development of a neural network, a systematic approach to this phase of the project was required. The initial goal for this RMS error was an overall network prediction error less than 0.01. The first network was composed of a single hidden layer network with five processing elements. This network was trained until the weights converged to the point that no further reductions in the error level were possible. The number of processing elements within this hidden layer was then increased to ten. Once again, the network was trained until the lowest possible error level was attained. This single hidden layer network training and testing was continued by incrementing the number of processing elements by five up to a total of 70 processing elements. The results in Table 4.2 indicate that after approximately 30 neurons, very little further reduction in error level was possible with a single layer network. The best results obtained by a single layer network was a 45 or 60 neuron network with an RMS error of 0.0495. However, the reduction in error from 0.0500 with the 30 neuron network was insignificant enough that the results did not warrant the use of these larger networks.

To further reduce this error, the network complexity was increased by adding an additional hidden layer. With a two hidden layer network, there are many more processing element combinations to explore. The two hidden layer network testing began with the first hidden layer containing 30 processing elements. From this point, the second layer varied the number of processing elements in the same manner as the
single layer network. The results are shown in Table 4.2. The best results obtained with a two hidden layer network had 30 processing elements in the first hidden layer and 25 processing elements in the second hidden layer at an overall training RMS error of 0.0515. These results were far from the initial RMS error goal, and therefore further network testing and development was performed.

Table 4.2: Network Development Results

<table>
<thead>
<tr>
<th>Neurons</th>
<th>Single Hidden Layer</th>
<th>RMS error</th>
<th>Two Hidden Layers</th>
<th>Neurons</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>RMS error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.0550</td>
<td></td>
<td></td>
<td>30</td>
<td>5</td>
<td></td>
<td>0.0530</td>
</tr>
<tr>
<td>15</td>
<td>0.0510</td>
<td></td>
<td></td>
<td>30</td>
<td>10</td>
<td></td>
<td>0.0570</td>
</tr>
<tr>
<td>30</td>
<td>0.0500</td>
<td></td>
<td></td>
<td>30</td>
<td>25</td>
<td></td>
<td>0.0515</td>
</tr>
<tr>
<td>45</td>
<td>0.0495</td>
<td></td>
<td></td>
<td>30</td>
<td>40</td>
<td></td>
<td>0.0515</td>
</tr>
<tr>
<td>60</td>
<td>0.0490</td>
<td></td>
<td></td>
<td>30</td>
<td>55</td>
<td></td>
<td>0.0520</td>
</tr>
<tr>
<td>75</td>
<td>0.0500</td>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td>70</td>
<td>0.0525</td>
</tr>
</tbody>
</table>

Alternative first hidden layer sizes investigated included 10, 20, 40, 55, and 70 processing elements. With each of these alternatives, the second hidden layer was varied in size similar to the manner mentioned previously. No other network with two hidden layers was able to go below the RMS error level achieved by the network with 30 processing elements in the first hidden layer and 25 processing elements in the
second hidden layer (see Appendix D). Furthermore, even though this two layer network had a slightly higher training RMS error than the single layer network, the testing RMS error was slightly lower for the two layer network, 0.1183 for the two layer network versus 0.1231 for the single layer network. Within this trained network, the RMS error of the individual ingredient flowrate output variables ranged from approximately 0.01 to 0.06. The error associated with the three main output variables of interest was even higher, which most likely contributed to the poor results. The pressure, temperature, and cost output variables had RMS errors of approximately 0.03, 0.12, and 0.09 respectively. Therefore, based on these results, it became apparent that additional project work will be required to reduce the network RMS error levels to achieve the desired accuracy.

4.6 Shell Program Development and System Performance

A shell program in 'C' code provided a means of linking each individual reactor represented by the neural networks into an overall series of reactors (Appendix E). The program first asked for the number of reactors in series to be analyzed. Following this, an initial typical selection on the degree of polymerization, DEG content, and reactor volume (the search variables) for each reactor in series was entered via an input file. The shell program then called each reactors' neural network in series with the given search variables' values. During program execution, the shell program was responsible
for ensuring that the outputs of one reactor were used as inputs to the next reactor.

When the final reactor in series was evaluated, the overall reactor series cost was
totaled and returned.

A manual search replaced the proposed ‘canned’ search routine for preliminary
testing since the target network RMS error was not achieved. A one-at-a-time, single
variable search approach was manually executed (Figure 4.2). An initial solution was
chosen as the starting point for the search, at which point the cost function was
evaluated. This starting solution was chosen with search variable values close to their
respective values shown to be optimal in Pattalachinti’s research. A single variable was
changed by a fixed amount. Then, the cost function was evaluated at this new point to
determine the effect of the change. If the resulting cost function was reduced, the same
variable was changed again in the same direction. If the change did not reduce the cost
function, then the variable was changed in the other direction to determine its effect. If
both variable directions were tried without a resulting decrease in the objective
function, then the same process was repeated on the next variable. This overall process
continued until no further changes in any of the search variables resulted in a decrease
in the objective function, at which point either a local or a global minimum was
reached.

The initial plans for this system was to have the shell program call a ‘canned’
search routine which would have manipulated the search variables and tracked the
results. The search routine would have been responsible for using the results generated
by the neural network system to determine the optimal solution set. Since the lowest
Figure 4.2: Flowchart of search heuristic
training RMS error level achieved by any network was significantly greater than the

target RMS error level set to achieve accurate results, the system was instead used to

validate the results obtained by the multi-reactor FORTRAN program. The search

results are provided in Appendix D.
CHAPTER 5: TEST RESULTS

5.1 Test Results

Test trials of the artificial neural network chemical reactor optimization system were performed to determine its ability to derive optimal reactor series design parameters. Three examples were performed with different numbers of reactors in the series and different final product characteristics. Test example one had a target degree of polymerization of 80 with two reactors in series. Test example two also had a target degree of polymerization of 80 but with three reactors in series. Test example three had a target degree of polymerization of 120 with five reactors in series. All three of the examples had an upper limit on the DEG content of 1.60 mole %. Initial feed specifications consisted of a molar flowrates of 20 kg mol/hr for ingredients as ‘EC’ and ‘FG’ and an input temperature of 290 degrees Fahrenheit based on Pattalachinti’s work. The first two test examples’ results were then compared to the results attained by similar test examples performed in the work of Pattalachinti. The results are shown in tables 5.1, 5.2, and 5.3.

The results of the test examples came up less than satisfactory. Each test example’s results came up far from those achieved by Pattalachinti. Furthermore, the search variables’ final values deviated significantly from values that even reflect realistic results. The cost of the reactor series ended up a negative value in each case, and is most likely due to the relatively large network error creating values outside of the
Table 5.1: Two reactor test problem results

<table>
<thead>
<tr>
<th>Reactor</th>
<th>Pattalanchinti Program</th>
<th>Neural Network System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Volume</td>
<td>DP</td>
</tr>
<tr>
<td>1</td>
<td>11.67</td>
<td>7.299</td>
</tr>
<tr>
<td>2</td>
<td>100.28</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 5.2: Three reactor test problem results

<table>
<thead>
<tr>
<th>Reactor</th>
<th>Pattalanchinti Program</th>
<th>Neural Network System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Volume</td>
<td>DP</td>
</tr>
<tr>
<td>1</td>
<td>2.58</td>
<td>5.116</td>
</tr>
<tr>
<td>2</td>
<td>4.30</td>
<td>17.256</td>
</tr>
<tr>
<td>3</td>
<td>37.11</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 5.3: Five reactor test problem results

<table>
<thead>
<tr>
<th>Reactor</th>
<th>Neural Network System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Volume</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
</tr>
</tbody>
</table>
normalized reactor cost variable's range. The procedure remained the same for the test examples, however, in that the desired result was a minimum cost reactor series, whether that value was negative or positive. It did appear, though, that the first reactor in the series, which had a single neural network model of its own, had more reasonable reactor cost results than those reactors that followed.

In evaluating each of the test examples, the reactor volume changed between the first reactor and the rest of the reactors in series, but remained the same for each reactor thereafter. Also, the remaining reactors all had the same solution values for the degree of polymerization and the DEG content. This indicates that the two networks in the system were, although linked together, operated significantly differently. The second network in the neural network system appeared to have a problem determining reliable results, and returned to a single solution state with redundant search variable values for each reactor in the series. This may be interpreted as the system wanted to only use two reactors in the entire series to reach the final product specifications, which is contrary to typical reactor series designs. Problems with the system also included calculated negative flowrates from one reactor to the next, which undoubtedly influenced the poor performance of the system. These problems may once again be attributed to the poor network training results.

The neural network optimization system did, however, present features which indicated that the networks had learned to a degree and that the overall system was performing as intended. First, the continuous cost objective function demonstrated peaks and valleys during the solution process. When increasing a search variable in one
direction, the cost function would change in a single direction until a peak or valley was encountered. At that point, any further changes in the search variable would change the cost function in the opposite direction. This indicates that the networks had learned the input-output relationships to a given degree.

Second, the shell program performed as intended. Input parameters were correctly read from an input file. The output variables from each network were fed as input variables to the following reactor, creating an overall reactor series. The cost functions for each reactor were tallied to a final overall reactor series cost, upon which the search direction was determined.

Finally, the manual search procedure used in this research was able to identify a minimum-cost reactor series using the values supplied by the neural networks. This shows the potential for calling a ‘canned’ search routine from the shell program to quickly search through the solution space, given the initial starting solution. These ideas together indicate that the potential remains for this system to become a very useful tool in reactor series optimization.
CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The objective of this research was to use artificial neural networks determine a minimum cost series of chemical reactors in the production of polyethylene terephthalate. Once properly trained and tested, artificial neural networks were used to model single chemical reactors. These neural network reactor models were then linked together through a shell program in ‘C’ code to form an overall series. This system was then used to search through the solution space for a global optimal solution.

Although the system did not perform as intended, the potential for this solution approach to develop into an efficient and accurate means of determining an optimal chemical reactor series remains a possibility at this point.

This research work began with modifications to an existing FORTRAN program in order to collect neural network training and testing data set examples relating to the chemical reactor processes. The data was analyzed by plotting the values on histograms for any conditions that may have affected the ability of the neural networks to train properly. A significant difference became apparent between the values from the first reactor and the values from the rest of the reactors in series. Due to these differences, two networks were required for the optimization system.
Initial network investigation was then performed to determine the appropriate network training parameters. Network development followed with a directed trial-and-error approach to determining the lowest error network architecture. Various network architectures were investigated, but none were able to achieve the desired RMS error goal of 0.01. The best network architecture was determined to be a network with 30 processing elements in the first layer and 25 processing elements in the second layer, with a training RMS error of 0.0515. Even with the less than desirable results, research continued to determine if the system could still be used as a check for other reactor optimization systems.

The last phase of the system development was to create a 'C' code shell program to coordinate all of the necessary activities within the system. In this program, the initial parameters are read from an input file and are passed on to the first neural network reactor model. From this point, the shell program passes the outputs of each reactor on as inputs to subsequent reactors. The shell program sums the overall reactor series cost for an evaluation at the end of each iteration, and the objective is to minimize the sum of the reactors' costs.

Three test examples of the neural network reactor series optimization system were completed to validate the results. The first test problem had two reactors in series, the second had three reactors in series, and the third had five reactors in series. A manual search routine was performed and results were compared to similar test problems from the work of Pattalachinti. Results turned out unsatisfactory, with unrealistic solutions varying significantly from those of Pattalachinti. In each final
solution, the first reactor appeared to have reasonable values for the reactor volume, degree of polymerization, and DEG content. However, the remaining reactors in series each had very small reactor volumes, and these reactors repeated the same degree of polymerization and DEG content as that of the first reactor. This indicates that the neural network system consistently tries to achieve the desired end product specifications through only two reactors in series which, for many cases, is not an efficient chemical manufacturing process. The unrealistic results were caused by the network training error levels that were above the RMS error goal of 0.01. This network error was multiplied down through the reactor series, with each reactor models’ errors being passed on to subsequent reactors with further error. However, a few positive system results included continuous properties that were displayed by the objective function, the networks were able to learn to a given error level, and the overall neural network system performed as intended.

6.2 Recommendations

Given the large RMS error level in training the networks and the poor test example results, this system is not currently recommended as a means of determining reactor series design parameters. However, given the potential benefits of such a system and the positive aspects identified in the test examples, the following items are recommended as future research work:
- Develop a lower error neural network for use in the neural network optimization system. This may exist through further experimentation with backpropagation neural network architecture and parameters or through the use of other types of artificial neural networks. Better trained networks should provide more accurate and reliable results in the optimization system.

- Develop a cost function external to the neural networks as a means of evaluating the search process. The external cost function would provide a more accurate and realistic cost evaluation upon which to base the search process. The search process would remain essentially the same, with the exception that the cost function would no longer be a value calculated by the networks. By simply ignoring the cost output variable of the existing trained networks, these networks could be used as they currently are within this process. The opportunity would exist, though, to further reduce the overall RMS error of the networks by eliminating the cost variable as an output, thereby simplifying the overall input-output relationship pattern. The cost variable appeared to add complexity to the network, demonstrated by its relatively high RMS error of 0.09 when compared to the flowrates RMS errors in the range of 0.01 - 0.06. This simplification should allow lower RMS error levels throughout all network outputs and provide better results than were achieved in this research. The possibility remains, however, that this error reduction would not have a big enough impact to dramatically improve the results.

The external cost function could be developed in one of two manners. The first method would be to independently develop the system of equations necessary for the
cost evaluation based on the calculations used in the FORTRAN program. However, an initial investigation into this possibility revealed that the equations are extremely complex and interrelated and would take a good deal of time to develop. A second alternative would be to try and determine a point in the existing FORTRAN program at which a subset of the variables could be entered via the neural network program and then have the FORTRAN program complete the cost function evaluation. However, this also appears to be a very involved process.

- Train the neural networks on a more robust data set. In this research work, only one set of initial reactor ingredient flowrates and material temperature were used to train the networks. Training the networks on a variety of starting conditions would generate more robust models for use in the optimization system.

- Train the neural networks using data from the FORTRAN program in the research work of Calmeyn [1995]. This program included changes that eliminated the discontinuities in the objective function. The elimination of these problems in formulating the neural network training and testing data sets might eliminate problems with the data itself. This in turn could provide a lower network training error and improve the search process for a minimum cost reactor series.

- Investigate alternative input-output data arrangement. Although the experimentation within this research indicated that the most appropriate input-output variables were as
chosen, other variable arrangements could ultimately prove better when considering the training and testing of the neural networks.
BIBLIOGRAPHY


APPENDICES
APPENDIX A: FORTRAN PROGRAM MODIFICATIONS

The first page of this section displays the program execution sequence for the modified FORTRAN program that was used to collect the necessary input and output data sets for training and testing of the neural networks in this research. The pages that follow contain the specific FORTRAN program subroutines that were modified.

MAIN

INPUT: Read input parameters

OUTPUT: Exit program if inequality constraints are initially violated

NEWTON: Main subroutine that finds values for input variables

CONSTRAINTS: Finds error in material balance constraints

JACOBIAN: Calculates derivatives for Jacobian matrix

INVERSE: Finds inverse of desired matrix

OBJECTIVE: Main subroutine that calculates objective function value (cost)

CONSTRAINTS:

JACOBIAN:

SENSITIVITY: Calculates derivatives for sensitivity functions

INVERSE:

COST: Determines cost for reactor

OUTPUT: Writes output file

COST:
C****************************************************************
C MAIN PROGRAM
C****************************************************************

IMPLICIT REAL*8(A-H,O-Z)

REAL*8
x KR,KRA,KRB,KTL,MOTORCS,MOTOROS,MUL,MUV,MW,MWN,MWW,MWV,
x NA,NP,NPCS,NNU,NRE,NPR

COMMON
x AC,AGITCS,AHC,AHJ,AHJE,AHX,
  x C(8),COILCS,CONDCS,CONV,CP,CS,
x DA,DCDFL(8,8),DD,DEG(1),DEGT,DEL(10),DELD(7),
x DELI(3),DELT,DELX(7),DERNUM(7,10),
x DFDG(15,15),DFDGI(9,9),
x DFDG0(9,9),DFDX(7,10),DFDXD(7,7),
x DFDXI(7,3),DFDXDI(7,7),DFDXU(7,10),DFDY(9,3),
x DFGDG0Y(9,3),DFRDFL(8,8),DFRDT(8),DFRDV(8),
x DFVDFL(8,8),DFVDNFL(8,8),DFVDV(8),DFVTD(8),DFVTDFL(8),
x DGDY(9,3),DG0DY(9,3),
x DGLTDFL(8),DHLDT(8),DHSDFL(8),
x DHVDNFL(8),DKRDT(22),DL,DOWCS,DOWO$,DNPCS, 
x DPFL(8),DPDT,DPDFL(8,8),DPPDT(8),DPRDT(8),
x DP(1),DP0,DPT,DQDX(10),DQDXD(7),DQDXI(3),
x DR,DRDFL(22,8),DRDT(22),DS,DTLM,
x DUX,DXLDFL(8,8),DXUDX(10,10),DXVDFL(8,8),DXVDV(8),
x EPS1,EPS2,EPS3,EQ(1),
x F(7),FO(7),FAIR,FAT,FET,FIT,FL(8),FLT,
x FL0(8),FLF(8),FLTF,FR(8),FSS,FV(8),FVT,
x G,GAIR,GLT,GLTF,GR(3),GS,GSJ,GSTEAM,GVT,
x H,HEXC$,HL(8),HLA(8),HLB(8),HW(1),
x IB(10),ID(8),IFLAG,IG(8),IO(8),IR,
x IRA(22),IRB(22),IRT(1),IS(8,22),IT(8),IX(22),
x KFLAG,GR(22),KRA(22),KRB(22),
x KTL,MUL,MUV,MW(8),MWV,MOTORCS,MOTOROS,MWN,MWW,
  x NA,NC,ND,NF1,NF2,NF3,NG,NI,NM,NNU,NP,NPCS,
  x NPR,NR,NRE,NS,NT,NV,NX,NZ,
x OC,P,PA,PI,PP(8),
x PR(8),PRA(8),PRB(8),PRC(8),PRO,PUMPC$, 
x R(22),REACCS,RG,RPUMPC$, 
x SCALEA(10),SCALEB(10),SJC$(4),SJETCS,SJETNPC(4), 
x SJETOS, SJOS$(4),STEP1,STEP2,SUF,
x T,TC1,TC2,TCI,TF,TH1,TH2,TOC,TPC,T0,
x UC,UHJ,UHC,UHR,UHX,UI,USAGE(4),
x V,VARDP(1),VARG(9),VARY(3),SDEV(3),VNPCS$,WS, 
x X(10),X0(10),XI(10),XF(1,10),

...
C Input starting guesses for X vector, set feed conditions, set
C physical property and reaction rate parameters, set desired
C product DP.

CALL INPUT

C Initialize function evaluation counters.

NF1 = 0
NF2 = 0
NF3 = 0

C Starting with initial guess for elements of X vector, use
C Newton-Raphson to find a set of dependent variables which
C satisfy the constraint equations.

CALL NEWTON

C Find the objective function value of the first feasible point
C and output results.

CALL OBJECTIVE(0,0,D0,Q)
C WRITE(*,900) (X(I),I=1,NR*NT),Q
C WRITE(*,901) IFLAG,NG,NF1,NF2,NF3,1
C WRITE(*,*)

C If optimization is not desired or if no feasible solution
C could be found using the initial guess, output results and
C stop.

IF( KFLAG .EQ. 0 .OR. IFLAG .EQ. 1 ) THEN
   CALL OUTPUT(Q)
   WRITE(1,*)
   WRITE(1,*)'These results have not been optimized'
   IF( IFLAG .EQ. 1 ) THEN
      WRITE(1,*)'Constraint equations not satisfied'
      WRITE(*,*)CONSTRAINT EQUATIONS NOT SATISFIED'
      WRITE(*,*')TRY A DIFFERENT STARTING POINT'
      WRITE(*,*')OR INCREASE NM'
   ENDIF
   STOP
ENDIF

C Begin main iteration loop.

DO 700 NG = 1,2000
C Save feasible solution in X0.

    Q0 = Q
    DO 300 I = 1,NV
    300 X0(I) = X(I)

C Conduct a linesearch along the reduced gradient.

    CALL LINESEARCH

    CALL OBJECTIVE(0,0,D0,Q1)
    C WRITE(*,900) (X(I),I=1,NR*NT),Q1
    C WRITE(*,901) IFLAG,NG,NF1,NF2,NF3,2
    C WRITE(*,*

C Save the starting point for Newton-Raphson search in X1.

    400 DO 500 I = 1,NV
    500 X1(I) = XCI)

C Starting with the result of the line search, use Newton-Raphson to find a set of dependent variables which satisfy the material and energy balances.

    CALL NEWTON

C Find the objective function value of the new feasible point and output results to the monitor.

    CALL OBJECTIVE(0,0,D0,Q)
    C WRITE(*,900) (X(I),I=1,NR*NT),Q
    C WRITE(*,901) IFLAG,NG,NF1,NF2,NF3,3
    C WRITE(*,**)

C Check to see if the objective function value for the new feasible point is close to the objective function value for the previous feasible point and stop the search if it is.

    IF( DABS(Q-Q0) .LT. EPSI ) THEN
        CALL OUTPUT(Q)
        STOP

C If the search is going the wrong way (increasing objective function value) or the Newton-Raphson search failed, pick a new starting point for the Newton-Raphson search which is half way between the current starting point and the previous feasible point. Then restart the Newton-Raphson search.

    ELSEIF( Q .GT. Q0 .OR. IFLAG .EQ. 1 ) THEN
DO 600 I = 1,NV
600  X(I) = 0.8D0*X0(I) + 0.2D0*X1(I)
     GOTO 400

C Otherwise, save the new feasible point and start a new
C iteration.

ENDIF
700 CONTINUE

C Stop the search if 1000 iterations have been completed.

WRITE(*,*) 'NO CONVERGENCE AFTER 2000 STEPS'
   CALL OUTPUT(Q)
   STOP

900 FORMAT(6E13.7)
901 FORMAT(6I13)

END
SUBROUTINE INPUT

IMPLICIT REAL*8(A-H,O-Z)

REAL*8
x KR,KRA,KRB,KTL,MOTORCS,MOTOROS,MUL,MUV,MW,MWN,MWW,MWV,
x NA,NP,NPCS,NNU,NRE,NPR

COMMON
x AC,AGITC$,AHC,AHJ,AHJE,AHX,
x C(8),COILCS,CONDSC$,$CONV,CP,CS,
x DA,DCDFL(8,8),DD,DEG(1),DEGT,DEL(10),DELD(7),
x DELI(3),DELT,DELX(7),DERNUM(7,10),
x DFDG(15,15),DFDG(9,9),
x DFDG(9,9),DFDX(7,10),DFDXD(7,7),
x DFDXI(7,3),DFDXDI(7,7),DFDUX(7,10),DFDY(9,3),
x DFGO(9,9),DFDFO(8,8),DFRDFT(8),DFRDV(8),
x DFVD(8,8),DFDVF(8,8),DFVD(8),DFVD(8),
x DGDY(9,3),DGDY(9,3),
x DGLTDFL(8),DHLD(8),DHSDFL(8),
x DHVDFL(8),DKRDFT(22),DL,DWCS,DOWOS,DNPC$,
x DPDFL(8),DPDT,DPDFL(8,8),DPPTD(8),DPRTD(8),
x DP(1),DP0,DPT,DQDX(10),DQDXD(7),DQDXI(3),
x DR,DRDFL(22,8),DRDT(22),DS,DTLM,
x DX,DXDFL(8,8),DXUD(10,10),DXVDFL(8,8),DXVDT(8),
x EPS1,EPS2,EPS3,EQ(1),
x F(7),FO(7),FAd,FAT,FIT,FI,FL(8),FLT,
x FL(8),FLF(8),FLTF,FR(8),FSS,FV(8),FVT,
x G,GAIR,GLT,GLTF,GR(3),GS,GSJ,GSTEAM,GVT,
x H,HEXCS,HL(8),HLA(8),HLB(8),HW(1),
x IB(10),ID(8),IFLAG,IG(8),IO(8),IR,
x IRA(22),IRB(22),IRT(1),IS(8,22),IT(8),IX(22),
x KFLG,KR(22),KRA(22),KRB(22),
x KTL,MUL,MUV,MW(8),MWW,MWV,MOTORCS,MOTOROS,MWN,MWW,
x NA,NA,ND,NS,NF1,NF2,NF3,NG,NI,NM,NNU,NP,NPCS,
x NPR,NR,NS,NT,NV,NW,NZ,
x OC,OP,PA,PB,PP(8),
x PR(8),PRA(8),PRB(8),PRC(8),PRO,PUPC$,
x R(22),REACCS,RG,RPUMPC$,
x SCALEA(10),SCALEB(10),SCJ(4),SJETC$,$SJETNC(4),
x SJET$,$SJET(4),STEP1,STEP2,SUF,
x T,TC1,TC2,TIC,TF,TH1,TH2,TOC,TPC,T0,
C Open I/O files.

OPEN(2,FILE = 'C:\cheproj\GRGR.INP')
OPEN(1,FILE = 'C:\cheproj\GRGR.OUT')

C Determine if optimization is to be done.

READ(2,*) KFLAG

C Set number of reactors.

READ(2,*) NR

C Set the reactor types. 0 for CSTR and 1 for Disk Ring.

DO 200 I = 1, NR
200 READ(2,*) IRT(I)

C Input target degree of polymerization, penalty factor for product variability, and penalty factor for DEG content.

READ(2,*) DPT
READ(2,*) DEGT
READ(2,*) PRO

C Input standard deviations for fluctuating operating parameters in each reactor.

DO 210 I = 1, NR
210 READ(2,*) (SDEV(3*(I-1)+J), J=1,3)

C Set the number of components in the input file

READ(2,*) NC

C Set the identifiers for the components included in the model. C ID=1 implies the component is included. Set the identifiers which indicate what type each component is. IT=1 for free species, IT=2 for end group, IT=3 for internal repeat unit. C IG=0 for wanted species, IG=1 for unwanted species (DEG). C IO saves the old numbering of the components involved.

DO 220 J = 1, NC
220 READ(2,*) ID(J), IT(J), IG(J)
C Eliminate deleted components from IT and IG. Set identifier
C which relates the component number in the reduced set to the
C component number in the complete set (IO).

        KKK = 0
        DO 230 K = 1,NC
           IF( ID(K) .EQ. 1 ) THEN
              KKK = KKK + 1
              IT(KKK) = IT(K)
              IG(KKK) = IG(K)
              IO(KKK) = K
           ENDIF
        230 CONTINUE

C Determine number of components in reduced set (NZ), total
C number of search variables for each reactors (NT), number
C of material balances used for each reactor (NB), total
C number of search variables for all of the reactors (NV),
C number of dependent variables (ND), the number of independent
C variables (NI), and the number of sensitivity variables (NS)

        NZ = KKK
        NT = NZ + 2
        NB = NZ - 1
        NV = NT*NR
        ND = NB*NR
        NI = NV - ND
        NS = 3*NR

C Set feed conditions.

        READ(2,*), T0
        DO 240 J = 1,NC
        240 READ(2,*), FLO(J)

C Eliminate feed components not included in this simulation.

        KKK = 0
        DO 250 K = 1,NC
           IF( ID(K) .EQ. 1 ) THEN
              KKK = KKK + 1
              FLO(KKK) = FLO(K)
           ENDIF
        250 CONTINUE

C Calculate total flows by species type.

        AAA = 0.D0
BBB = 0.D0
CCC = 0.D0
DO 260 I = 1,NZ
IF( IT(I) .EQ. 1 ) AAA = AAA + FL0(I)
IF( IT(I) .EQ. 2 ) BBB = BBB + FL0(I)
IF( IT(I) .EQ. 3 ) CCC = CCC + FL0(I)
260 CONTINUE
FFT0 = AAA
FETO = BBB
FIT0 = CCC

C Calculate total liquid species flowrate for VLE model.

FAT0 = FFT0 + 0.5D0*FETO + FIT0

C Calculate degree of polymerization

DP0 = 1.D0 + 2.D0*FIT0/FETO

C Set search parameters.

READ(2,*) EPS1
READ(2,*) EPS2
READ(2,*) EPS3
READ(2,*) STEP1
READ(2,*) STEP2
READ(2,*) DX
READ(2,*) NM

C Set the scaling parameters which are used to translate the
C dependent variables into a scaled form.

DO 270 I = 1,NC+2
270 READ(2,*) SCALEA(I),SCALEB(I)

C Eliminate scaling parameters for deleted components.

KKK = 0
DO 280 K = 1,NC
IF( ID(K) .EQ. 1 ) THEN
   KKK = KKK + 1
   SCALEA(2+KKK) = SCALEA(2+K)
   SCALEB(2+KKK) = SCALEB(2+K)
ENDIF
280 CONTINUE

C If there is more than one reactor in the simulation, copy the
C scaling factors for the first reactor to the array positions
C for all reactors.
IF (NR .GT. 1) THEN
  DO 290 J = 2,NR
  DO 290 I = 1,NT
    SCALEA(NT*(J-1)+I) = SCALEA(I)
  290  SCALEB(NT*(J-1)+I) = SCALEB(I)
ENDIF

C Create scaling matrix. Begin by initializing the matrix to
C zero, then fill in the non-zero elements. (The current one-
C to-one correspondence between unscaled and scaled elements
C makes this a diagonal matrix.

  DO 300 I = 1,NV
  DO 300 J = 1,NV
300  DXUDX(I,J) = 0.D0

  DO 310 I = 1,NV
310  DXUDX(I,I) = SCALEB(I)

C Input initial guess for search variables.

  READ(2,*)
  READ(2,*)

  DO 360 I = 1,NR
  READ(2,*) V
  READ(2,*) T
  DO 320 J = 1,NC
320  READ(2,*) FL(J)

C Delete components which aren't to be included in the
C simulation.

    KKK = 0
    DO 330 K = 1,NC
      IF( ID(K) .EQ. 1 ) THEN
        KKK = KKK + 1
        FL(KKK) = FL(K)
      ENDIF
    330  CONTINUE

C Scale variables before beginning the search.

    X(NT*(I-1)+1) = (V-SCALEA(NT*(I-1)+1))/SCALEB(NT*(I-1)+1)
    X(NT*(I-1)+2) = (T-SCALEA(NT*(I-1)+2))/SCALEB(NT*(I-1)+2)
    DO 340 J = 3,NT
340  X(NT*(I-1)+J) = (FL(J-2)-SCALEA(NT*(I-1)+J))/
C Save starting values for search variables for later output.

    XS(I,1) = V
    XS(I,2) = T
    DO 350 J = 3,NT
    350 XS(I,J) = FL(J-2)

C Go on to next reactor.

    360 CONTINUE

C Determine which of the independent variables are in the basis.
C That is, which variables are contained in Xd.

    KKK = 0
    DO 370 I = 1,NR
    DO 370 J = 1,NC+2
    KKK = KKK + 1
    370 READ(2,*), IB(KKK)

    KKK = 0
    LLL = 0
    DO 380 I = 1,NR
    DO 375 K = 1,2
    LLL = LLL + 1
    KKK = KKK + 1
    375 IB(KKK) = IB(LLL)
    DO 380 K = 3,NC+2
    LLL = LLL + 1
    IF( ID(K-2) .EQ. 1 ) THEN
      KKK = KKK + 1
      IB(KKK) = IB(LLL)
    ENDIF
    380 CONTINUE

C Input the species physical property parameters.

    READ(2,*)
    READ(2,*)
    READ(2,*)
    READ(2,*)
    READ(2,*)
    READ(2,*)
    DO 390 I = 1,NC
    READ(2,*) MW(I)
    READ(2,*) PRA(I),PRB(I),PRC(I)
C Delete physical property parameters for excluded components.

      KKK = 0
      DO 395 K = 1,NC
         IF( ID(K) .EQ. 1 ) THEN
            KKK = KKK + 1
            MW(KKK) = MW(K)
            PRA(KKK) = PRA(K)
            PRB(KKK) = PRB(K)
            PRC(KKK) = PRC(K)
            HLA(KKK) = HLA(K)
            HLB(KKK) = HLB(K)
         ENDIF
      395 CONTINUE

C Set liquid density, spray density, vapor viscosity, heat capacity.

      DL = 1300. DO
      DS = 1113. DO
      MUV = .054 DO
      CP = 0.5 DO

C Set heat transfer parameters.

      UHJ = 1000. DO
      UHC = 1000. DO
      UHX = 1500. DO
      KTL = 0.215 DO

C Set the number of reactions in the input file (NX).

      READ(2,*), NX

C Set reaction identifiers.

      READ(2,*)
      READ(2,*)
      READ(2,*)
      READ(2,*)
      DO 400 J = 1,NX
         READ(2,*), (IS(I,J), I = 1,NC), IX(J)
      400 CONTINUE
      DO 410 I = 1,NX
      410 READ(2,*), IRA(I), IRB(I), KRA(I), KRB(I)
C Delete reaction parameters for excluded reactions.

    KKK = 0
    DO 430 K = 1,NX
    IF( IX(K) .EQ. 1 ) THEN
        KKK = KKK + 1
        DO 420 I = 1,NC
        420 IS(I,KKK) = IS(I,K)
        IRA(KKK) = IRA(K)
        IRB(KKK) = IRB(K)
        KRA(KKK) = KRA(K)
        KRB(KKK) = KRB(K)
    ENDIF
    430 CONTINUE
    NX = KKK

C Delete reaction parameters for excluded components.

    KKK = 0
    DO 450 K = 1,NC
    IF( ID(K) .EQ. 1 ) THEN
        KKK = KKK + 1
        DO 440 I = 1,NX
        440 IS(KKK,I) = IS(K,I)
    ENDIF
    450 CONTINUE

C Reassign component identifiers for reactants and products
C since component numbers have been re-identified.

    DO 460 I = 1,NX
    DO 460 J = 1,NZ
    IF( IRA(I) .EQ. IO(J) ) THEN
        IRA(I) = J
    ENDIF
    460 CONTINUE

    DO 470 I = 1,NX
    DO 470 J = 1,NZ
    IF( IRB(I) .EQ. IO(J) ) THEN
        IRB(I) = J
    ENDIF
    470 CONTINUE

C Set ideal gas constant.

    RG = 1.987 D0

C Input values for vacuum system costing.
FAIR = 3.49 D-1
GAIR = 1.00 D1
CS = 8.82 D-3
SUF = 0.8 D0
FSS = 1.5 D0
TH1 = 298.65 D0
TH2 = 298.15 D0
TC1 = 293.15 D0
TC2 = 293.15 D0

C Input maximum entrained particle diameter. (m)
DD = 5. D-5

C Input miscellaneous constants. (Units of G are m/hr/hr)
PI = 3.141592 D0
G = 1.27 D8

C Check inequality constraints and stop if any are violated.

DO 480 I = 1,NV
IF( X(I) .LT. 0.D0 ) THEN
   WRITE(*,*) 'STARTING POINT VIOLATES CONSTRAINT'
   CALL OUTPUT(0.D0)
   STOP
ENDIF
480 CONTINUE

C Return to main program
RETURN
END
SUBROUTINE CONSTRAINTS

IMPLICIT REAL*8(A-H,O-Z)

REAL*8
x KR,KRA,KRB,KTL,MOTORC$,MOTORO$,MUL,MUV,MW,MWN,MWW,MWV,
x NA, NP,NPCS$,NNU,NRE,NPR

COMMON
x AC,AGITC$,AHC,AHJ,AHJE,AHX,
x C(8),COILC$,CONDC$,CONV,CP,CS,
x DA,DCDFL(8,8),DD,DEG(1),DEGT,DEL(10),DELD(7),
x DEL(3),DELT,DELX(7),DERNUM(7,10),
x DFDG(15,15),DFDGI(9,9),
x DFDGO(9,9),DFDX(7,10),DFDXD(7,7),
x DFDXI(7,3),DFDXDI(7,7),DFDXU(7,10),DFDY(9,3),
x DFg0DG0Y(9,3),DFRDFL(8,8),DFRDT(8),DFRDV(8),
x DFVDFL(8,8),DFVDFLF(8,8),DFVDT(8),DFVTD(8),
x DGDY(9,3),DG0DY(9,3),
x DGLTDFL(8),DHLDT(8),DHSDFLF(8),
x DHVDFLF(8),DKRDT(22),DL,DOWC$,DOWO$,DNPC$,x DPDFL(8),DPDT,DPPDFL(8,8),DPPDT(8),DPRDT(8),
x DP(1),DP0,DPT,DQDX(10),DQDXD(7),DQDXI(3),
x DR,DRDFL(22,8),DRDT(22),DS,DTLM,
x DX,DXLDFL(8,8),DXUDX(10,10),DXVDFL(8,8),DXVDT(8),
x EPS1,EPS2,EPS3,EQ(1),
x F(7),FO(7),FAIR,FAT,FET,FIT,FL(8),FLT,
x FL0(8),FLF(8),FLT,F,R(8),FSS,FR(8),FVT,
x G,GAIR,GLT,GLTF, GR(3),GS,GSI,GSTEAM,GVT,
x H,HEXC$,HL(8),HLA(8),HLB(8),HW(1),
x IB(10),ID(8),IFLAG,IG(8),IO(8),IR,
x IRA(22),IRB(22),IRT(1),IS(8,22),IT(8),IX(22),
x KFLAG,KR(22),KRA(22),KRB(22),
x KTL,MUL,MUV,MW(8),MWV,MOTORC$,MOTORO$,MWN,MWW,
x NA,NB,ND,NF1,NF2,NF3,NG,NI,NM,NNU,NP,NPCS$,x NPR,NR,NRE,NS,NT,NV,NX,NZ,
x OC,P,PA,PI,PP(8),
x PR(8),PRA(8),PRB(8),PRC(8),PRO,PUMPC$,x R(22),REACC$,RG,RPUMPC$,x SCALEA(10),SCALEB(10),SJC$(4),SJETC$,SJETNPC(4),
x SJETOS,SJOS$(4),STEP1,STEP2,SUF,
x T,TC1,TC2,TCI,TF,TH1,TH2,TOC,TPC,T0,
x UC, UHJ, UHC, UHR, UHX, UJ, USAGE(4),
x V, VARDP(1), VARG(9), VARY(3), SDEV(3), VNPCS, WS,
x X(10), X0(10), X1(10), XF(1, 10),
x XL(8), XP(10), XS(1, 10), XV(8), XX(10), NLS

C******************************************************************************
C                  INITIALIATION
C******************************************************************************

C Increment function evaluation counter.

    NF1 = NF1 + 1

C Convert search variables to unscaled form and explicit
C notation.

    J = NT*(IR - 1) + 1
    V = SCALEA(J) + SCALEB(J)*XX(J)
    J = J + 1
    T = SCALEA(J) + SCALEB(J)*XX(J)
    DO 100 K = 1, NZ
    100 FL(K) = SCALEA(J+K) + SCALEB(J+K)*XX(J+K)

C Set feed temperature and flow rates.

    IF( IR .EQ. 1 ) THEN
        DO 110 K = 1, NZ
        110 FLF(K) = FL0(K)
        TF = T0
    ELSE
        J = J - NT
        DO 120 K = 1, NZ
        120 FLF(K) = SCALEA(J+K) + SCALEB(J+K)*XX(J+K)
        IF ( J .EQ. 2 ) THEN
            TF = SCALEA(J) + SCALEB(J)*XX(J)
        ENDIF
    ENDIF

C******************************************************************************
C                  EVALUATION OF CONSTRAINTS FOR NEWTON-RAPHSON SEARCH
C******************************************************************************

C Calculate total molar feed flowrate.

    AAA = 0.D0
    DO 130 K = 1, NZ
    130 AAA = AAA + FLF(K)
    FLTF = AAA
C Calculate total mass feed flowrate.

    AAA = 0.D0
    DO 140 K = 1,NZ
    140 AAA = AAA + MW(K)*FLF(K)
    GLTF = AAA

C Calculate reaction rate constants.

    DO 150 J = 1,NX
    KR(J) = KRA(J)*DEXP(-KRB(J)/RG/T)
    IF(IRT(IR).EQ.1)THEN
    KR(J) = 0.5D0*KR(J)
    ENDIF
    150 CONTINUE

C Calculate total liquid mass product flowrate.

    AAA = 0.D0
    DO 160 K = 1,NZ
    160 AAA = AAA + FL(K)*MW(K)
    GLT = AAA

C Calculate species concentrations.

    DO 170 K = 1,NZ
    170 C(K) = FL(K)*DL/GLT

C Calculate reaction rates.

    DO 180 J = 1,NX
    180 R(J) = KR(J)*C(IRA(J))*C(IRB(J))

C Calculate equilibrium status. This is needed for output only.

    EQ(IR) = 1.D0 - R(2)/R(1)

C Calculate species rates of formation and the sum of
C rates of formations of the species.

    DO 200 I = 1,NZ
    AAA = 0.D0
    DO 190 J = 1,NX
    190 AAA = AAA + IS(I,J)*R(J)
    200 FR(I) = AAA*V

C Calculate total liquid product molar flowrate.
AAA = 0.D0
DO 210 K = 1,NZ
210 AAA = AAA + FL(K)
FLT = AAA

C Calculate vapor product molar flowrate.
FVT = FLTF - FLT

C Calculate total flows by species type.

AAA = 0.D0
BBB = 0.D0
CCC = 0.D0
DO 285 I = 1,NZ
IF( IT(I) .EQ. 1 ) AAA = AAA + FL(I)
IF( IT(I) .EQ. 2 ) BBB = BBB + FL(I)
IF( IT(I) .EQ. 3 ) CCC = CCC + FL(I)
285 CONTINUE
FFT = AAA
FET = BBB
FIT = CCC

C Calculate total liquid species flowrate for VLE model.
FAT = FFT + 0.5D0*FET + FIT

C Calculate liquid product mole fractions.
C******** XL IS CHANGED FOR END GROUPS********
DO 220 K = I,NZ
XL(K) = FL(K)/FAT
IF( IT(K).EQ.2 )XL(K) = 0.5D0*XL(K)
220 CONTINUE
C*********************************************

C Calculate pure component vapor pressure.
DO 230 K = 1,NZ
230 PR(K) = 760*DEXP(PRA(K)-PRB(K)/(PRC(K)+T))

C Calculate partial pressures.
DO 240 J = 1,NZ
240 PP(J) = XL(J)*PR(J)

C Calculate total pressure.
AAA = 0.D0
DO 250 J = 1,NZ
250 AAA = AAA + PP(J)
    P = AAA

C Calculate vapor product mole fractions.

    DO 260 I = 1,NZ
260 XV(I) = PP(I)/P

C Calculate vapor product molar flowrates.

    DO 270 K = 1,NZ
270 FV(K) = XV(K)*FVT

C Find errors in component material balances.

    DO 280 I = 1,NB
280 F(NB*(IR-1)+I) = FL(I) - FLF(I) - FR(I) + FV(I)

C Calculate degree of polymerization

    DP(IR) = 1.D0+2.D0*FIT/FET
    AAA = 0.D0
    DO 290 I = 1,NZ
    IF(IG(I).EQ.1) THEN
        AAA = AAA + XL(I)
    ENDIF
    290 CONTINUE
    DEG(IR) = AAA

RETURN
END
**SUBROUTINE JACOBIAN**

This subroutine calculates analytical derivatives for the Jacobian matrix used in the Newton-Raphson search. The elements of the matrix are identical to elements of matrices needed for the sensitivity functions.

**SUBROUTINE JACOBIAN**

IMPLICIT REAL*8(A-H,O-Z)

REAL*8
  x KR,KRA,KRB,KTL,MOTORC$,MOTORO$,MUL,MUV,MW,MWN,MWW,MWV,
  x NA,NP,NPCS$,NNU,NRE,NPR

COMMON
  x AC,AGITCS$,AHC,AHJ,AHJE,AHX,
  x C(8),COILCS$,CONDC$,CONV,CP,CS,
  x DA,DCDFL(8,8),DD,DEG(1),DELT,DEL(10),DELD(7),
  x DELI(3),DELT,DELX(7),DERNUM(7,10),
  x DFDG(15,15),DFDG(9,9),
  x DFDO(9,9),DFDX(7,10),DFDXD(7,7),
  x DFDXI(7,3),DFDXDI(7,7),DFDXU(7,10),DFDY(9,3),
  x DFGO0G(9,3),DFRDFL(8,8),DFRD(8),DFRDV(8),
  x DFVDFL(8,8),DFVDFLF(8,8),DFVD(8),DFVTDFL(8),
  x DGDY(9,3),DG0DY(9,3),
  x DGLTDFL(8),DHLD(8),DHSDFL(8),
  x DHVDFL(8),DKRDT(22),DL,DOWC$,DOWO$,DNPCS$,
  x DPFL(8),DPDT,DPD(8,8),DPD(8),DPD(8),DPRD(8),
  x DP(1),DP0,DPT,DXD(10),DXD(7),DXDI(3),
  x DR,DRDFL(22,8),DRTD(22),DS,DTLM,
  x DX,DXLDFL(8,8),DXUD(10,10),DXVDFL(8,8),DXVD(8),
  x EPS1,EPS2,EPS3,EQ(1),
  x F(7),FO(7),FAIR,FAT,FET,FIT,FL(8),FLT,
  x FL0(8),FLF(8),FLT,F(8),FSS,FV(8),FVT,
  x G,GAIR,GLT,GLTF,GR(3),GS,GSJ,GSTEAM,GVT,
  x H,HEXC$,HL(8),HLA(8),HLB(8),HW(1),
  x IB(10),ID(8),IFLAG,IG(8),IO(8),IR,
  x IAR(22),IRB(22),IRT(1),IS(8,22),IT(8),IX(22),
  x KFLAG,KR(22),KRA(22),KRB(22),
  x KTL,MUL,MUV,MW(8),MWV,MOTORC$,MOTORO$,MWN,MWW,
  x NA,ND,NF1,NF2,NG,NI,NM,NNU,NP,NPCS$,
  x NPR,NR,NRE,NS,NT,NX,NZ,
  x OC,P,PA,PI,PP(8),
  x PR(8),PRA(8),PRB(8),PRC(8),PRO,PUMPC$,
  x R(22),REACS$,RG,RPUMPC$,
  x SCALEA(10),SCALEB(10),SJCS(4),SJETCS$,SJETNPC(4),
C Increment function evaluation counter.

NF2 = NF2 + 1

C****************************************************************
C EVALUATION OF DERIVATIVES FOR NEWTON-RAPHSON JACOBIAN AND
C SENSITIVITY FUNCTIONS
C****************************************************************

C Calculate partial derivatives of material balances w.r.t.
C species molar flowrates. Note that pressure is not fixed
C for this calculation.

DO 360 K = 1,NZ
DO 350 L = 1,NZ
IF( IT(L) .EQ. 2) THEN
   DXLDFL(K,L) = -XL(K)/FAT*0.5D0
ELSE
   DXLDFL(K,L) = -XL(K)/FAT
ENDIF
350 CONTINUE

IF(IT(K) .EQ. 2) THEN
   DXLDFL(K,K) = DXLDFL(K,K) + 0.5D0/FAT
ELSE
   DXLDFL(K,K) = DXLDFL(K,K) + 1.0D0/FAT
ENDIF
360 DXLDFL(K,K) = DXLDFL(K,K) + 1.0D0/FAT

DO 380 K = 1,NZ
DO 370 L = 1,NZ
370 DCDFL(K,L) = -C(K)/GLT*MW(L)
380 DCDFL(K,K) = DCDFL(K,K) + DL/GLT

DO 390 J = 1,NX
DO 390 L = 1,NZ
390 DRDFL(J,L) = KR(J)*( C(IRB(J))*DCDFL(IRB(J),L)
     + C(IRA(J))*DCDFL(IRA(J),L) )

DO 410 I = 1,NZ
DO 410 L = 1,NZ
AAA = 0.0D0
DO 400 J = 1,NX
400 AAA = AAA + IS(I,J)*DRDFL(J,L)
410 DFRDFL(I,L) = AAA*V

DO 420 K = 1,NZ
   AAA = 0.D0
DO 420 L = 1,NZ
420 DPPDFL(K,L) = PR(K)*DXLDFL(K,L)

DO 430 I = 1,NZ
430 DFVTDFL(I) = -1.D0

DO 450 L = 1,NZ
   AAA = 0.D0
DO 440 J = I,NZ
440 AAA = AAA + DPPDFL(J,L)
450 DPDFL(L) = AAA

DO 460 I = 1,NZ
   DO 460 L = 1,NZ
460 DXVDFL(I,L) = (DPPDFL(I,L)-DPDFL(L)*XV(I))/P

DO 470 I = 1,NZ
   DO 470 L = 1,NZ
470 DFVDFL(I,L) = DXVDFL(I,L)*FVT+XV(I)*DFVTDFL(L)

DO 480 I = 1,NB
   DO 480 L = 1,NZ
   AAA = DFVDFL(I,L) - DFRDFL(I,L)
   IF( I .EQ. L ) AAA = AAA + 1.D0
480 DFDXU(NB*(IR-1)+I,NT*(IR-1)+2+L) = AAA

C*****************************************************************
C Calculate partial derivatives of material balances w.r.t.
C reactor temperature. Note that pressure is not fixed for
C this calculation.

DO 530 J = 1,NX
530 DKRDT(J) = KR(J)*KRB(J)/(RG*T*T)

DO 540 J = 1,NX
540 DRDT(J) = DKRDT(J)*C(IRA(J))*C(IRB(J))

DO 560 I = 1,NZ
   AAA = 0.D0
DO 550 J = 1,NX
550 AAA = AAA + IS(I,J)*DRDT(J)
560 DFRDT(I) = AAA*V
DO 570 K = 1,NZ
570 DPRDT(K) = PR(K)*(PRB(K)/(T+PRC(K))**2)

DO 580 K = 1,NZ
580 DPPDT(K) = DPRDT(K)*XL(K)

AAA = 0.D0
DO 590 K = 1,NZ
590 AAA = AAA + DPPDT(K)
DPDT = AAA

DO 600 K = 1,NZ
600 DXVDT(K) = (DPPDT(K)-XV(K)*DPDT)/P

DO 610 K = 1,NZ
610 DFVDT(K) = DXVDT(K)*FVT

DO 620 I = 1,NB
620 DFDXU(NB*(IR-I)+I,NT*(IR-I)+2) = DFVDT(I) - DFRDT(I)

C****************************************************************
C Find partial derivatives of material balances w.r.t. reactor
C volume.

DO 760 I = 1,NZ
AAA = 0.D0
DO 750 J = 1,NX
750 AAA = AAA + IS(I,J)*R(J)
760 DFRDV(I) = AAA

DO 770 I = 1,NB
770 DFDXU(NB*(IR-I)+I,NT*(IR-I)+1) = - DFRDV(I)

C****************************************************************

C Include effect of one reactor on the next reactor in the
C train (most elements are zero and were previously set). This
C does not apply to the first reactor. These terms occur because
C all of the outlet component flows from each reactor are the
C inlet flows for the next reactor.

IF ( IR .GT. 1 ) THEN
   DO 825 I = 1,NB
      DO 825 J = 1,NX
825   DFDXU(NB*IR-NB+I,NT*(IR-I)+2+J) = XV(I)
   DO 830 M = 1,NB
830   DFDXU(NB*IR-NB+M,NT*IR=NT*2+2+M) = -1.D0 + XV(M)
ENDIF
C Include effect of variable scaling after entire matrix is filled.

IF(IR .EQ. NR) THEN
   DO 890 I = 1,ND
   DO 890 K = 1,NV
      AAA = 0.0
   DO 880 J = 1,NV
      880 AAA = AAA + DFDXU(I,J)*DXUDX(J,K)
   890 DFDX(I,K) = AAA
ENDIF

C DO 685 I = 1,ND
C 685 WRITE(*,686) (DFDX(I,J),J=1,NV)
C 686 FORMAT(7E11.5)
ENDIF

C Select appropriate columns of DFDX to be in DFDXD and DFDXI

IF( IR .EQ. NR ) THEN
   KD = 0
   KI = 0
   DO 930 L = 1,NV
      IF( IB(L) .EQ. 1 ) THEN
         KD = KD + 1
      DO 910 I = 1,ND
      910 DFDXD(I,KD) = DFDX(I,L)
      ELSE
         KI = KI + 1
      DO 920 I = 1,ND
      920 DFDXI(I,KI) = DFDX(I,L)
      ENDIF
   930 CONTINUE
ENDIF
RETURN
END
APPENDIX B: DATA EXTRACTION PROGRAM

This section displays the 'C' program that was used to extract the desired variables from the FORTRAN output files and print them into a single file for easier access throughout the project.

/* This program extracts the data required for this project from the output files created by the FORTRAN program. This program prompts for the file name, and then reads in the desired data and appends each data set to an output file. */

#include "stdio.h"
void main()
{

    /* VARIABLE DECLARATION */
    char filename[15], strng[100];
    int answer, i, num;
    float fdp, pdp, rv, rt, rp, cost, feed, out, deg;

    /* POINTER DECLARATION */
    FILE *fptr, *ptr;

    /**************** BEGIN MAIN ******************/

    /* POINT TO AN OUTPUT FILE TO WHICH DATA IS APPENDED */
    fptr = fopen("real.txt", "a");

    /* BEGIN LOOP TO EXTRACT DATA FROM FORTRAN OUTPUT FILES */
    do
    {
        /* PROMPT FOR FILE NAME */
        printf("Enter file: ");
```c
scanf("%s", filename);

/* POINT TO THE FILENAME ENTERED BY USER TO READ THE FILE */
ptr = fopen(filename, "r");

fgets(strng, 11, ptr);
fgets(strng, 50, ptr);
fgets(strng, 35, ptr);
fgets(strng, 45, ptr);

/* READ THE 'FEED DEGREE OF POLYMERIZATION' */
fscanf(ptr, "%f", &fdp);
fgets(strng, 50, ptr);

fgets(strng, 34, ptr);

/* READ THE 'PRODUCT DEGREE OF POLYMERIZATION' */
fscanf(ptr, "%f", &pdp);

fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 40, ptr);
fgets(strng, 40, ptr);

/* READ THE 'REACTOR VOLUME' */
fscanf(ptr, "%f", &rv);

/* PRINT REACTOR VOLUME */
fprintf(fptr, "%4.2f", rv);

fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
```
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 40, ptr);

/* READ THE REACTOR TEMP. */
scanf(ptr, "%f", &rt);
/* PRINT THE REACTOR TEMP. TO FILE */
fprintf(fptr, "\t%5.2f", rt);

fgets(strng, 100, ptr);
fgets(strng, 40, ptr);

/* READ THE REACTOR PRESSURE */
scanf(ptr, "%f", &rp);
/* PRINT REACTOR PRESSURE TO FILE */
fprintf(fptr, "\t%f", rp);

fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 100, ptr);
fgets(string, 15, ptr);

/* READ DEG CONTENT */
scanf(ptr, "%f", &deg);

/* PRINT DEG CONTENT TO FILE */
fprintf(fptr, "%.9f", deg);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 40, ptr);
fgets(strng, 40, ptr);
fgets(strng, 40, ptr);

/* READ COST */
scanf(ptr, "%f", &cost);

/* PRINT COST TO FILE */
fprintf(fptr, "%.9f", cost);

/*****************************************************************************/

/* CLOSE FILE AND REOPEN TO READ IN PRIOR VALUES */
fclose(ptr);
ptr = fopen(filename, "r");

fgets(strng, 11, ptr);
fgets(strng, 50, ptr);
fgets(strng, 35, ptr);
fgets(strng, 45, ptr);

/* READ FEED DEGREE OF POLYMERIZATION */
scanf(ptr, "%f", &fdp);
/* PRINT FEED DEGREE OF POLYMERIZATION TO FILE */
fprintf(fptr, "%.6f", fdp);

fgets(strng, 50, ptr);

/* READ PRODUCT DEGREE OF POLYMERIZATION */
scanf(ptr, "%f", &pdp);
/* PRINT PRODUCT DEGREE OF POLYMERIZATION TO FILE */
fprintf(fptr, "%.6f", pdp);

fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100,ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);

/* READ IN ID NUMBER AND FEED TEMPERATURE */
fsscanf(ptr, "%d", &num);
fsscanf(ptr, "%f", &feed);

/* PRINT FEED TEMPERATURE TO FILE */
fprintf(fptr, "\n	%11.8f", feed);

/* LOOP AND EXTRACT EIGHT INGREDIENT FEED CONDITION ID NUMBERS AND VALUES AND PRINT TO FILE */
for(i=0; i<8; i++)
{
    fsscanf(ptr, "%d", &num);
    fsscanf(ptr, "%f", &feed);
    fprintf(fptr, "\t%11.8f", feed);
} /* END LOOP */

fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);
fgets(strng, 100, ptr);

/* READ IN ID NUMBERS, INITIAL GUESS AT REACTOR VOLUME AND TEMPERATURE, AND FINAL REACTOR VOLUME AND TEMPERATURE VALUES. PRINT FINAL REACTOR VOLUME AND TEMPERATURE VALUES TO OUTPUT FILE. */
fgets(strng, 17, ptr);
fsscanf(ptr, "%f", &out);
fprintf(fptr, "\n%11.8f", out);
fsscanf(ptr, "%d", &num);
fsscanf(ptr, "%f", &out);
fsscanf(ptr, "%f", &out);
fsscanf(ptr, "%f", &out);
fsscanf(ptr, "%f", &out);
fprintf(fptr, "\t%11.8f", out);

/* LOOP AND EXTRACT ID NUMBERS, INITIAL GUESS AT OUTPUT FLOWRATES, AND FINAL OUTPUT FLOWRATES. PRINT FINAL OUTPUT FLOWRATE VALUES TO OUTPUT FILE. */
for(i=0; i<8; i++)
{
    fsscanf(ptr, "%d", &num);
    fsscanf(ptr, "%f", &out);
}
fscanf(ptr, "%f", &out);
fprintf(fptr, "t%11.8f", out);
fscanf(ptr, "%f", &out);
}

/* PROMPT TO CONTINUE */
printf("Another?: ");
scanf("%d", &answer);

fclose(ptr);
fprintf(fptr, "n******************************************************************************
********
");
} while (answer == 1); /* CONTINUE IF USER ENTERS '1'. ELSE, END */

fclose(fptr);
} /* END PROGRAM */
APPENDIX C: NEURAL NETWORK VARIABLE HISTOGRAMS

The following section displays the histograms for specific neural network variables, from which it was determined that two networks would be required to adequately model the reactor series.
Ingredient 'ZG' Flowrate Histogram

Ingredient 'FC' Flowrate Histogram
Ingredient 'EC' Output Flowrate Histogram

Ingredient 'ZG' Output Flowrate Histogram
Ingredient 'FC' Output Flowrate Histogram

<table>
<thead>
<tr>
<th>Bin Values</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.023-</td>
<td>400</td>
</tr>
<tr>
<td>0.0464-</td>
<td></td>
</tr>
<tr>
<td>0.0697-</td>
<td></td>
</tr>
<tr>
<td>0.0929-</td>
<td></td>
</tr>
<tr>
<td>0.1161-</td>
<td></td>
</tr>
<tr>
<td>0.1393-</td>
<td></td>
</tr>
<tr>
<td>0.1626-</td>
<td></td>
</tr>
<tr>
<td>0.1858-</td>
<td></td>
</tr>
<tr>
<td>0.2090-</td>
<td></td>
</tr>
</tbody>
</table>

Ingredient 'FD' Output Flowrate Histogram

<table>
<thead>
<tr>
<th>Bin Values</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.012-</td>
<td>400</td>
</tr>
<tr>
<td>0.0241-</td>
<td></td>
</tr>
<tr>
<td>0.0361-</td>
<td></td>
</tr>
<tr>
<td>0.0482-</td>
<td></td>
</tr>
<tr>
<td>0.0602-</td>
<td></td>
</tr>
<tr>
<td>0.0723-</td>
<td></td>
</tr>
<tr>
<td>0.0843-</td>
<td></td>
</tr>
<tr>
<td>0.0964-</td>
<td></td>
</tr>
<tr>
<td>0.1084-</td>
<td></td>
</tr>
</tbody>
</table>
Ingredient 'ZD' Output Flowrate Histogram

Ingredient 'ED' Output Flowrate Histogram
Ingredient 'FG' Output Flowrate Histogram

Bin values:

<table>
<thead>
<tr>
<th>Bin Values</th>
<th># of Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0-1.1</td>
<td>400</td>
</tr>
<tr>
<td>1.1-2.3</td>
<td></td>
</tr>
<tr>
<td>2.3-3.5</td>
<td></td>
</tr>
<tr>
<td>3.5-4.7</td>
<td></td>
</tr>
<tr>
<td>4.7-5.9</td>
<td></td>
</tr>
<tr>
<td>5.9-7.1</td>
<td></td>
</tr>
<tr>
<td>7.1-8.3</td>
<td></td>
</tr>
<tr>
<td>8.3-9.5</td>
<td></td>
</tr>
<tr>
<td>9.5-10.0</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX D: NEURAL NETWORK DEVELOPMENT RESULTS

The following section summarizes the results of the backpropagation artificial neural network development within this research. The first column indicates the number of neurons in the first layer and the second column indicates the number of neurons in the second layer of the network.

<table>
<thead>
<tr>
<th>First layer</th>
<th>Second layer</th>
<th>RMS error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-</td>
<td>0.0550</td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>0.0510</td>
</tr>
<tr>
<td>30</td>
<td>-</td>
<td>0.0495</td>
</tr>
<tr>
<td>45</td>
<td>-</td>
<td>0.0490</td>
</tr>
<tr>
<td>60</td>
<td>-</td>
<td>0.0490</td>
</tr>
<tr>
<td>75</td>
<td>-</td>
<td>0.0500</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0.0578</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>0.0545</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>0.0525</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>0.0535</td>
</tr>
<tr>
<td>20</td>
<td>55</td>
<td>0.5300</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>0.0570</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>0.0515</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>-------</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>0.0515</td>
</tr>
<tr>
<td>30</td>
<td>55</td>
<td>0.0520</td>
</tr>
<tr>
<td>30</td>
<td>70</td>
<td>0.0525</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>0.0545</td>
</tr>
<tr>
<td>40</td>
<td>25</td>
<td>0.0540</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>0.0550</td>
</tr>
<tr>
<td>40</td>
<td>70</td>
<td>0.0525</td>
</tr>
<tr>
<td>55</td>
<td>10</td>
<td>0.0530</td>
</tr>
<tr>
<td>55</td>
<td>55</td>
<td>0.0540</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
<td>0.0525</td>
</tr>
<tr>
<td>70</td>
<td>40</td>
<td>0.0520</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>0.0525</td>
</tr>
</tbody>
</table>
APPENDIX E: NEURAL NETWORK SHELL PROGRAM

The following section displays the neural network shell program which was used in this research.

```c
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<stdlib.h>
#include<math.h>

/** DECLARE FUNCTIONS **/
int rl(float *Yin, float *Yout);
int r2(float *Yin, float *Yout);
void results(int rea, int i, float *input, float *output, float costsum, FILE *fwrite,
FILE *fwrite2);

/** BEGIN MAIN **/
void main()
{

/** VARIABLE DECLARATION **/
float pdp, deg, Vol, costsum;
int YorN, rea, ingredients, variables, i, j;
int totinputs, totoutputs;
float *input, *output, *Yin, *Yout;

char str[81];

/** DECLARE POINTERS **/
FILE *fread, *fwrite, *fwrite2;

****** BEGIN EXECUTION *****/

/******* OPEN PROGRAM INPUT FILE 'NN_INPUT.DAT' AND CREATE OUTPUT FILES ******/
/** 'NNOUT.TXT' AND 'NNSUM.TXT'. READ IN THE INPUT PARAMETERS, AND ******/
/*** DYNAMICALLY ALLOCATE MEMORY FOR SPECIFIC VARIABLE ARRAYS. ******/
/*****************************/

fread = fopen("nn_input.dat", "r");
fwrite = fopen("nnout.txt", "w");
fwrite2 = fopen("nnsum.txt", "w");
```
fgets(str, 80, fread);
scanf(fread, "%d", &ingredients); /* READ NUMBER OF CHEM INGREDIENTS */

fgets(str, 80, fread);
fgets(str, 80, fread);
fscanf(fread, "%d", &variables); /* READ NUMBER OF ADDITIONAL VARIABLES */

totinputs = ingredients + variables + 1; /* DETERMINE TOTAL INPUTS */
totoutputs = ingredients + variables; /* AND OUTPUTS TO NETWORK */

/*** INPUT THE DESIRED NUMBER OF REACTORS IN SERIES TO BE SEARCHED ***/

clrscr();
printf("Enter the number of reactors in series: ");
scanf("%d", &rea);

/cll************** DYNAMICALLY ALLOCATE MEMORY FOR ARRAYS  /
/cll****** INPUT', 'OUTPUT', 'YIN', AND 'YOUT'  /
/cll***************************************************************************/

if(NULL == (input = (float *)calloc((totinputs * rea), sizeof(float))))
{
    puts("Can't get the memory for the data array.");
    abort();
}

if(NULL == (output = (float *)calloc((totoutputs) * rea),
        sizeof(float))))
{
    puts("Can't get the memory for the data array.");
    abort();
}

if(NULL == (Yin = (float *)calloc((totinputs),
       sizeof(float))))
{
    puts("Can't get the memory for the data array.");
    abort();
}

if(NULL == (Yout = (float *)calloc((totoutputs),
        sizeof(float))))
{
    puts("Can't get the memory for the data array.");
abort();
}

fgets(str, 80, fread);
fgets(str, 80, fread);
fgets(str, 80, fread);

/** read initial ingredient flowrates from input file ***/
for(i=0; i<ingredients; i++)
    fscanf(fread, "%f", &input[i + 4]);

fgets(str, 80, fread);
fgets(str, 80, fread);
fgets(str, 80, fread);

/** read initial input temperature */

/** begin search loop. continue with search while user entered variable 'YorN' equals 1. ***/
do {
    clrscr();

    /* prompt for user input on search variables reactor volume, pdp, and deg content for each reactor by looping for number of reactors */
    for(i = 0; i < rea; i++)
        {
            printf("Enter the volume for reactor %d: ", i + 1);
            scanf("%f", &input[(i*12) + 2]);

            printf("Enter the desired pdp for reactor %d: ", i + 1);
            scanf("%f", &input[((i*12) + 0)]);

            printf("Enter the desired deg for reactor %d: ", i + 1);
            scanf("%f", &input[((i*12) + 1)]);
        } /* end search variable input loop */

/** begin evaluation loop */

for(i = 0; i < rea; i++)
    {
        clrscr();

        /* set inputs in proper format and call nn functions */
if(i == 0) /* FIRST REACTOR VARIABLE ASSIGNMENTS */
{
  /* ASSIGN VALUES TO PROPER VARIABLE NAMES FOR NN */
  for(j=0; j<(totinputs); j++)
    Yin[j] = input[j];

  /* CALL REACTOR 1 NEURAL NETWORK */
  r1(Yin, Yout);

  costsum = 0;
}
else /* REACTOR VARIABLE ASSIGNMENTS FOR ALL */
/* REACTORS BESIDES THE FIRST */
{
  /* INPUT TEMP. = REACTOR TEMP FROM PREVIOUS REACTOR */
  input[(i * 12) + 3] = output[((i - 1) * 12) + 1];

  /* SET INPUTS TO REACTOR AS OUTPUTS FROM PREVIOUS */
  /* REACTOR */
  for(j=4; j<(totinputs); j++)
    input[(i * 12) + j] = output[((i - 1) * 12) + (j - 1)];

  /* ASSIGN VALUES TO PROPER VARIABLE NAMES FOR NN */
  for(j=0; j<(totinputs); j++)
    Yin[j] = input[(i * 12) + j];

  /* CALL REACTOR 2 NEURAL NETWORK */
  r2(Yin, Yout);
}

/***** COLLECT THE OUTPUTS FROM THE NEURAL NETWORK *****/
for(j=0; j<(totoutputs); j++)
  output[(i * 12) + j] = Yout[j];

/***** ACCUMULATE COST THROUGH REACTOR SERIES *****/
  costsum = costsum + output[(i * 12) + 2];

/***** CALL 'RESULTS' FUNCTION TO PRINT RESULTS *****/
results(rea, i, input, output, costsum, fwrite, fwrite2);

getch();
}
clrscr();

/* PROMPT TO CONTINUE SEARCH OR NOT */
printf("Another? ");
```c
scanf("%d", &YorN);
}
```
printf("c5 = 		%i09.6f 		 c6 = 		%9.6f\n", input[(i*12) + 8], output[(i*12) + 8]);
printf("c6 = 		%9.6f 		 c7 = 		%9.6f\n", input[(i*12) + 9], output[(i*12) + 9]);
printf("c7 = 		%9.6f 		 c8 = 		%9.6f\n", input[(i*12) + 10], output[(i*12) + 10]);
printf("c8 = 		%9.6f", input[(i*12) + 11]);
printf("\n\n Total cumulative cost = %f", costsum);

/** PRINT DETAILED RESULTS TO OUTPUT FILE **/

fprintf(fwrite, "Results for reactor %d", i+1);
fprintf(fwrite, "\n\n\n\n Volume = %9.6f", input[(i*12) + 2]);
fprintf(fwrite, "\n\n\n\n DEG = %9.6f", input[(i*12) + 1]);
fprintf(fwrite, "\n\n\n\n PDP = %9.6f", input[(i*12) + 0]);
if(i == (rea - 1))
fprintf(fwrite, "\n\n\n\n Total Cost = %f", costsum);

***** PRINT SUMMARY RESULTS TO ANOTHER OUTPUT FILE *****

fprintf(fwrite2, "\n\n\n\n Results for reactor %d", i+1);
fprintf(fwrite2, "\n\n\n\n Volume = %9.6f", input[(i*12) + 2]);
fprintf(fwrite2, "\n\n\n\n DEG = %9.6f", input[(i*12) + 1]);
if(i == (rea - 1))
fprintf(fwrite2, "\n\n\n\n Total Cost = %f", costsum);
} /* END FUNCTION RESULTS */
APPENDIX F: TWO REACTOR SEARCH RESULTS

The following section outlines the search program results on a two reactor series with a target degree of polymerization of 80.

ITERATION 1:
Results for reactor 1
Volume = 7.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 236534672.000000

ITERATION 2:
Results for reactor 1
Volume = 8.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 236129696.000000

ITERATION 3:
Results for reactor 1
Volume = 9.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 235724784.000000

ITERATION 4:
Results for reactor 1
Volume = 10.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 235326352.000000
ITERATION 5:
Results for reactor 1
Volume = 12.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 234578336.000000

ITERATION 6:
Results for reactor 1
Volume = 15.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 233707936.000000

ITERATION 7:
Results for reactor 1
Volume = 20.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 233566992.000000

ITERATION 8:
Results for reactor 1
Volume = 25.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 235809504.000000

ITERATION 9:
Results for reactor 1
Volume = 22.000000  PDP = 4.000000  DEG = 0.700000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 234163280.000000
ITERATION 10:
Results for reactor 1
Volume = 18.000000  PDP =  4.000000  DEG =  0.700000

Results for reactor 2
Volume = 85.000000  PDP =  80.000000  DEG =  1.600000
Total Cost = 233370496.000000

ITERATION 11:
Results for reactor 1
Volume = 18.000000  PDP =  5.000000  DEG =  0.700000

Results for reactor 2
Volume = 85.000000  PDP =  80.000000  DEG =  1.600000
Total Cost = 230415392.000000

ITERATION 12:
Results for reactor 1
Volume = 18.000000  PDP =  6.000000  DEG =  0.700000

Results for reactor 2
Volume = 85.000000  PDP =  80.000000  DEG =  1.600000
Total Cost = 232899488.000000

ITERATION 13:
Results for reactor 1
Volume = 18.000000  PDP =  5.000000  DEG =  0.600000

Results for reactor 2
Volume = 85.000000  PDP =  80.000000  DEG =  1.600000
Total Cost = 231327584.000000

ITERATION 14:
Results for reactor 1
Volume = 18.000000  PDP =  5.000000  DEG =  0.800000

Results for reactor 2
Volume = 85.000000  PDP =  80.000000  DEG =  1.600000
Total Cost = 229528480.000000
ITERATION 15:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 0.900000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 228651184.000000

ITERATION 16:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.000000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 227770992.000000

ITERATION 17:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.200000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 225965648.000000

ITERATION 18:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.400000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 224061728.000000

ITERATION 19:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.500000

Results for reactor 2
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 223064624.000000
ITERATION 20:  
Results for reactor 1  
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2  
Volume = 85.000000  PDP = 80.000000  DEG = 1.600000  
Total Cost = 222035600.000000

ITERATION 21:  
Results for reactor 1  
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2  
Volume = 87.000000  PDP = 80.000000  DEG = 1.600000  
Total Cost = 224425776.000000

ITERATION 22:  
Results for reactor 1  
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2  
Volume = 83.000000  PDP = 80.000000  DEG = 1.600000  
Total Cost = 219208512.000000

ITERATION 23:  
Results for reactor 1  
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2  
Volume = 80.000000  PDP = 80.000000  DEG = 1.600000  
Total Cost = 213956208.000000

ITERATION 24:  
Results for reactor 1  
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2  
Volume = 75.000000  PDP = 80.000000  DEG = 1.600000  
Total Cost = 201620272.000000
ITERATION 25:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2
Volume = 70.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 183264960.000000

ITERATION 26:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2
Volume = 60.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 126640624.000000

ITERATION 27:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2
Volume = 50.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 65335332.000000

ITERATION 28:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2
Volume = 40.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 27354684.000000

ITERATION 29:
Results for reactor 1
Volume = 18.000000  PDP = 5.000000  DEG = 1.600000

Results for reactor 2
Volume = 20.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 2120516.250000
ITERATION 30:
Results for reactor 1
Volume = 18.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 10.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -1256560.250000

ITERATION 31:
Results for reactor 1
Volume = 18.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 5.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -2190738.000000

ITERATION 32:
Results for reactor 1
Volume = 18.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 3.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -2480226.500000

ITERATION 33:
Results for reactor 1
Volume = 18.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -2732610.000000

ITERATION 34:
Results for reactor 1
Volume = 16.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -2274897.500000
ITERATION 35:
Results for reactor 1
Volume = 20.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -3123849.750000

ITERATION 36:
Results for reactor 1
Volume = 25.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -3831654.500000

ITERATION 37:
Results for reactor 1
Volume = 30.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -4179452.250000

ITERATION 38:
Results for reactor 1
Volume = 40.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -3716282.750000

ITERATION 39:
Results for reactor 1
Volume = 35.000000 PDP = 5.000000 DEG = 1.600000

Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -4167732.250000
ITERATION 40:
Results for reactor 1
Volume = 30.000000 PDP = 6.000000 DEG = 1.600000
Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -2482092.500000

ITERATION 41:
Results for reactor 1
Volume = 30.000000 PDP = 4.000000 DEG = 1.600000
Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -4733819.500000

ITERATION 42:
Results for reactor 1
Volume = 30.000000 PDP = 3.000000 DEG = 1.600000
Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -3159360.750000

ITERATION 43:
Results for reactor 1
Volume = 30.000000 PDP = 4.000000 DEG = 1.500000
Results for reactor 2
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -4191388.250000

ITERATION 44:
Results for reactor 1
Volume = 30.000000 PDP = 4.000000 DEG = 1.600000
Results for reactor 2
Volume = 2.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -4651634.000000
ITERATION 45:
Results for reactor 1
Volume = 32.000000  PDP = 4.000000  DEG = 1.600000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4493670.000000

ITERATION 46:
Results for reactor 1
Volume = 28.000000  PDP = 4.000000  DEG = 1.600000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4832650.000000

ITERATION 47:
Results for reactor 1
Volume = 25.000000  PDP = 4.000000  DEG = 1.600000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4763733.000000

ITERATION 48:
Results for reactor 1
Volume = 27.000000  PDP = 4.000000  DEG = 1.600000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4836619.500000

ITERATION 49:
Results for reactor 1
Volume = 26.000000  PDP = 4.000000  DEG = 1.600000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4813125.000000
ITERATION 50:
Results for reactor 1
Volume = 27.000000  PDP = 5.000000  DEG = 1.600000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4012860.750000

ITERATION 51:
Results for reactor 1
Volume = 27.000000  PDP = 3.000000  DEG = 1.600000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4439182.000000

ITERATION 52:
Results for reactor 1
Volume = 27.000000  PDP = 4.000000  DEG = 1.500000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4384025.000000

ITERATION 53:
Results for reactor 1
Volume = 27.000000  PDP = 4.000000  DEG = 1.600000
Results for reactor 2
Volume = 2.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4752039.500000

ITERATION 54:
Results for reactor 1
Volume = 28.000000  PDP = 4.000000  DEG = 1.600000
Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4832650.000000
ITERATION 55:
Results for reactor 1
Volume = 26.000000  PDP = 4.000000  DEG = 1.600000

Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4813125.000000

ITERATION 56:
Results for reactor 1
Volume = 27.000000  PDP = 4.000000  DEG = 1.600000

Results for reactor 2
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4836619.500000
APPENDIX G: THREE REACTOR TEST RESULTS

The following section outlines the search program results on a three reactor series with a target degree of polymerization of 80.

--------------------------------------------------------------------------------------------------------------------------

ITERATION 1:
Results for reactor 1
Volume = 5.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 113734368.000000

ITERATION 2:
Results for reactor 1
Volume = 4.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 120729672.000000

ITERATION 3:
Results for reactor 1
Volume = 6.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 107352616.000000
ITERATION 4:
Results for reactor 1
Volume = 8.000000 PDP = 8.000000 DEG = 0.300000

Results for reactor 2
Volume = 5.000000 PDP = 14.000000 DEG = 0.700000

Results for reactor 3
Volume = 38.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = 96209040.000000

ITERATION 5:
Results for reactor 1
Volume = 12.000000 PDP = 8.000000 DEG = 0.300000

Results for reactor 2
Volume = 5.000000 PDP = 14.000000 DEG = 0.700000

Results for reactor 3
Volume = 38.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = 79037904.000000

ITERATION 6:
Results for reactor 1
Volume = 20.000000 PDP = 8.000000 DEG = 0.300000

Results for reactor 2
Volume = 5.000000 PDP = 14.000000 DEG = 0.700000

Results for reactor 3
Volume = 38.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = 57490864.000000

ITERATION 7:
Results for reactor 1
Volume = 30.000000 PDP = 8.000000 DEG = 0.300000

Results for reactor 2
Volume = 5.000000 PDP = 14.000000 DEG = 0.700000
Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 42765308.000000

ITERATION 8:
Results for reactor 1
Volume = 40.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 34671464.000000

ITERATION 9:
Results for reactor 1
Volume = 60.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 30081072.000000

ITERATION 10:
Results for reactor 1
Volume = 80.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 65710996.000000

ITERATION 11:
Results for reactor 1
Volume = 70.000000  PDP = 8.000000  DEG = 0.300000
Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 37149412.000000

ITERATION 12:
Results for reactor 1
Volume = 65.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 32196548.000000

ITERATION 13:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 29705348.000000

ITERATION 14:
Results for reactor 1
Volume = 50.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 30485466.000000
ITERATION 15:
Results for reactor 1
Volume = 55.000000  PDP = 7.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 38344592.000000

ITERATION 16:
Results for reactor 1
Volume = 55.000000  PDP = 9.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 31028564.000000

ITERATION 17:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.300000

Results for reactor 2
Volume = 5.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 29054012.000000

ITERATION 18:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.400000
Results for reactor 2
Volume = 5.000000 PDP = 14.000000 DEG = 0.700000

Results for reactor 3
Volume = 38.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = 28476164.000000

ITERATION 19:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000 DEG = 0.500000

Results for reactor 2
Volume = 5.000000 PDP = 14.000000 DEG = 0.700000

Results for reactor 3
Volume = 38.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = 27955186.000000

ITERATION 20:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000 DEG = 0.600000

Results for reactor 2
Volume = 5.000000 PDP = 14.000000 DEG = 0.700000

Results for reactor 3
Volume = 38.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = 27478054.000000

ITERATION 21:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 4.000000 PDP = 14.000000 DEG = 0.700000

Results for reactor 3
Volume = 38.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = 27092166.000000
ITERATION 23:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 3.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 26763222.000000

ITERATION 24:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 2.000000  PDP = 14.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 26486116.000000

ITERATION 25:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 15.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 26256156.000000

ITERATION 26:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 15.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 26469950.000000

ITERATION 27:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000
Results for reactor 2
Volume = 1.000000  PDP = 13.000000  DEG = 0.700000
Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 26041736.000000

ITERATION 28:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000
Results for reactor 2
Volume = 1.000000  PDP = 12.000000  DEG = 0.700000
Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 25826726.000000

ITERATION 29:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000
Results for reactor 2
Volume = 1.000000  PDP = 11.000000  DEG = 0.700000
Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 25611176.000000

ITERATION 30:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000
Results for reactor 2
Volume = 1.000000  PDP = 10.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 25395112.000000

ITERATION 31:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 24961508.000000

ITERATION 32:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.800000

Results for reactor 3
Volume = 38.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 25201796.000000

ITERATION 33:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 37.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 22374844.000000
ITERATION 34:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000   DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000   DEG = 0.700000

Results for reactor 3
Volume = 36.000000 PDP = 80.000000   DEG = 1.600000
Total Cost = 19982314.000000

ITERATION 35:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000   DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000   DEG = 0.700000

Results for reactor 3
Volume = 33.000000 PDP = 80.000000   DEG = 1.600000
Total Cost = 13851148.000000

ITERATION 36:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000   DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000   DEG = 0.700000

Results for reactor 3
Volume = 30.000000 PDP = 80.000000   DEG = 1.600000
Total Cost = 9053564.000000

ITERATION 37:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000   DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000   DEG = 0.700000
Results for reactor 3
Volume = 25.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = 3303757.500000

ITERATION 38:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 20.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -461246.000000

ITERATION 39:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 15.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -2920059.750000

ITERATION 40:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 10.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -4524472.500000

ITERATION 41:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 5.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -5571701.500000

ITERATION 42:
Results for reactor 1
Volume = 55.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -6141811.000000

ITERATION 43:
Results for reactor 1
Volume = 50.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -5642983.000000

ITERATION 44:
Results for reactor 1
Volume = 60.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -6224964.000000

ITERATION 45:
Results for reactor 1
Volume = 70.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -4457621.000000

ITERATION 46:
Results for reactor 1
Volume = 65.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -5755668.000000

ITERATION 47:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -6141811.000000

ITERATION 48:
Results for reactor 1
Volume = 60.000000 PDP = 7.000000 DEG = 0.700000
Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -2271615.250000

ITERATION 49:
Results for reactor 1
Volume = 60.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -5898356.500000

ITERATION 50:
Results for reactor 1
Volume = 60.000000 PDP = 8.000000 DEG = 0.600000

Results for reactor 2
Volume = 2.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -5878826.500000

ITERATION 51:
Results for reactor 1
Volume = 60.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 9.000000 DEG = 0.700000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -6143999.000000
ITERATION 52:
Results for reactor 1
Volume = 60.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.800000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -6051625.500000

ITERATION 53:
Results for reactor 1
Volume = 60.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
Volume = 2.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -6103260.500000

ITERATION 54:
Results for reactor 1
Volume = 55.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -6141811.000000

ITERATION 55:
Results for reactor 1
Volume = 65.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000

Results for reactor 3
ITERATION 56:
Results for reactor 1
Volume = 60.000000 PDP = 7.000000 DEG = 0.700000
Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000
Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -2271615.250000

ITERATION 57:
Results for reactor 1
Volume = 60.000000 PDP = 8.000000 DEG = 0.600000
Results for reactor 2
Volume = 1.000000 PDP = 8.000000 DEG = 0.700000
Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -5898356.500000

ITERATION 58:
Results for reactor 1
Volume = 60.000000 PDP = 8.000000 DEG = 0.700000
Results for reactor 2
Volume = 2.000000 PDP = 8.000000 DEG = 0.700000
Results for reactor 3
Volume = 1.000000 PDP = 80.000000 DEG = 1.600000
Total Cost = -5878826.500000

ITERATION 59:
Results for reactor 1
Volume = 60.000000 PDP = 8.000000 DEG = 0.700000
Results for reactor 2
Volume = 1.000000  PDP = 9.000000  DEG = 0.700000

Results for reactor 3
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -6143999.000000

ITERATION 60:
Results for reactor 1
Volume = 60.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.800000

Results for reactor 3
Volume = 1.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -6051625.500000

ITERATION 61:
Results for reactor 1
Volume = 60.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.700000

Results for reactor 3
Volume = 2.000000  PDP = 80.000000  DEG = 1.600000
Total Cost = -6103260.500000
APPENDIX H: FIVE REACTOR TEST RESULTS

The following section outlines the search program results on a five reactor series with a target degree of polymerization of 120.

ITERATION 1:
Results for reactor 1
Volume = 6.000000  PDP = 4.000000  DEG = 0.100000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 146468416.000000

ITERATION 2:
Results for reactor 1
Volume = 4.000000  PDP = 4.000000  DEG = 0.100000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 166862784.000000

ITERATION 3:
Results for reactor 1
Volume = 8.000000  PDP = 4.000000  DEG = 0.100000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 132543680.000000

ITERATION 4:

Results for reactor 1
Volume = 10.000000  PDP = 4.000000  DEG = 0.100000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 125256592.000000

ITERATION 5:

Results for reactor 1
Volume = 15.000000  PDP = 4.000000  DEG = 0.100000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000
Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 138292816.000000

ITERATION 6:
Results for reactor 1
Volume = 12.000000  PDP = 4.000000  DEG = 0.100000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 124778016.000000

ITERATION 7:
Results for reactor 1
Volume = 12.000000  PDP = 3.000000  DEG = 0.100000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 327008576.000000
ITERATION 8:
Results for reactor 1
Volume = 12.000000 PDP = 5.000000 DEG = 0.100000

Results for reactor 2
Volume = 5.000000 PDP = 8.000000 DEG = 0.200000

Results for reactor 3
Volume = 5.000000 PDP = 13.000000 DEG = 0.200000

Results for reactor 4
Volume = 2.000000 PDP = 43.000000 DEG = 0.500000

Results for reactor 5
Volume = 6.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = 9921500.000000

ITERATION 9:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.100000

Results for reactor 2
Volume = 5.000000 PDP = 8.000000 DEG = 0.200000

Results for reactor 3
Volume = 5.000000 PDP = 13.000000 DEG = 0.200000

Results for reactor 4
Volume = 2.000000 PDP = 43.000000 DEG = 0.500000

Results for reactor 5
Volume = 6.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = 9374115.000000

ITERATION 10:
Results for reactor 1
Volume = 12.000000 PDP = 7.000000 DEG = 0.100000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 20113142.000000

ITERATION 11:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 5.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 7600388.000000

ITERATION 12:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 6.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000
Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 9109296.000000

ITERATION 13:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000
Results for reactor 2
Volume = 4.000000  PDP = 8.000000  DEG = 0.200000
Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000
Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000
Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 6220222.500000

ITERATION 14:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000
Results for reactor 2
Volume = 3.000000  PDP = 8.000000  DEG = 0.200000
Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000
Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000
Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 4957681.000000

ITERATION 15:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 2.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 3802635.000000

ITERATION 16:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 1.000000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 2745689.000000

ITERATION 17:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 8.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000
Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 2251315.250000

ITERATION 18:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 9.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 2329299.750000

ITERATION 19:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 7.000000  DEG = 0.200000

Results for reactor 3
Volume = 5.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 2173874.500000

ITERATION 20:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 5.000000 PDP = 13.000000 DEG = 0.200000

Results for reactor 4
Volume = 2.000000 PDP = 43.000000 DEG = 0.500000

Results for reactor 5
Volume = 6.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = 2097031.750000

ITERATION 21:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.300000

Results for reactor 3
Volume = 5.000000 PDP = 13.000000 DEG = 0.200000

Results for reactor 4
Volume = 2.000000 PDP = 43.000000 DEG = 0.500000

Results for reactor 5
Volume = 6.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = 2242109.500000

ITERATION 22:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 6.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 2523631.500000

ITERATION 23:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 4.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 1719914.625000

ITERATION 24:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 3.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 1388206.875000

ITERATION 25:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 2.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 1098059.875000

ITERATION 26:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 1.000000  PDP = 13.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 845992.000000
ITERATION 27:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 13.000000 DEG = 0.200000

Results for reactor 4
Volume = 2.000000 PDP = 43.000000 DEG = 0.500000

Results for reactor 5
Volume = 6.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = 733242.375000

ITERATION 28:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 14.000000 DEG = 0.200000

Results for reactor 4
Volume = 2.000000 PDP = 43.000000 DEG = 0.500000

Results for reactor 5
Volume = 6.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = 797341.375000

ITERATION 29:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000
Results for reactor 3
Volume = 0.500000  PDP = 12.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 668992.250000

ITERATION 30:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 10.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 540104.125000

ITERATION 31:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 7.000000  DEG = 0.200000

Results for reactor 4
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 345841.000000

ITERATION 32:  
Results for reactor 1  
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000  
Results for reactor 2  
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000  
Results for reactor 3  
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000  
Results for reactor 4  
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000  
Results for reactor 5  
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000  
Total Cost = 280837.125000

ITERATION 33:  
Results for reactor 1  
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000  
Results for reactor 2  
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000  
Results for reactor 3  
Volume = 0.500000  PDP = 6.000000  DEG = 0.300000  
Results for reactor 4  
Volume = 2.000000  PDP = 43.000000  DEG = 0.500000  
Results for reactor 5  
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000  
Total Cost = 61982.000000

ITERATION 34:  
Results for reactor 1  
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000
Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 3.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 417729.000000

ITERATION 35:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 1.000000  PDP = 43.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 165925.500000

ITERATION 36:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
ITERATION 37:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 45.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 184843.750000

ITERATION 38:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 40.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = 13603.375000
ITERATION 39:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 35.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -155900.000000

ITERATION 40:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 25.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -489285.375000

ITERATION 41:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000
Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 15.000000 DEG = 0.500000

Results for reactor 5
Volume = 6.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = -813812.500000

ITERATION 42:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 10.000000 DEG = 0.500000

Results for reactor 5
Volume = 6.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = -971990.000000

ITERATION 43:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 8.000000 DEG = 0.500000
Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -1034366.000000

ITERATION 44:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.500000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -1096179.750000

ITERATION 45:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.600000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -925040.000000

ITERATION 46:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000
Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.400000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -1267936.750000

ITERATION 47:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.300000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -1440301.750000

ITERATION 48:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000
Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 6.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -1613251.250000

ITERATION 49:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 7.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -1485729.500000

ITERATION 50:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 5.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -1730672.000000
ITERATION 51:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 5
Volume = 4.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = -1838791.250000

ITERATION 52:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 5
Volume = 3.000000 PDP = 120.000000 DEG = 1.600000
Total Cost = -1938345.250000

ITERATION 53:
Results for reactor 1
Volume = 12.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 2.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -2030014.750000

ITERATION 54:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 1.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -2114421.500000

ITERATION 55:
Results for reactor 1
Volume = 12.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000
Results for reactor 5
Volume = 0.500000  PDP = 120.000000  DEG = 1.600000
Total Cost = -2154072.000000

ITERATION 56:
Results for reactor 1
Volume = 13.000000  PDP = 6.000000  DEG = 0.200000
Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000
Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.500000
Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000
Results for reactor 5
Volume = 0.500000  PDP = 120.000000  DEG = 1.600000
Total Cost = -2993731.750000

ITERATION 57:
Results for reactor 1
Volume = 15.000000  PDP = 6.000000  DEG = 0.200000
Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000
Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000
Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000
Results for reactor 5
Volume = 0.500000  PDP = 120.000000  DEG = 1.600000
Total Cost = -6045706.000000

ITERATION 58:
Results for reactor 1
Volume = 20.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 5
Volume = 0.500000 PDP = 120.000000 DEG = 1.600000
Total Cost = -10907512.000000

ITERATION 59:
Results for reactor 1
Volume = 30.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 5
Volume = 0.500000 PDP = 120.000000 DEG = 1.600000
Total Cost = -15651308.000000

ITERATION 60:
Results for reactor 1
Volume = 40.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000
Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 0.500000  PDP = 120.000000  DEG = 1.600000
Total Cost = -10798989.000000

ITERATION 61:
Results for reactor 1
Volume = 35.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 0.500000  PDP = 120.000000  DEG = 1.600000
Total Cost = -15058676.000000

ITERATION 62:
Results for reactor 1
Volume = 25.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 0.500000  PDP = 120.000000  DEG = 1.600000
Total Cost = -14100042.000000
ITERATION 63:
Results for reactor 1
Volume = 30.000000  PDP = 5.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 0.500000  PDP = 120.000000  DEG = 1.600000
Total Cost = -1126878.500000

ITERATION 64:
Results for reactor 1
Volume = 30.000000  PDP = 6.000000  DEG = 0.100000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 0.500000  PDP = 120.000000  DEG = 1.600000
Total Cost = -14007652.000000

ITERATION 65:
Results for reactor 1
Volume = 30.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 1.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 5
Volume = 0.500000 PDP = 120.000000 DEG = 1.600000
Total Cost = -15444014.000000

ITERATION 66:
Results for reactor 1
Volume = 30.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 1.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 5
Volume = 0.500000 PDP = 120.000000 DEG = 1.600000
Total Cost = -15591331.000000

ITERATION 67:
Results for reactor 1
Volume = 30.000000 PDP = 6.000000 DEG = 0.200000

Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000

Results for reactor 4
Volume = 1.000000 PDP = 6.000000 DEG = 0.200000
Results for reactor 5
Volume = 0.500000 PDP = 120.000000 DEG = 1.600000
Total Cost = -15547539.000000

ITERATION 68:
Results for reactor 1
Volume = 30.000000 PDP = 6.000000 DEG = 0.200000
Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000
Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000
Results for reactor 4
Volume = 0.500000 PDP = 7.000000 DEG = 0.200000
Results for reactor 5
Volume = 0.500000 PDP = 120.000000 DEG = 1.600000
Total Cost = -15618788.000000

ITERATION 69:
Results for reactor 1
Volume = 30.000000 PDP = 6.000000 DEG = 0.200000
Results for reactor 2
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000
Results for reactor 3
Volume = 0.500000 PDP = 6.000000 DEG = 0.200000
Results for reactor 4
Volume = 0.500000 PDP = 6.000000 DEG = 0.300000
Results for reactor 5
Volume = 0.500000 PDP = 120.000000 DEG = 1.600000
Total Cost = -15474404.000000

ITERATION 70:
Results for reactor 1
Volume = 30.000000  PDP = 6.000000  DEG = 0.200000

Results for reactor 2
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 3
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 4
Volume = 0.500000  PDP = 6.000000  DEG = 0.200000

Results for reactor 5
Volume = 1.000000  PDP = 120.000000  DEG = 1.600000
Total Cost = -15613993.000000
ABSTRACT

This paper describes a neural network approach to optimize the capital plus operating cost of a series of reactors to produce PET to a target degree of polymerization (DP) and quality (DEG). Results from a non-linear optimization solution to a single reactor are used to train two 2-layer neural networks. A shell program in C code links the neural networks into an overall series and calls an unconstrained search algorithm to identify the optimal parameter values for each reactor. Two, three, and five reactor test problems are generated and the results are compared to those in research by Pattalanchinti [94]. However, the large training RMS error per reactor precludes this as the sole method for reactor optimization.