2003

Hybrid neural networks models for a membrane reactor

Mohammed Al-Yemni
West Virginia University

HYBRID NEURAL NETWORKS MODELS FOR A MEMBRANE REACTOR

by

Mohammed Al-Yemni

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Chemical Engineering

Approved by

Dr. Ray Y. K. Yang, Committee Chairperson
Dr. Eung H. Cho
Dr. Powsiri Klinkhachorn

Department of Chemical Engineering

Morgantown, West Virginia
2003

Keywords: Neural Networks, Hybrid Model, Membrane Reactor

Abstract

Hybrid Neural Networks Models for a
Membrane Reactor

Mohammed Al-Yemni

Artificial neural networks (ANN) have become an established discipline and have gained extensive interest within chemical engineering. In recent years, research effort has focused on the use of "hybrid artificial neural networks" (HANN) models that combine both the deterministic and the ANN elements. Several methods have been proposed for combining ANN with first principle relations. In this thesis, a new hybrid scheme, which is similar to that developed by Kasprow for a space-independent and time-dependent fed-batch microbial reactor, was developed for a space-dependent steady-state enzymatic reactor. This scheme combines ANN with mass balances and assumed rate expressions. It was shown that this new hybrid scheme performed significantly better than both black-box ANN model and the hybrid ANN with only mass balance equations. An enzymatic tubular membrane reactor (TMR) was selected as a case study due to the availability of a reliable deterministic/computational model, which can provide simulated process data as needed, as well as its potential industrial importance. Also, two modeling schemes were developed, a fully 'black box' model (BANN), based on ANN technique only, and a simple hybrid model, combining ANN with mass balances (HANN1). Qualitative and quantitative comparisons of the predicted profiles of the above three modeling schemes indicated that the new hybrid scheme (HANN2) performed better than the other two schemes. As a result of adding biochemical knowledge, in the form of mass balances and simplified rate expressions, the new hybrid scheme allowed the process data to be interpolated and extrapolated more accurately.

Acknowledgements

I would like to thank my research advisor, Dr. Ray Yang, for his guidance, encouragement, corrections, and patience. I would also like to thank my other committee members Dr. Eung Cho and Dr. Powsiri Klinkhachorn for agreeing to be on my committee and scheduling their valuable time. Their guidance has been a good source of improvement in my work.

I would like to express my deep appreciation to my family for their support and patience. I would also like to thank the management of Saudi Basic Industries Corp. (SABIC) for their support in this endeavor.

Last but not the least I would like to thank my friends and the faculty of Chemical Engineering Department, who are responsible for the knowledge that I gained over the years of my stay at West Virginia University.

Table of Contents

List of Figures

List of Tables

Nomenclature

α	Regularization parameter
β	Regularization parameter
γ	Effective number of parameters
μ	Training algorithm tunable parameter
$\mathbf{a}$	Input vector to ANN
a_i	Output calculated via HANN or BANN models.
$\boldsymbol{b}$	Output vector from ANN
BANN	Black box ANN model
C_B	Concentration of cellobiose in lumen (tube) side, $g\ l^{-1}$
$\bar{C}_B$	Concentration of cellobiose in shell side, $g\ l^{-1}$
C_G	Concentration of glucose in lumen (tube) side, $g\ l^{-1}$
$\bar{C}_G$	Concentration of gulcose in shell side, $g\ l^{-1}$
C_S	Concentration of cellulose, g l^{-1}
C_{S0}	Feed substrate concentration, g l^{-1}
$\boldsymbol{e}$	Vector of ANN errors
E_D	Sum squared training set data errors

E_T	Sum squared testing set data errors
E_W	Sum squared weight
E_0	Feed enzyme concentration, g l^{-1}
F	Objective function
G	An approximation to the Hessian matrix **H**
H	Hessian matrix
HANN1	First hybrid ANN model
HANN2	Second hybrid ANN model
I	Identity matrix
J	Jacobian matrix
K_i	Inhibition constant (cellobiose), g l^{-1}
K_i'	Inhibition constant (glucose), g l^{-1}
K_m	Michaelis-Menten constant (cellobiose), g l^{-1}
K_m'	Michaelis-Menten constant (glucose), g l^{-1}
N	Number of data sets
NSE	Normalized root mean square error
MPE	Median percent error
L	Length of TMR, cm
L_P	Hydraulic permeability, l g^{-1} min^{-1}
p	Input vector

$\mathbf{p_n}$	Normalized vector
P	Pressure at any distance z on the lumen side, g cm^{-2}
P_F	Pressure at the entrance of the module, g cm^{-2}
P_P	Pressure on the shell side, g cm^{-2}
P_R	Pressure at the exit of the module, g cm^{-2}
ΔP_T	Transmembrane pressure drop, g cm^{-2}
r_i	Rate of formation of chemical species i, g l^{-1} min^{-1}
r_m	Maximum reaction rate (cellibiose), g l^{-1} min^{-1}
$r_m^{'}$	Maximum reaction rate (glucose), g l^{-1} min^{-1}
$\sum r_i$	Net rate of formation of g cm^{-2} species i, g l^{-1} min^{-1}
R_1	Membrane tube (lumen) radius, cm
R_2	Inside radius of the reactor (shell side), cm
TMR	Tubular membrane reactor
S	Number of hidden layer
T_j	Threshold of node j
V	Volumetric flow rate at any distance z on the lumen side, l min^{-1}
$\overline{V}$	Volumetric flow rate at any distance z on the shell side, l min^{-1}
V_F	Volumetric flow rate at the entrance of the reactor, l min^{-1}
$\boldsymbol{w_k}$	vector of current weight and biases

W_{ij}	Weight factor of variable i in node j
X_j	Total activation of a node j

1. Introduction

In processes involving chemical and biochemical reactions, mathematical models used for reactor design, simulation and optimization are generally deterministic ones that are developed based on first principles. Undoubtedly, a deterministic model is of advantage for easy analysis and reliable extrapolation. However, the development of such a model that is reliable and accurate is usually difficult, due to the complexity of coupled reaction and transport phenomena usually involved in such processes. For this reason, one of the most difficult problems in the control and optimization of biotechnological processes is the construction of reliable models of the system. In addition, due to economic and time constraints, in most cases, reliable deterministic models based on fundamental principles and detailed kinetic studies are not readily available. Thus, it would be of great advantage to find some simple and rapid ways of describing biochemical processes, which are accurately enough for optimization and control [12].

In recent years several methods have been proposed to achieve this goal. One of them is the use of artificial neural networks (ANN), which offers a tool for direct use of process data to generate input-output relationships [1]. ANN has become an established discipline and has gained extensive interest within chemical engineering. Most chemical engineering processes are non-linear and are too complex to be modeled by conventional modeling and simulation techniques. ANN, on the other hand, overcomes the limitations of the conventional approach by extracting the desired information directly from the process data.

Another alternative for modeling chemical and biochemical process is the use of hybrid ANN (HANN) models, in which the aspects of the process whose quantitative behavior is well understood are described by deterministic mathematical equations, while the rest are described by ANN [1,17]. These models are expected to perform better than black-box ANN (BANN) models, in which only ANN, but not deterministic equations, is involved, since generalization and extrapolation are confined only to the uncertain parts of the process, and the basic model is always consistent with first principles.

The main objective of this work is to test a new hybrid neural networks model that combines ANN with mass balances and assumed simplified rate-expressions. The prediction of this model will be compared to the predictions of a black box ANN model and a hybrid ANN model with only mass balances equations included as a first-principle part. An enzymatic tubular membrane reactor (TMR) will be used as the "base process" for studying these modeling approaches.

At first, a deterministic model of this process (reactor) was used to generate process data. This step is described in detail in the first two sections of chapter 3 (3.1 and 3.2). Then, a fully 'black box' model, based on the ANN technique, was developed using just the process data. No information about the process was included in this model. The development of this model is described in section 3.4 and the performance of this modeling approach is evaluated in chapter 4. After that, first-principle information in the form of mass balances equations (ODEs) was introduced separately into the 'black-box' model to generate the first hybrid model (HANN1). In HANN1 the ANN was used to predict rate of reactions. The development of this model is described in section 3.5 and

the performance of this modeling approach is evaluated in chapter 4. The second hybrid scheme was developed using a new hybrid scheme developed by Kasprow [17], called "hybrid neural networks/parameters model". This hybrid scheme combines ANN with mass balances and assumed rate expressions. In order to test the superiority of this new scheme, two models were developed (HANN2a and HANN2b) using smoothed and non-smoothed data. The development of the second hybrid scheme (HANN2) is described in section 3.6, and its comparison with other ANN schemes is presented in chapter 4.

2. Literature Review

2.1 ANN

Artificial neural networks (ANN) are computational systems whose architecture and operation are inspired from our knowledge about biological neural cells (neurons) in the brain. ANN grew out of research in artificial intelligence; specifically, attempts to mimic the fault-tolerance and capacity to learn of biological neural systems by modeling the low-level structure of the brain. Although ANN have been around since the late 1950's, it was not until the mid-1980's that algorithms became sophisticated enough for general applications [3]

In recent years ANN have emerged as a practical technology, with successful applications in many fields. ANN are applicable in virtually every situation in which a relationship between the input and output variables exists, even when that relationship is very complex [13].

ANN have found commercial applications in a variety of areas in bioprocessing and chemical engineering. Some examples include product design, formulation and manufacturing; process monitoring and diagnosis; process modeling; process control; and process optimization.

2.2 Node of ANN

ANN consists of massively interconnected simple processing elements, known as "neurons" or "nodes". Therefore, the starting point for any kind of ANN analyses are a model node whose behavior follows closely to our understanding of how real neurons work. Most of materials presented in this section are taken form Baughman and Liu [3]. The phrase “node” will be used in lieu of others throughout.

Figure 2.1 summarizes, as an example, the basic features of a node using five input variables a_1, a_2,….,a_5 [3].

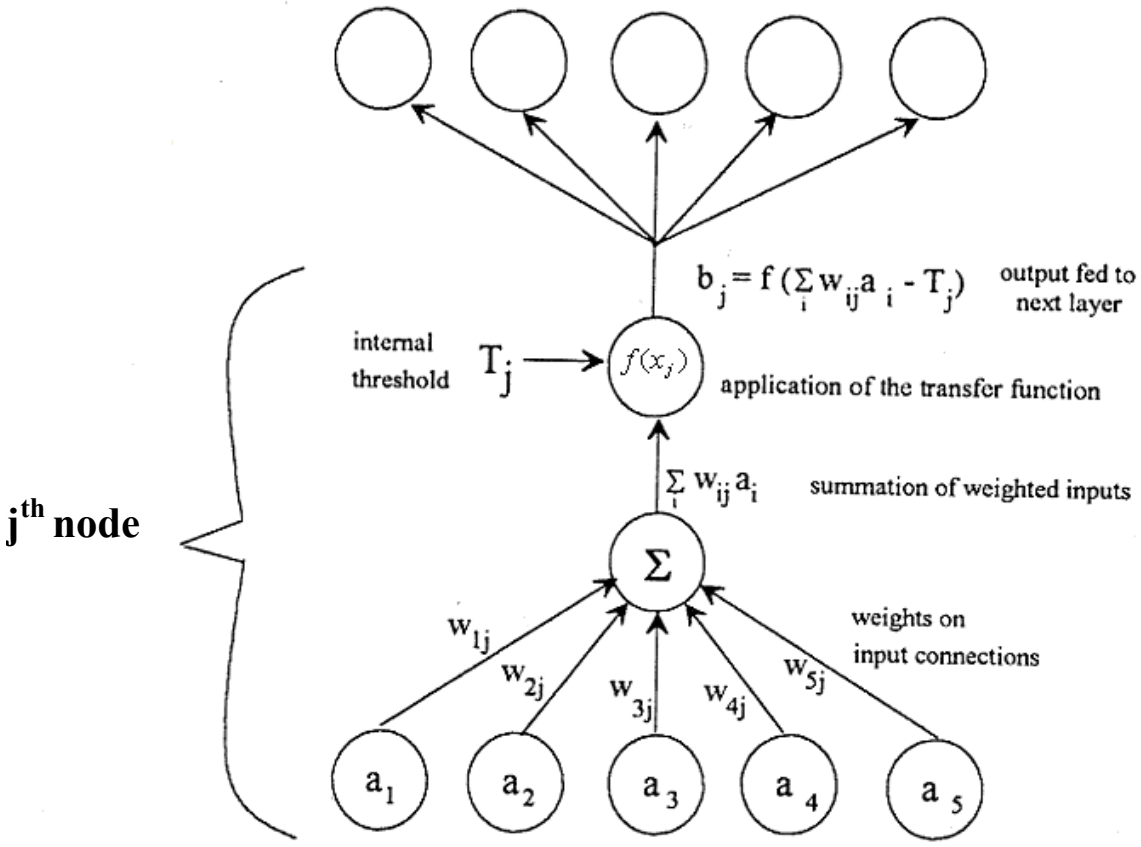

Figure 2.1. Summary of a node anatomy

As seen, the inputs to the j^{th} node are represented as an input vector, $\mathbf{a}$, with components a_i (i =1 to 5). The node manipulates these inputs, or activities, to give the output, b_j, which can then form the part of the input to other nodes. Every input is multiplied by its corresponding weight factor W_{ij} and the node uses this weighted input to perform further calculations. Weight factors can have either an inhibitory or an excitatory effect. If we adjust W_{ij} such that $W_{ij}a_i$ is positive (and preferably large), we tend to excite the node. If $W_{ij}a_i$ is negative, it inhibits the node. Finally, if $W_{ij}a_i$ is very small in magnitude relative to other signals, the input signal a_i will have little or no effect.

The next important factor governing the output from a node is the internal threshold . The internal threshold for the j^{th} node, denoted T_j, controls activation of that node. T_j is also known as "bias". The node calculates all its W_{ij} a_i's, sums the terms together, and then calculates the total activation, x_j, by subtracting the internal threshold value T_j:

$$\text{Total Activation} = x_j = \sum_{i=1}^{n} (W_{ij} a_i) - T_j, \tag{2.1}$$

where n is the number of input variables.

If T_j is large and positive, the node has a high internal threshold, which inhibits node output. Conversely, if T_j is zero (or negative, in some cases), the node has a low internal threshold, which excites the node. Some, but not necessarily all, nodes have an internal threshold.

The final factor governing a node's output is the transfer function. Once the node calculates the dot product of vector $\boldsymbol{W_j}$ with vector $\mathbf{a}$, and subtracts the threshold T_j (as described above), it passes this result to a transfer function, $f(x_j)$. Thus, output b_j from the j^{th} node is:

$$b_j = f(x_j) = f(\boldsymbol{W_j} * \mathbf{a} - T_j) = f\left(\sum_{i=1}^{n} (W_{ij} a_i) - T_j\right) \tag{2.2}$$

A particular transfer function is chosen to satisfy some specification of the problem that the ANN is attempting to solve. It may be a linear or nonlinear function; however, mathematicians and scientists have found sigmoid (S-shaped) functions particularly useful. A typical sigmoid function is shown in Figure 2.2. Here $y = f(x)$.

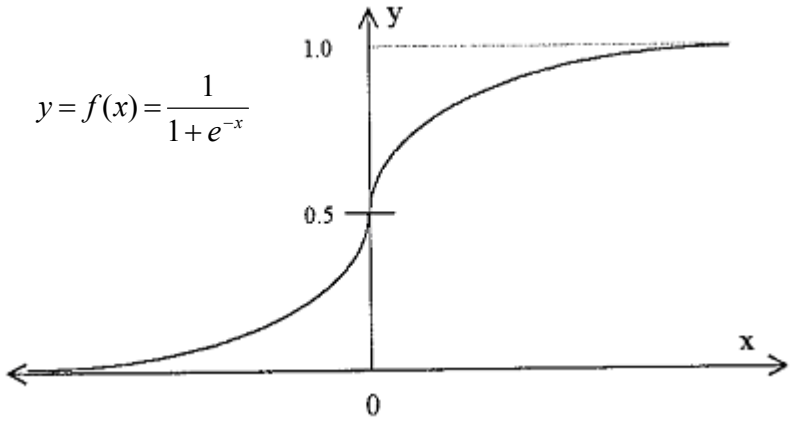

Figure 2.2. A sigmoid transfer function

This function is monotonically increasing, with limiting values of 0 (at $x_j = -\infty$) and 1 (at $x_j = \infty$). All sigmoid functions have upper and lower limiting values. Because of these limiting values, sigmoid functions are also called threshold functions. For the function shown in Figure 2.2 the threshold-function output is zero at very low input values. At very high input values, the output value is one.

Another useful transfer function is the hyperbolic, with limiting values of -1 and + 1. A typical hyperbolic transfer function is shown in Figure 2.3.

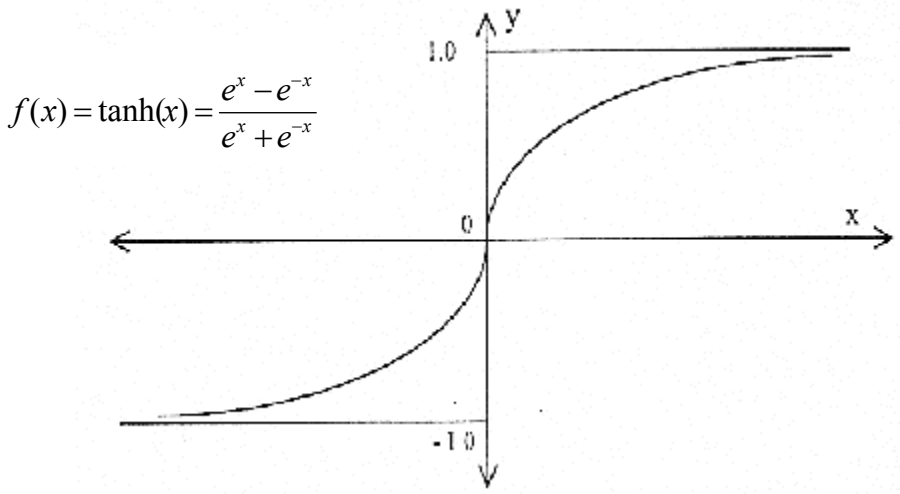

Figure 2.3. A hyperbolic transfer function.

As the biological and chemical processing systems become more complex and nonlinear, the advantages of the hyperbolic transfer function become more apparent. The hyperbolic transfer function outperforms the sigmoid transfer function in many cases. Two features distinguish the hyperbolic transfer function:

a. The slope of the hyperbolic transfer function is much greater than the slope of the sigmoid function, which, means that it shows a greater response to a small deviation in the input variable. Therefore, it can better distinguish between small variations in the input variable and can generate a much more nonlinear response.

b. The hyperbolic transfer function has a negative response for a negative input value and a positive response for a positive input value, while the sigmoid function always has a positive response.

2.3 Topology of ANN

The topology or architecture of ANN refers to how its nodes are interconnected. Although there are several ANN configurations possible, feed-forward ANN is widely used for chemical engineering applications. Feed-forward ANN always consists of at least two hierarchical layers of nodes: a hidden layer, and an output layer. A typical two-layered feed-forward ANN is shown in Figure 2.4.

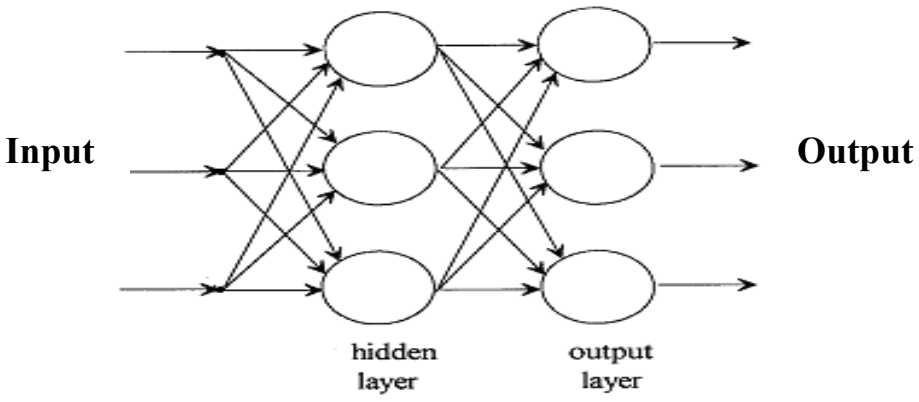

Figure 2.4. A typical two-layered feed-forward ANN

All the nodes in a layer are connected to all the nodes of the adjacent layers, and there are no connections among the nodes in the same layer. The network is constructed in such a way that each layer is fully connected to the next layer. In other words, every node in the hidden layer will send its output to every node in the output layer. The number of nodes in the hidden layers can be varied based on the complexity of the problem and the size of the input information. However, the number of nodes in the output layer is set by the number of output variables.

Multilayer ANN are more powerful than single-layer ANN. It has been shown that any continues real-valued function can be approximated by a two-layered ANN to any arbitrary degree of accuracy, given a sufficient number of nodes in the hidden layer [15, 16].

2.4 Training ANN

Generally, when we first build an ANN, we pre-specify the topology, that is, we specify the interconnections, but leave the numerical values of the weights up to the training phase. Learning or training is the process where the ANN approximates the function mapping from system inputs to outputs, given a set of observations of its inputs and the corresponding outputs. The phrase “training” will be used in lieu of others from now on. Training implies that the node somehow changes its input/output relationship in response to the environment via changes in the values of their weights.

There are many different approaches to train ANN, most fall in one of two groups: supervised training and unsupervised training. The primary training method and the one we use in this work is called backpropagation (BP), which is one of the most important methods for the supervised training of multi-layer feed-forward ANN, when dealing with function approximation problems.

2.4.1 Backpropagation (BP)

BP has been applied to a wide variety of practical problems and it has proven very successful in its ability to model nonlinear relationships. BP derives its name from the fact that error signals are propagated backward through the ANN on a layer-by-layer basis. This is done by adjusting the connecting weight of ANN, in such a way as to minimize the sum of squared errors (E_D) between desired and calculated outputs. For each training set of ANN with n output variables the E_D is defined below [8]:

$$E_D = \sum_{i=1}^{n} (t_i - b_i)^2 , \tag{2.5}$$

where t_i is the desired target output, b_i is the output calculated via ANN. The weights for each connection are initially randomized. When the ANN undergoes training, the errors, $t_i - b_i$, i = 1,2,…,n, are propagated backward through the net, as the connection weights are updated during each iteration. Repeated iterations result in a converged set of connection weights, yielding an ANN that exhibit the relationships between sets of input data and the corresponding sets of target values used in training.

2.5 Black-box ANN (BANN)

ANN in its original form as described so far has typically been used as of the black-box type, that is, no prior knowledge about the process was assumed; the goal was to develop a process model based only on observations of its input-output behavior. With availability of enough experimental data about the process, engineers usually can develop such a "black-box ANN" (BANN) model without too much difficulties. There are a lot of examples in the open literatures for the application of this approach in chemical and biochemical process. For example, Baughman and Liu [3] applied this approach for several chemical processes. Also, Kasprow [17] applied BANN for continuous and batch biochemical processes. Modeling with BANN quite often is the only possible method when no process knowledge is available [19]. However, being essentially black box models, they may be of poor ability for extrapolation and are difficult for interpretation and analysis of the behaviors of the process.

2.6 Hybrid ANN (HANN)

In recent years, there has been an increasing interest in developing modeling methods that address the problems associated with BANN. Recently, research effort has focused on the use of "hybrid ANN" (HANN) models that combine both the deterministic and the BANN elements [17,19]. For example Psichogios and Ungar [19] considered the case of a fed-batch bioreactor, using cell mass and substrate balances as the deterministic section of their HANN model. According to Kasprow [17], there have been three types of HANN methods for combining neural networks with process models. Briefly, the first method

uses ANN to predict the rate of change of one or more state variables; these rates are then used in a mass-balance expression. The second uses ANN to determine additive corrections to an assumed simple model. The third uses ANN to predict constant model parameters. The following three sub-sections, mainly taken from Kasprow [17], discuss these three types in more details.

a) HANN Involving Rate Prediction

In this type of HANN, the aspects of the problem whose quantitative behavior is well understood are described by deterministic mathematical equations, while ANN describe the unknown kinetics. Several research groups applied this HANN procedure to biochemical processes [1,3,19,23]. Also, it was applied to non-biological system such as a continuous stirred tank reactor (CSTR) [7], batch biochemical reactors [7], and a fluidized bed reactor [24].

b) HANN involving Additive Corrections

Another HANN modeling seen in the literature uses neural networks to provide an additive correction to simple process models. In this approach the ANN represents the complexity of the true system that cannot be accounted for in the simple assumed model. The basic idea behind this technique is that the ANN will model the process nonlinearities, thus enabling the complete hybrid model to capture more complex dynamic. For example Thompson and Kramer [22] used this approach for modeling a simulated fed-batch penicillin process.

c) HANN Involving Parameter Prediction

The third HANN approach uses ANN to provide values for constant parameters in a first-principle model. Therefore, in this approach, the partial first-principles model specifies process variable interactions from physical considerations; the ANN complements this model by estimating unmeasured process parameters in such a way as to satisfy the first principles constraints. Nonparametric estimation is needed since no knowledge is available about these parameters. Such models are expected to perform better than BANN models in process identification tasks, since generalization and extrapolation are confined only to the uncertain parts of the process while the basic model is always consistent with first principles and does not allow a physical variable interactions [22]. This approach was used to model a wall-cooled fixed-bed reactor [21], converting benzene to maleic anhydride, using a neural network to predict the overall heat transfer coefficient based on the benzene flow rate, coolant temperature, and air flow rate. Also, Kasprow [17] applied this approach for modeling biochemical process.

26.1 Kasprow's Hybrid Neural Networks/Parameters Model

This approach was developed by Kasprow [17] as an improvement to the hybrid model developed by Tholudur and Ramirez [23]. Tholudur and Ramirez formed a hybrid model in which a simple mass balance is combined with neural networks that predict the protein expression rate, protein secretion rate, growth rate, and yield of cells on substrate. In their approach, the neural network training data is found by solving the mass balance equations for the rates and the yield coefficient. This approach, which was called "hybrid neural

networks/specific rate model" by Kasprow, neglects any prior knowledge about the expected relationship between the state variables and growth rate.

In Kasprow's approach, the mass balance equations are used to form the underling model structure. Then, knowledge of a rough relationship between the state variables and the rates is included in the form of a simple rate model. Rather than correcting the rate model predictions, the rate model parameters are modified based on the state variables. This is a more powerful correction, and has a basis in biochemistry since the model parameters have a physical meaning. Finally, the parameters are not constrained to one constant value for an entire fermentation, but allowed to vary as the state variables change.

This new hybrid scheme was shown to perform significantly better than both a black-box neural networks model and hybrid neural networks/parameters models. This HANN/parameters approach was developed and used to model an enzymatic tubular membrane reactor (TMR).

2.7 Tubular Membrane Reactor (TMR)

In this project a continuous-flow tubular membrane reactor (TMR) for enzymatic saccharification of pretreated lignocellulosic biomass to glucose and cellobiose [11] was selected as the "base process" for studying HANN models. This is mainly due to the availability of a reliable deterministic model which can provide "simulated" process data

as needed, as well as its potential industrial importance in the future. A schematic diagram of the configuration of the TMR is shown in Figure 2.5 [11].

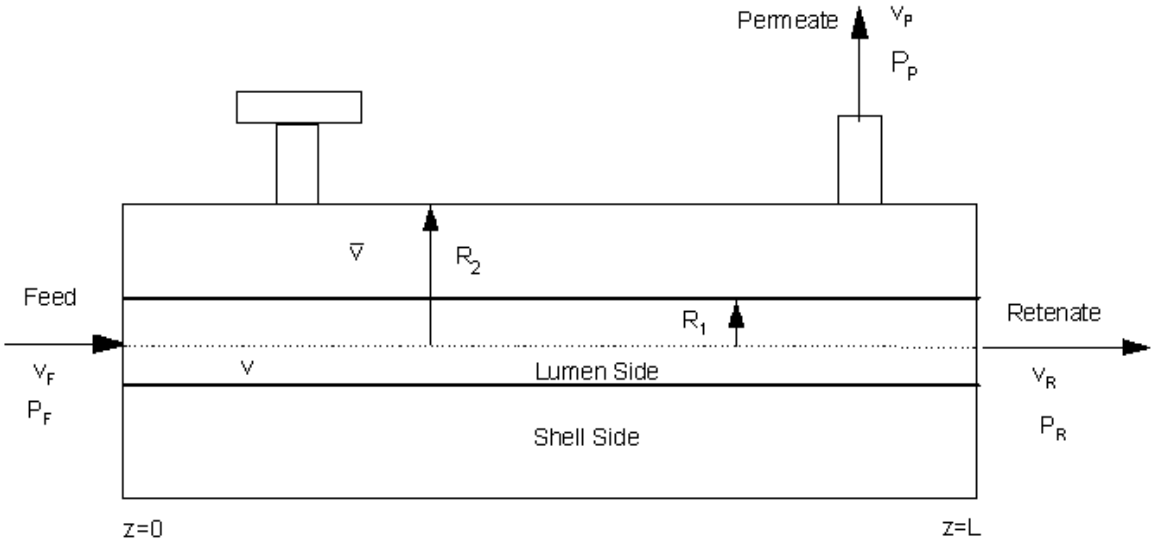

Figure 2.5 Schematic of TMR

The reactor consists of shell and tube sides and tubular membranes made of organic or inorganic membrane. Only one membrane tube with radius, R_1, is shown inside a cylindrical housing of radius, R_2. In reality, this membrane reactor may consist of several to a large number of polysulfone (organic) or ceramic (inorganic) membrane tubes. Thus the TMR may be either a "polysulfon TMR" (PTMR) or a "ceramic TMR" (CTMR). In this work, effort is focused only on PTMR, although the methodologies are equally applicable to CTMR. The TMR has the advantages of: (1) simultaneous reaction and separation in one reactor hence reducing capital cost; (2) enhanced reaction rate throughout the reactor, as a result of removal of inhibitory products; and (3) easy scale up [11].

The associated FORTRAN program of the TMR model, developed here at the Bioreaction Engineering Laboratory at WVU [11], can be used with confidence to generate "simulated" process data for most occasions for training ANN. The use of computer-generated data (from deterministic models) superimposed with purposely added random "noise" for use as "simulated experimental data" for ANN research is a widely used approach among ANN researchers [17].

2.7.1 Enzymatic Saccharification of Cellulose

The conversion of biomass to liquid fuels, such as ethanol, has been of much interest during the 20th century. Ethanol can be produced either by hydrolysis of cellulose to glucose and then fermentation of glucose to produce ethanol, or alternatively via simultaneous saccharification and fermentation (SSF). Although, large-scale production of ethanol from lignocellulosic material in its infancy, commercial production of ethanol from starch has been in existence for many years. The focus of research and development efforts in biomass conversion has currently switched to that of a bio-refinery concept, i.e., to the production of industrially important chemicals, in addition to alcohol.

Hydrolysis is a chemical decomposition process that uses water to split chemical bonds of substances. There are two types of hydrolysis, acidic and enzymatic with the later being the most promising approach [11]. Feedstocks that may be appropriate for enzymatic hydrolysis typically are plant-based materials containing cellulose. These include forest wastes and sawmill residues, agricultural residues, urban wastes, and waste papers [18].

All plants have structural components composed of lignocelluloses fibers, which in turn are comprised of three major fractions: cellulose, hemicelluloses, and lignin. Cellulose consists of a vary long chain of glucose and can be broken down chemically or biologically into glucose and cellulose. The sugars can then be fermented using yeast or bacteria to produce a large number of chemicals.

The use of enzymes for cellulose saccharification has several advantages over acid such as the production of fewer by-products and higher yield of desirable products, hence, less purification is required [11]. The feasibility of enzymatic processes are limited by the cost of enzyme cellulose, so enzyme use must be optimized. One possible approach is the use of membrane bioreactors. Membrane allows continuous removal of inhibitory products (glucose and cellobiose), thus increases conversion to sugars, and also makes more efficient use of the enzymes [18]. The enzymatic saccharification of cellulose is a complex process requiring the participation of cellulase, an enzyme complex. A kinetic model for enzymatic saccharification of cellulose is essential not only for a better understanding of its mechanism, but also for scaling-up of the enzymatic reactors involved.

2.7.2 Kinetics of Saccharification of Cellulose

The kinetics of enzymatic hydrolysis of cellulose has been extensively studied and several kinetic models have been proposed [11]. Cellobiose and glucose are the major products, formed during the enzymatic hydrolysis of cellulose. Most of the models assume that the production of sugars by enzymatic saccharification is a two-step process (two reactions in series) involving the conversion of intermediate cellobiose to glucose:

$$\textit{Cellulose}(\mathrm{S}) \longrightarrow \textit{Cellobiose}\ (\mathrm{B}) \longrightarrow \textit{Glucose}\ (\mathrm{G}) \qquad (2.8)$$

In this work, a two-step competitive product inhibition model is adopted. The competitive product inhibition rate expressions are [11].

$$r_B = \frac{r_m C_S}{K_m + C_S + \frac{K_m}{K_i} C_B} \qquad (2.9)$$

$$r_G = \frac{r_m^{'} C_B}{K_m^{'} + C_B + \frac{K_m^{'}}{K_i^{'}} C_G} \qquad (2.10)$$

The kinetic parameters of this model are listed in Table 3.2 and the symbols used are listed in Nomenclature.

3. Modeling of TMR via ANN

3.1 Deterministic Model of the TMR and Its Numerical Solutions

A deterministic model for enzymatic saccharification of pretreated lignocellulosic biomass to glucose and cellobiose using TMR was developed in our Bioreaction Engineering Laboratory [11]. The essential assumptions of the TMR model are as follows: (1) steady state operation of the reactor; (2) plug flows with negligible axial and radial dispersions in both lumen and shell sides; (3) isothermal operation; (4) negligible concentration polarization; (5) enzymes are completely retained by the membranes; and (6) cellulase deactivation during the period of reactor operation is negligible. In addition, the two-step reaction scheme (Eq. 2.8) has been adopted for hydrolysis of cellulose to cellobiose and glucose. The initial-value type ordinary differential equations (ODEs) used to model this reactor, together with their initial conditions [11], are listed in Table 3.1. A computer programs in FORTRAN was developed by Gauba [11] to solve this set of ODEs using the stiff ODE solver, LSODE. This computer model is capable of providing steady-state concentration profiles of cellulose, glucose, and cellobiose in both lumen and shell sides under different operation conditions.

Since the modeling development of ANN in this work is done using MATLAB, the FORTRAN code of the TMR model was transformed to MATLAB code using MATLAB (version 6). The MATLAB version of the TMR model, **tmr.m**, is listed in Appendix B1.

Table 3.1 Deterministic model of TMR

Tube Side (Lumen)

$$\frac{dC_S}{dz} = \frac{\pi R_1^2}{v}[-r_B - \frac{C_S}{\pi R_1^2}\frac{dv}{dz}]; \qquad C_S(0) = C_{So} \quad (1)$$

$$\frac{dC_B}{dz} = \frac{\pi R_1^2}{v}[-r_B - r_G - \frac{C_B}{\pi R_1^2}\frac{dv}{dz} - (\frac{2L_P\Delta P_T}{R_1})C_B; \qquad C_B(0) = 0 \quad (2)$$

$$\frac{dC_G}{dz} = \frac{\pi R_1^2}{v}[r_G - \frac{C_G}{\pi R_1^2}\frac{dv}{dz} - (\frac{2L_P\Delta P_T}{R_1})C_G]; \qquad C_G(0) = 0 \quad (3)$$

Shell Side

$$\frac{d\bar{C}_B}{dz} = \frac{\pi R_1^2}{\bar{v}}[-\frac{\bar{C}_B}{\pi R_1^2}\frac{d\bar{v}}{dz} + (\frac{2L_P\Delta P_T}{R_1})C_B]; \qquad \bar{C}_B(0) = 0 \quad (4)$$

$$\frac{d\bar{C}_G}{dz} = \frac{\pi R_1^2}{\bar{v}}[-\frac{\bar{C}_G}{\pi R_1^2}\frac{d\bar{v}}{dz} + (\frac{2L_P\Delta P_T}{R_1})C_G]; \qquad \bar{C}_G(0) = 0 \quad (5)$$

Where:

$$\frac{dv}{dz} + 2\pi R_1 L_P \Delta P_T = 0; \qquad \Delta P_T = P_F - P_P + (\frac{P_R - P_F}{L})z$$

$$v = -(2\pi R_1 L_P)[(P_F - P_P)z + (\frac{P_R - P_F}{L})\frac{z^2}{2}] + v_F; \qquad v_F = v + \bar{v}$$

$$r_B = \frac{r_m C_S}{K_m + C_S + \frac{K_m}{K_i}C_B}; \qquad r_G = \frac{r_m' C_B}{K_m' + C_B + \frac{K_m'}{K_i'}C_G}$$

Table 3.2 Parameters used in the TMR Model

R_1 : 0.3 cm	L : 200 cm
P_P : 1033.82 g cm^{-2}	P_F : 1100 g cm-2
P_R : 1070 g cm^{-2}	L_P : 2.5 x 10-7 cc g^{-1} min^{-1}
E_o : 0.152 g l^{-1}	r_m : 1.39x10^{-3} g cc^{-1} min^{-1}
$r_m^{'}$: 1.22x10^{-3} g cc^{-1} min^{-1}	K_m : 42.18x10^{-3} g cc^{-1}
$K_m^{'}$: 198.34x10^{-3} g cc^{-1}	K_i : 1.89x10^{-3} g cc^{-1}
$K_i^{'}$: 0.66x10^{-3} g cc^{-1}	

Although the LSODE solver was used to solve the system of ODEs in the FORTRAN cod, the stiffness of this system was not investigated before. However, to be consistent with the FORTRAN code, a MATLAB stiff ODE solver, **ode23s**, was used to solve ODEs in the MATLAB version of the TMR model. In order to verify the correctness of the numerical results obtained, both programs (FORTRAN and MATLAB) were tested using the same initial conditions (C_{so}=0.0025 g/l and v_F = 0.6 ml/min). The outputs from the two programs are listed in Appendices A1 and A2, and it can be seen that they match each other up to six digits.

3.2 Generation of "Process Data" for ANN Development

As mention before, the use of computer-generated data superimposed with "noise" for use as "simulated experimental data" for ANN research is widely used among researchers

[17]. This approach allows quick and easy generation of training data at different conditions. Also, the amount of noise in the training data can be controlled, and ANN predictions can be compared to a true underlying model.

Numerical simulations were conducted using the MATLAB version of the deterministic TMR model to generate data, which after treatments will be used to train the ANN models. The deterministic model was solved for a variety of feed conditions; this resulted in a data set of 26 cases, i.e., 26 cases of data, each corresponding to a different operation condition. The initial conditions of these cases are shown in Table 3.3. Figure 3.1 illustrate the location of the non-zero initial conditions of each case using a two-dimensional plot of C_{so} vs. v_F.

In order that the computes-generated simulation data would more closely represent actual process data, random noise was added to each of the datum. Noise values were determined at each point by sampling from a normal distribution having zero mean and a standard deviation equal to 3.0% of the values of the datum. A MATLAB function, **noise.m**, in Appendix B2 was developed to add random noise to the simulation data. These noisy values were then considered to be the "process data". Comparison between noise-free data and data with 3% noise for case-1 (C_{so}=0.0025 g/l and v_F = 0.6 ml/min) is shown in Figure 3.2.

Table 3.3 Initial Conditions for Training, Interpolation and Extrapolation Cases

Cases	v_F ml min^{-1}	C_{So} g l^{-1}	C_{Bo} g l^{-1}	C_{Go} g l^{-1}	$\bar{C}_{Bo}$ g l^{-1}	$\bar{C}_{Go}$ g l^{-1}
Training cases						
1	0.6	0.0025	0	0	0	0
2	0.62	0.0016	0	0	0	0
3	0.66	0.0013	0	0	0	0
4	0.67	0.0018	0	0	0	0
5	0.70	0.0009	0	0	0	0
6	0.71	0.0014	0	0	0	0
7	0.75	0.0017	0	0	0	0
8	0.79	0.0005	0	0	0	0
9	0.82	0.0019	0	0	0	0
10	0.83	0.0024	0	0	0	0
11	0.86	0.0007	0	0	0	0
12	0.89	0.0009	0	0	0	0
13	0.91	0.0026	0	0	0	0
14	0.93	0.0011	0	0	0	0
15	0.95	0.0010	0	0	0	0
16	0.97	0.0006	0	0	0	0
17	0.99	0.0021	0	0	0	0
18	1.01	0.0004	0	0	0	0
19	1.03	0.0021	0	0	0	0
20	1.05	0.0018	0	0	0	0
Interpolation Cases						
21	0.6	0.0010	0	0	0	0
22	0.73	0.0015	0	0	0	0
23	1	0.0005	0	0	0	0
Extrapolation Cases						
24	0.65	0.0045	0	0	0	0
25	1.2	0.0003	0	0	0	0
26	1.3	0.0035	0	0	0	0

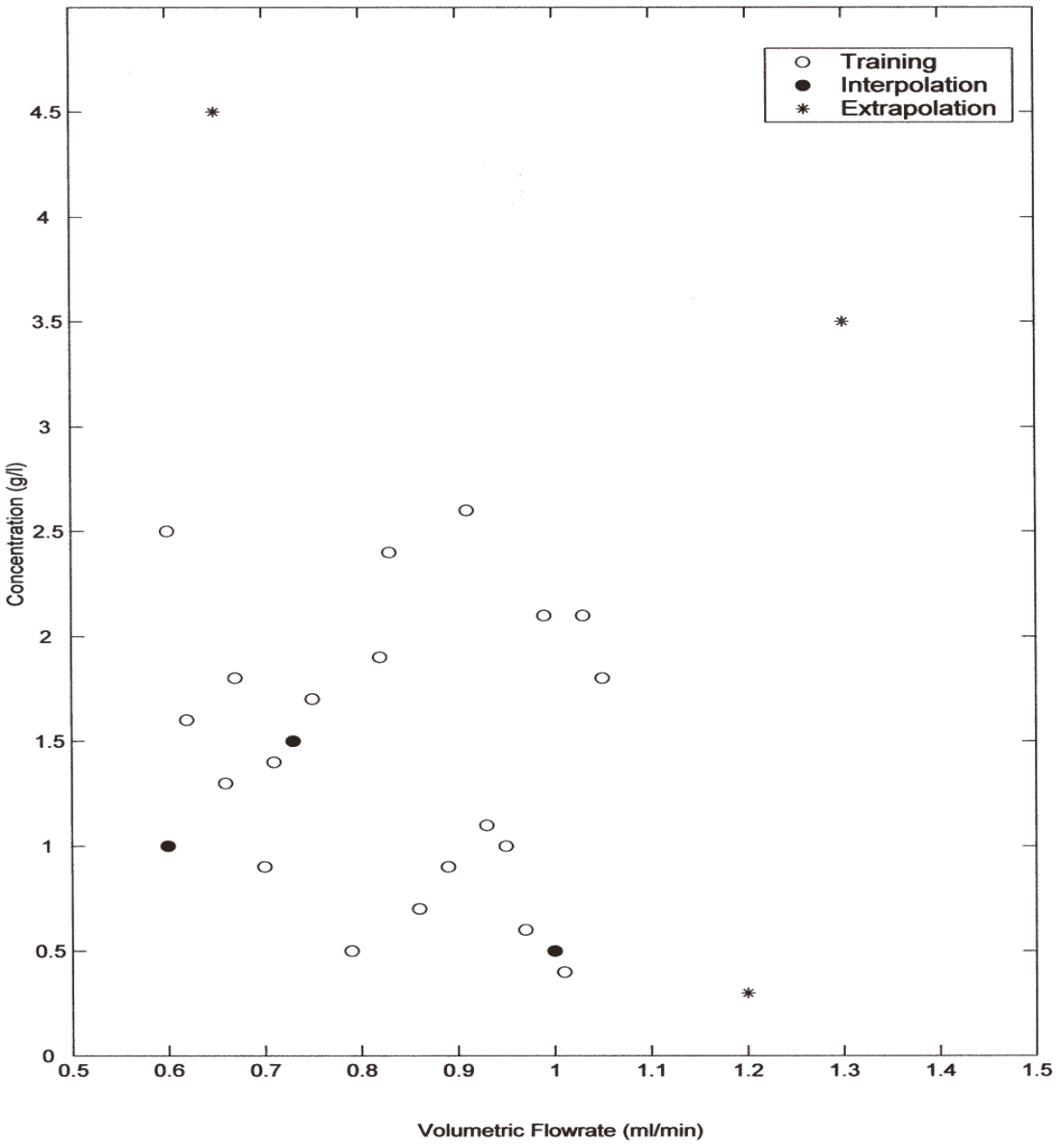

Figure 3.1 Cases with different feed conditions to TMR

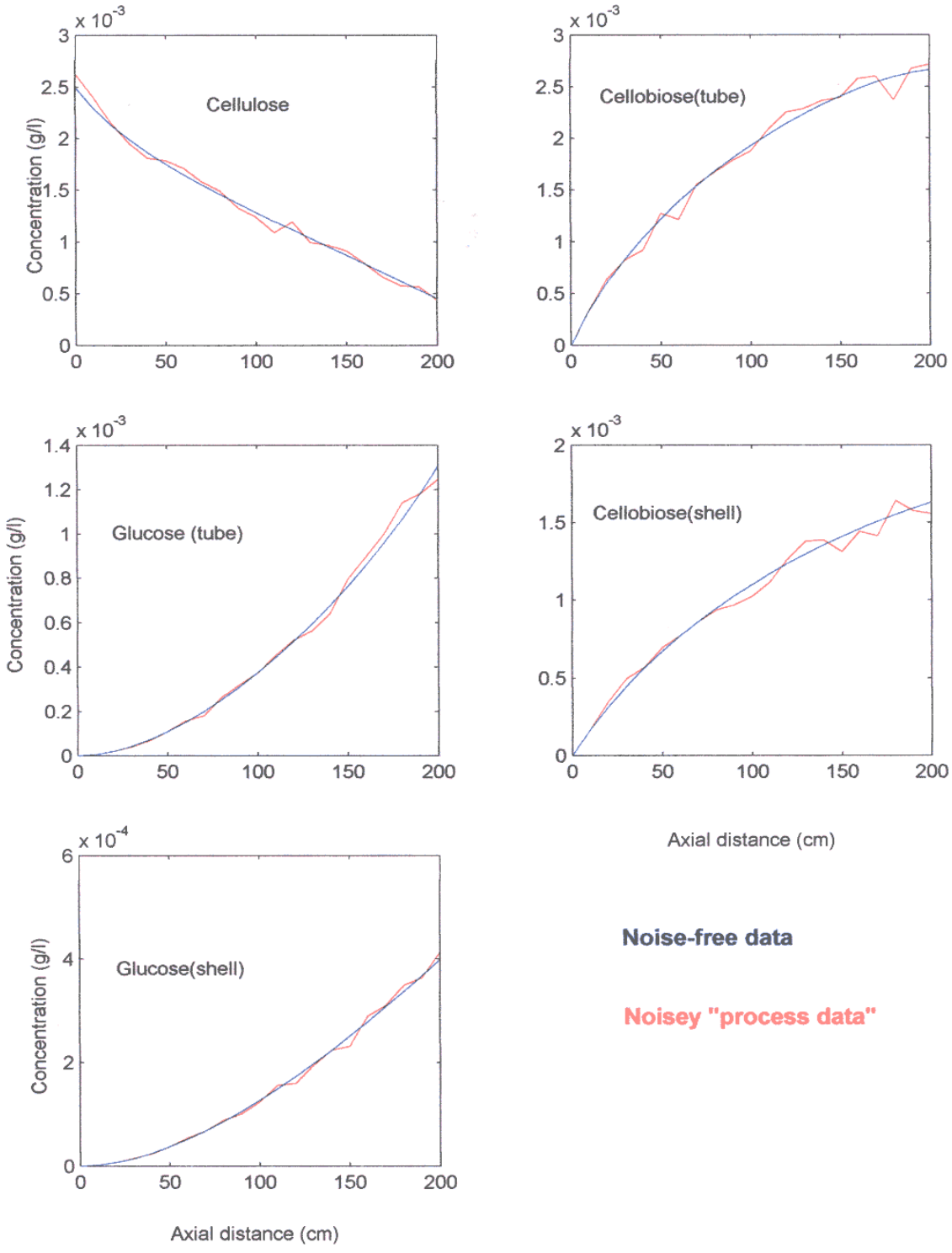

Figure 3.2 Comparison of noise-free and noisy "process data"(case1)

In this work, developing hybrid ANN model (HANN) requires determining the rate of reactions (r_B and r_G) based on the process data. First derivative of the state variable with respect to z_i gives its rate of change. The simplest way to estimate the derivative is to divide the change in the state variable by the change in position Δz between two subsequent points, $\Delta C_i / \Delta z$. However, this type of numerical differentiation will increase the effects of noise, resulting in derivative estimates of lower reliability than the data they are based on. The solution for this problem is to smooth the process data, and to use the smoothed process data to train ANN and estimate derivatives. Therefore, smoothing spline was used to treat the process data sets. The smoothing spline was implemented using a built-in MATLAB function **spaps.m** [4]. Using case1 (C_{so}=0.0025 g/l and v_F = 0.6 ml/min) as an example, Figures 3.3-3.7 illustrate the generation of smoothed process data and first derivative by applying the MATLAB function **spaps.m**.

The MATLAB scripts files **bann_data.m**, **hann1_data.m**, **hann2a_data.m,** and **hann2b_data.m** presented in Appendix B3, B4, B6, and B7 show the creation of training, interpolation and extrapolation data sets for ANN development. Twenty six sets of "smoothed process data" were generated, the first 20 cases have been selected for training, the next three cases (21-23) for interpolations, and the last three cases (24-26) for extrapolations. They are shown in Table 3.3 on by initial conditions. These 26 sets of smoothed process data are the "foundation" of ANN models for TMR to be described next.

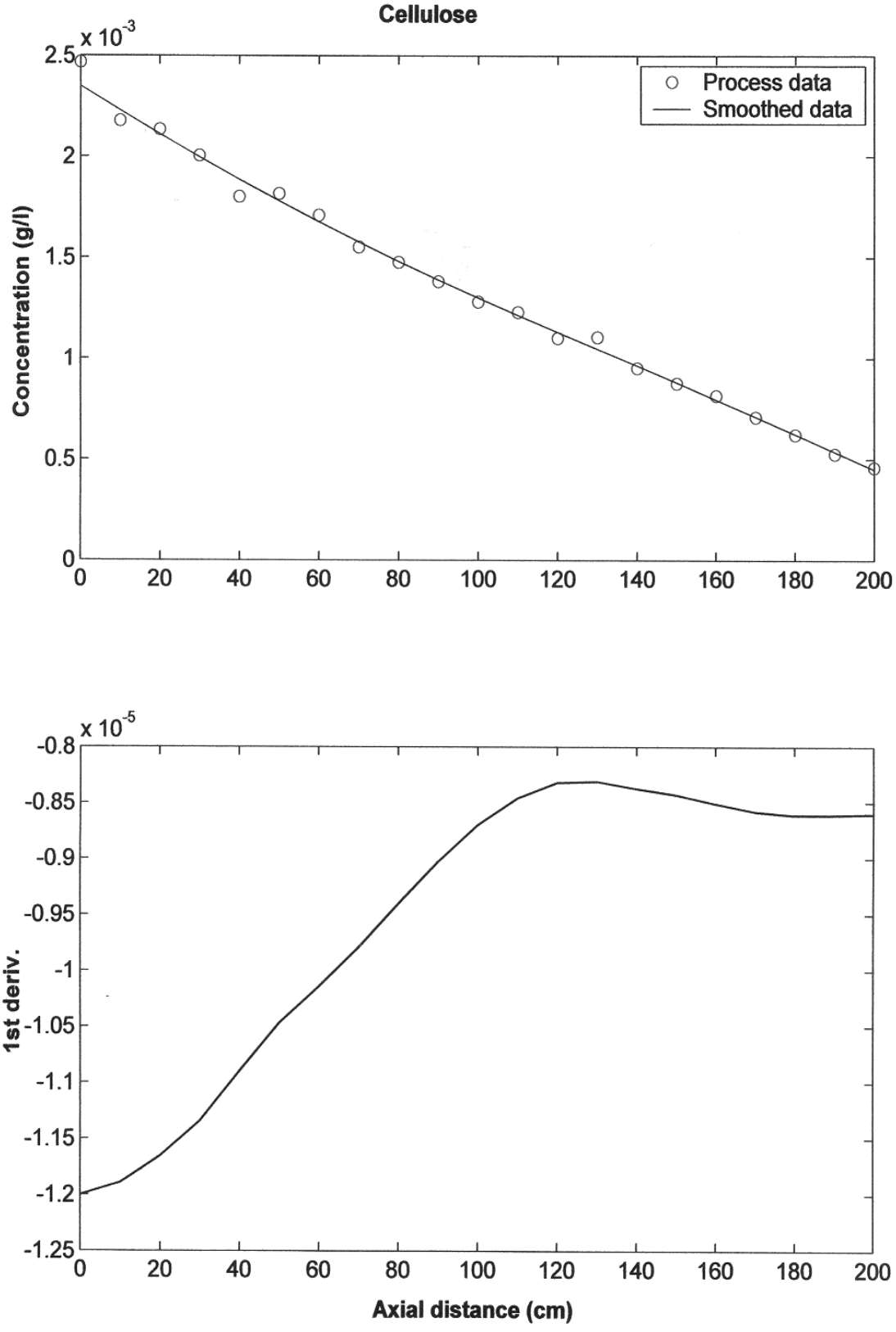

Figure 3.3 Smoothed process data and first derivative of cellulose (case 1)

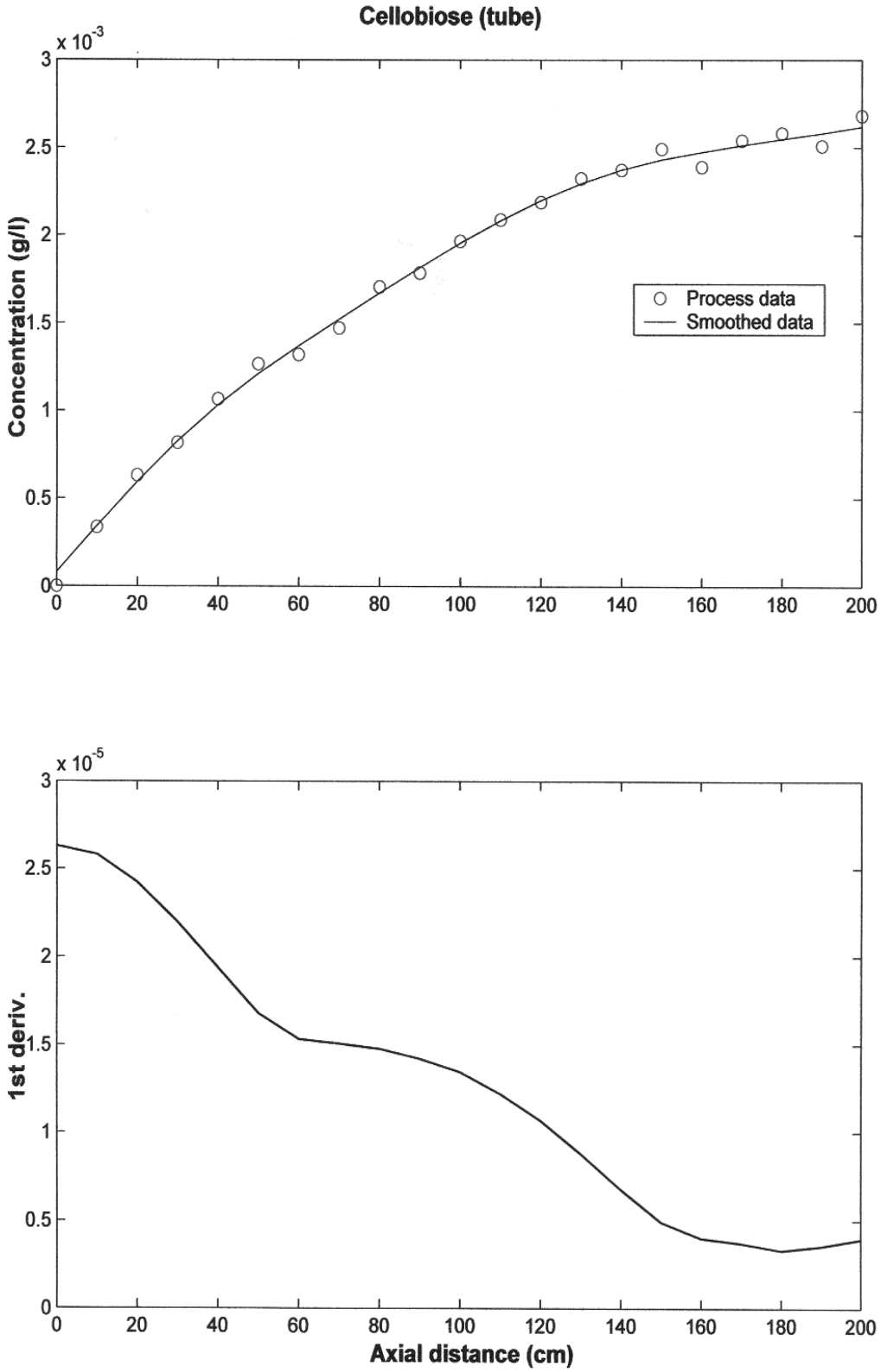

Figure 3.4 Smoothed process data and first derivative of cellobiose in tube side (case 1).

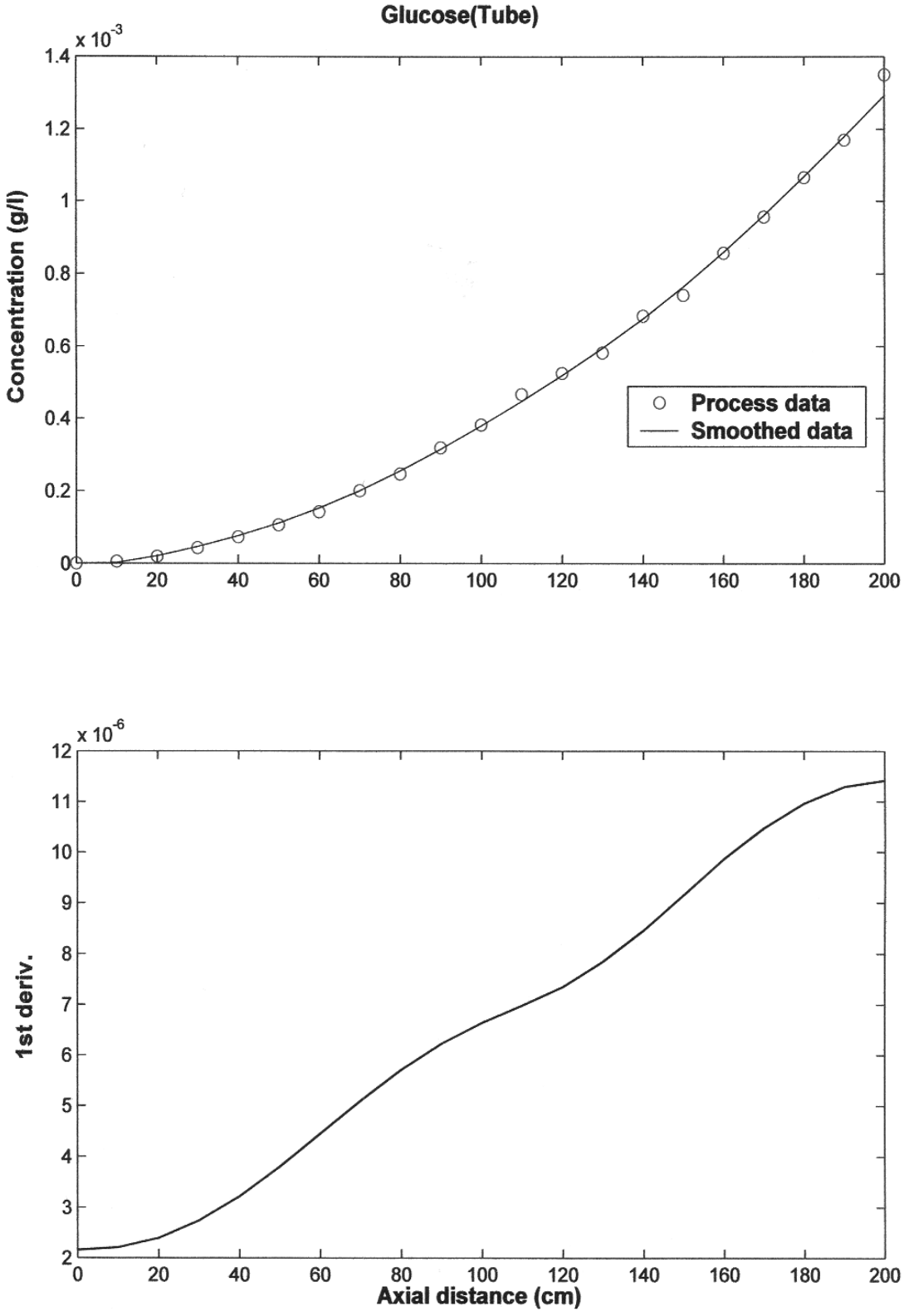

Figure 3.5 Smoothed process data and first derivative of glucose in tube side (case 1).

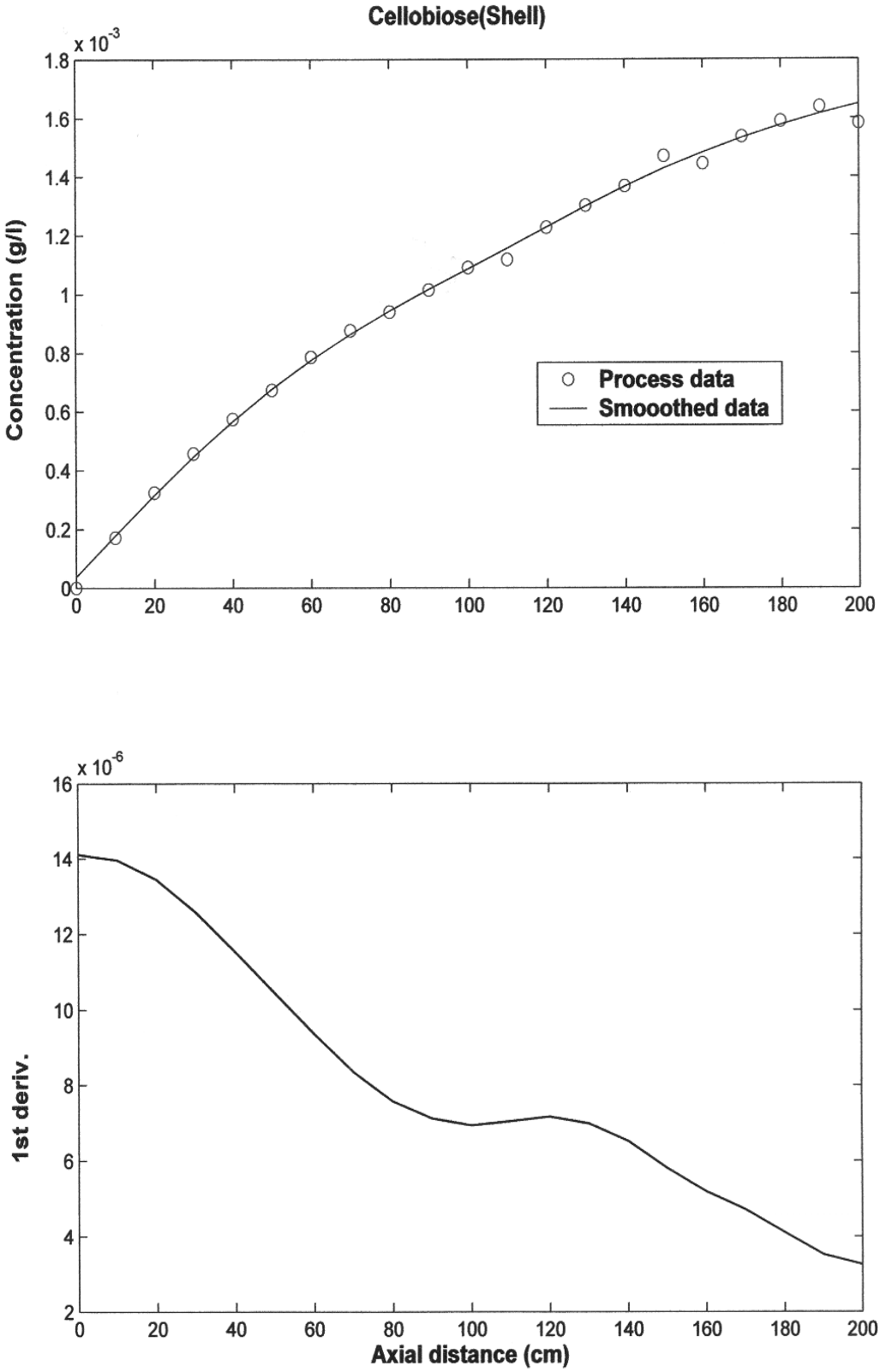

Figure 3.6 Smoothed process data and first derivative of cellobiose in shell side (case 1).

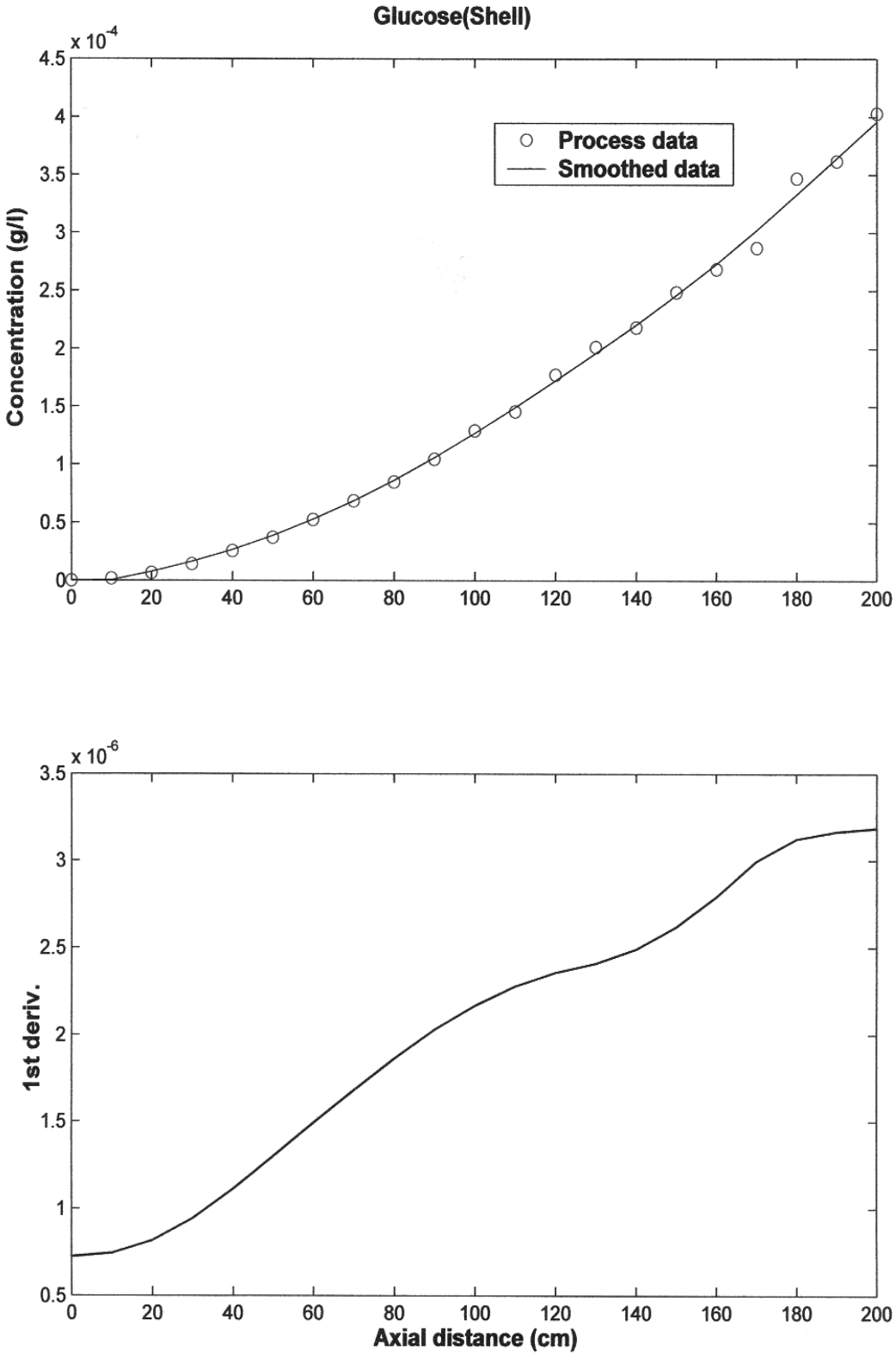

Figure 3.7 Smoothed process data and first derivative of glucose in shell side (case 1).

3.3 Development of ANN models for TMR

BANN and HANN models require the development of ANN to represent the relationships between input and output variables. These ANN were developed and trained using MATLAB Neural Network Toolbox, ver. 4.0 [8]. Based on the process data generated in Section 3.2 the following steps were followed to develop the ANN part of both modeling approaches.

Step 1: Normalization

In order to train the ANN properly, it is necessary to normalize the input and target output data, so that they are all approximately of the same order of magnitude. This is done to make sure that the errors in each of the output nodes are roughly comparable. Otherwise the errors from variables having large magnitude will be weighted too strongly in the training via backpropagation (BP). All the process data were normalized to be in the range of [-1,1]. This was done by using a built-in MATLAB function, **premnmx.m**. This function uses the following equation to perform the normalization:

$$\mathbf{p}_n = \frac{2(\mathbf{p} - \min(\mathbf{p}))}{(\max(\mathbf{p}) - \min(\mathbf{p}))} - 1 \qquad (3.1)$$

where $\mathbf{p}$ is a vector of the original process data, $\mathbf{p}_n$ is a vector of the normalized process data, and $\min(\mathbf{p})$ and $\max(\mathbf{p})$ are respectively the minimum and maximum elements in the vector $\mathbf{p}$. If **premnmx.m** is used to normalize the training data, then the ANN will be

trained to produce output in the range [-1,1]. Therefore, a second MATLAB function, **postmnmx.m**, was used to convert these outputs back into the same units that were used for the original data. This function uses the following equation to perform the de-normalization:

$$\mathbf{p} = 0.5(\mathbf{p}_n + 1)(\max(\mathbf{p}) - \min(\mathbf{p})) + \min(\mathbf{p}) \qquad (3.2)$$

Step 2: ANN architecture

As mention before, it has been shown that a two-layer ANN is capable of representing any continues real-valued function to any arbitrary degree of accuracy, given a sufficient number of nodes in the hidden layer [15,16]. Therefore, in all modeling approaches in this work, a two-layer (hidden layer and output layer) ANN was used, with the output layer node having a "linear" transfer function, i.e., no transformation performed. For the hidden layer, a hyperbolic transfer function was used, because it is the most efficient one, as described in Section 2.2. The number of nodes in the output layer correspond to the number of output variables for each models. However, the number of nodes in the hidden layer was optimized for each model during the training of ANN in each model in order to lessen the chances of over fitting the training data, and to provide the most robust extrapolation possible.

Step 3: Training

Backpropagation (BP) was used to train all ANN developed in this work. There are many variations of BP in MATLAB Neural Network Toolbox [8]. It is not straightforward to know a priori which training algorithm will be the most efficient for a given problem. It depends on many factors, including the complexity of the problem, the number of data points in the training set, and whether the ANN is being used for pattern recognition or function approximation. However, it has been found that on function approximation problems, for ANN that contains up to a few hundred weights, the Levenberg-Marquardt algorithm will have the fastest convergence. The Levenberg-Marquardt algorithm uses matrix $\mathbf{G}$ as an approximation to the Hessian matrix $\mathbf{H}$ in the following iteration scheme [8]:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \mathbf{G}^{-1}(\boldsymbol{w}_k)\,\mathbf{J}^{T}(\boldsymbol{w}_k)\boldsymbol{e}(\boldsymbol{w}_k) \tag{3.3a}$$

where $$\mathbf{G} = \mathbf{H} + \mu\,\mathbf{I} \tag{3.3b}$$

$\boldsymbol{w}_k$ is a vector of current weight and biases, $\mathbf{J}$ is the jacobian matrix that contains first derivatives of the ANN errors with respect to the weights and biases, $\boldsymbol{e}$ is a vector of ANN errors, $\mathbf{I}$ is identity matrix, and μ is the tunable parameter of the Levenberg-Marquardt algorithm. When the scalar μ is zero, this is just Newton's method, using the approximate to Hessian matrix $\mathbf{G}$. When μ is large, this becomes gradient descent method with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is

increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm [8].

One of the problems that occur during ANN training is over-fitting, i.e., poor generalization. The error on the training set is driven into very small value, but when new data is presented to the ANN the error is large. One method for improving ANN generalization is to use ANN that is just large enough to provide an adequate fit. Unfortunately, it is difficult to know beforehand how large an ANN should be for a specific application. There are a few techniques for avoiding over-fitting implemented in the MATLAB Neural Networks Toolbox. One of them is to use MATLAB function, **trainbr.m ,** which employs Bayesian regularization techniques [8].

Typically, training aims to reduce the sum of squared errors. However, regularization adds an additional term to avoid over-fitting; the objective function becomes [9,10]:

$$F = \beta E_D + \alpha E_W \ , \tag{3.4}$$

where: $E_W = \sum_{i=1}^{n} w_i^2$; $\alpha = \frac{\gamma}{2E_W}$;

$\beta = \frac{n-\gamma}{2E_D}$; $\gamma = N - tr(\mathbf{H}^{-1})2\alpha$;

E_W is the sum of squares of the network weights, α and β are the regulation parameters, N is the total number of parameters (weights and biases) in the ANN, γ is a measure of

how many parameters in the ANN are effectively used in reducing the error function, $tr(\mathbf{H}^{-1})$ is the trace of the inverse of Hessian matrix **H**.

The problem with regularization techniques is that it is difficult to determine the optimum value for α and β. One approach to optimize these regulation parameters automatically is the Bayesian framework of David MacKay [10]. In this framework, the weights and biases of the network are assumed to be random variables with specified distributions. The regularization parameters are related to the unknown variances associated with these distributions. A detailed discussion of the use of Bayesian regularization, in combination with Levenberg-Marquardt training, can be found in Foresee and Hagn [10].

Bayesian regularization has been implemented in the function **trainbr.m**. One feature of this function is that it provides a measure of how many network parameters (weights and biases) are being effectively used by the ANN. **trainbr.m** function was used to train and determine the optimum number of hidden nodes in all the ANN developed in this work.

3.4 Modeling TMR Using BANN

In this approach only the ANN was used to model the TMR system. No information about the process is included in this type of model; the ANN network must extract the relationships between input and output variables from the process data. Therefore, the BANN model is purely "empirical".

The total length of TMR being numerically simulated is set at 200 cm. A schematic diagram of this model is shown in Figure 3.8.

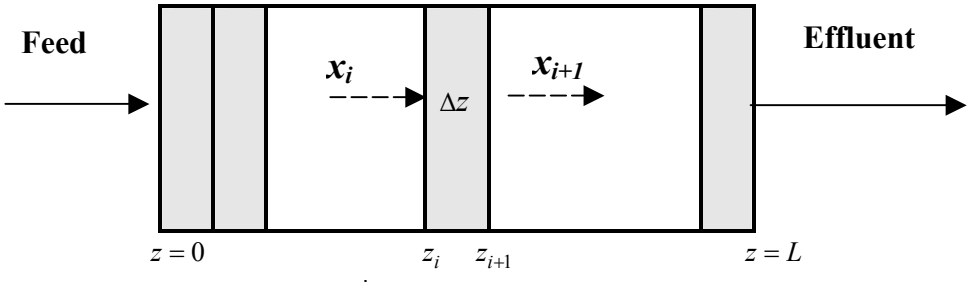

Figure 3.8 A schematic representation of the TMR being numerically simulated.

The deterministic model was set to provide steady-state concentration profiles of cellulose, glucose, and cellobiose in both lumen and shell sides at each 10 cm length. As shown in Figure 3.8, for each segment there is an input vector $\boldsymbol{x}_i$ (at $z = z_i$) and an output vector $\boldsymbol{x}_{i+1}$ (at $z = z_{i+1} = z_i + \Delta z$) of the variables involved and the output vector is served as an input vector to the next segment until the end of the reactor. In BANN model the ANN is used to simulate each 10 cm segment of TMR model. Therefore, each training cases consists of a total of 20 pairs vectors of input and output variables. The ANN of this model has six input variables, the first five are the concentrations of cellulose, cellobiose, and glucose in tube and shell sides of the TMR. The sixth is the inlet feed flow rate, in order to allow the ANN to discriminate between different training cases. The output variables are five and they are the concentrations of cellulose, cellobiose, and glucose in tube and shell sides at the output of each segment. A schematic representation of the BANN model is shown in Figure 3.9. All the 20 training cases were normalized between [-1,1] and prepared as input and output vectors. A total of 400 pairs

of input/output vectors were used to train the ANN. A MATLAB script file (**bann_data.m**) in Appendix B3 was used to generate the training cases for BANN model using the feed conditions of the training cases in Table 3.3.

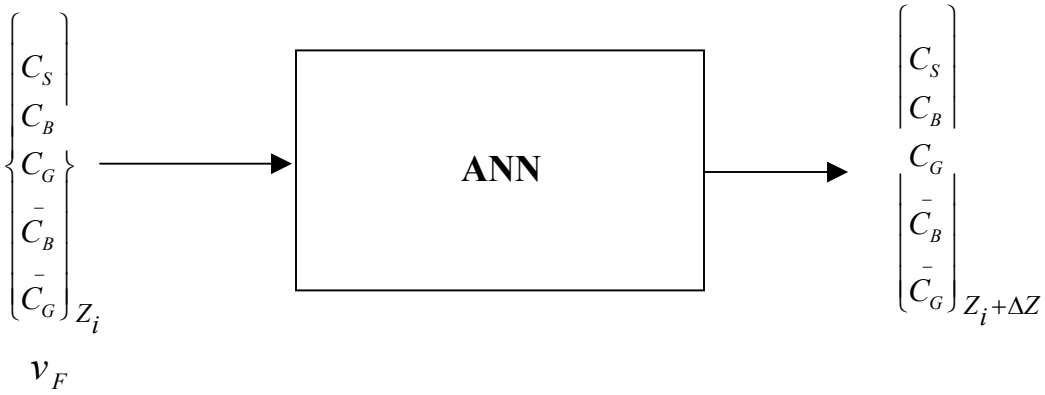

Figure 3.9 BANN for TMR

A single-hidden-layer ANN was used for this model. This ANN has five nodes in the output layer and uses linear transfer functions there. A hyperbolic transfer function was used in the hidden layer. The number of the nodes in the hidden layer was optimized by using **trainbr.m**. All 20 training cases were used to train different configurations, i.e, different ANN with different nodes in the hidden layer. The interpolation cases (21-23) were used for testing. As shown in Table 3.4, the number of nods in the hidden layer was varied between 1 to 20 nodes. Table-3.4 summarized the training results of all ANN obtained by using **trainbr.m**. In additional to E_D and E_W the following parameters are illustrated: S, N, γ and E_T, where S is the number of nodes in the hidden layer, E_T is the sum of squared errors on the test set containing the interpolation cases. Figure 3.10 shows the trend of E_D, E_T, N, and γ vs. S. It can be seen that for all ANN with $S \geq 10$ the effective number of parameters remain constant, even though the actual number of

parameters increase as the size of ANN becomes larger. This indicates that the ANN with 10 nodes in the hidden layer is the smallest ANN with sufficient complexity to fit the data but not to over-fit them. The performance of this ANN was evaluated by using recall, interpolation and extrapolation cases and the results of this evaluation are discussed in Chapter 4.

Table-3.4 ANN Development for BANN Model

S	E_D	E_W	E_T	N	γ
1	31.939	7.6204	4.2098	17	16.2
2	3.9469	13.401	0.5226	29	27.5
3	0.2236	30.580	0.0326	41	38.7
4	0.1673	21.004	0.0194	53	50.5
5	0.1282	51.563	0.0186	65	61.9
6	0.1167	29.909	0.0171	77	71.4
7	0.1167	24.579	0.0184	89	79.2
8	0.1166	19.607	0.0167	101	86.4
9	0.1093	20.368	0.0164	113	94.4
10*	0.1065	25.941	0.0154	125	105
11	0.1064	22.281	0.0155	137	105
12	0.1067	19.920	0.0152	149	105
15	0.1067	19.784	0.0154	185	105
20	0.1067	18.997	0.0158	245	105

*Optimum number of nodes in the hidden layer.

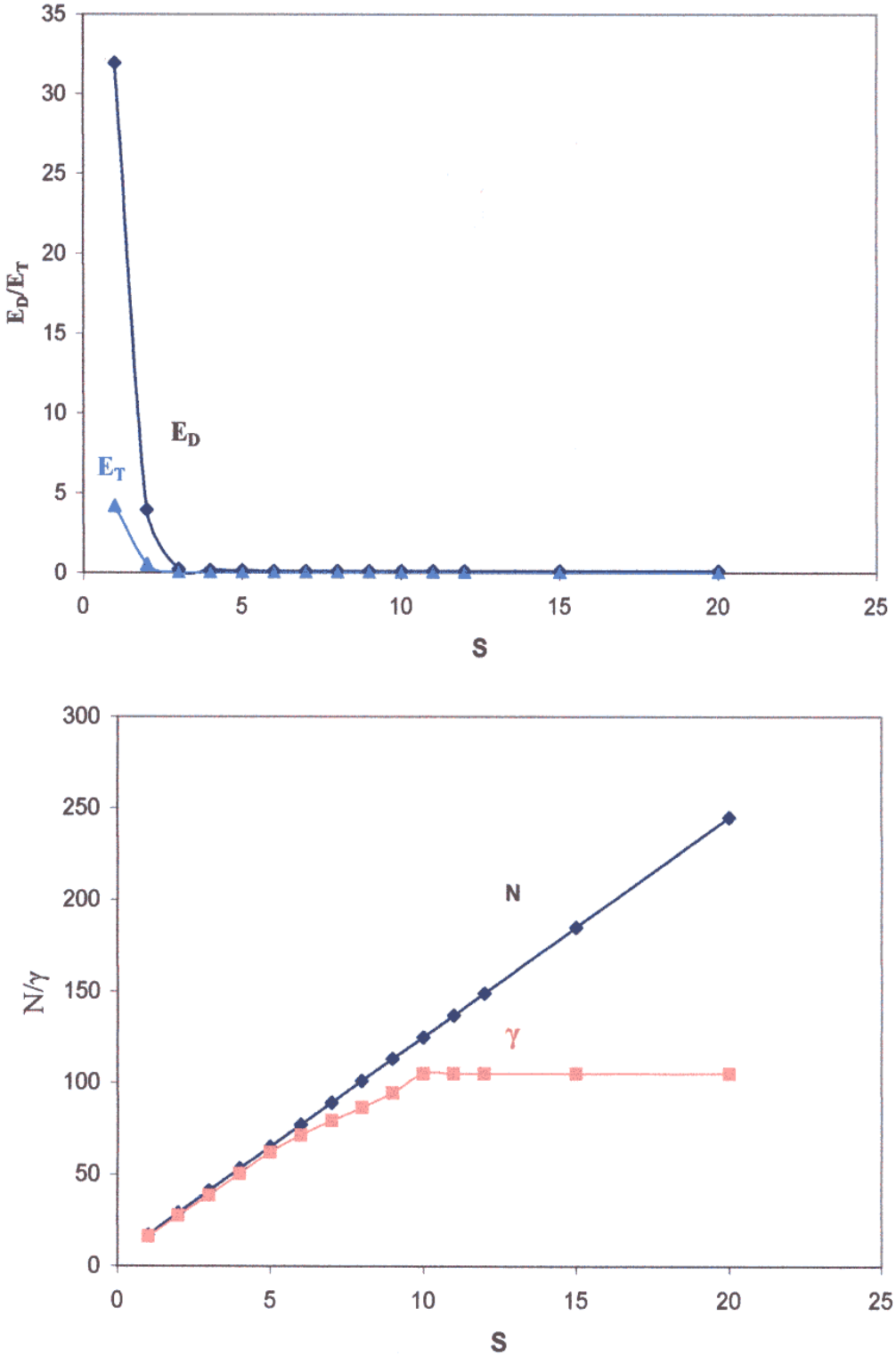

Figure 3.10 ANN development for BANN model

3.5 Modeling TMR Using HANN1

The second modeling approach developed is a hybrid ANN model (HANN1). The deterministic model of the TMR consists of material balance equations (ODEs) and the reaction rates expression involved (Table 3.1). In this approach the expression of the rates of production of cellobiose and glucose, r_B and r_G respectively, are assumed to be unknown and the ANN is used to predict them. In this manner, the ANN becomes nonparametric estimator of the reaction rates. Therefore, the central idea of this modeling approach is the combination of first principles model, in the form of mass balance equations (ODEs) with ANN, which approximates the unknown kinetics, in order to form a combined model structure which can be characterized as a hybrid ANN model (HANN1). A schematic representation of the HANN1 model is shown in Figure 3.11.

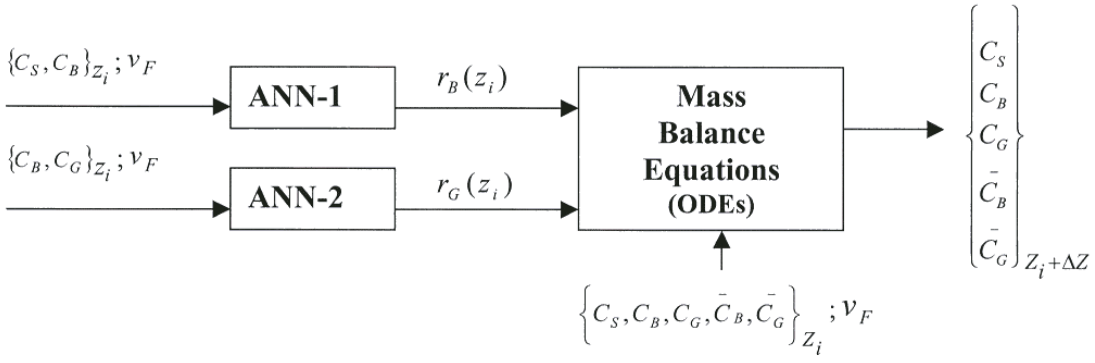

Figure 3.11 HANN1 for TMR

As shown in this figure, two ANN are used to predict r_B and r_G. The first one (ANN-1) is given as input the concentrations of cellulose and cellobiose in tube sides, as well as

the feed flowrate, v_F to predict the rate of formation of cellobiose (r_B). The second one (ANN-2) is given as input the concentrations of cellobiose and glucose in tube sides and the feed flowrate, v_F to predict the rate of production of glucose (r_G).

The training of ANN for BANN model was straightforward, because the BANN model consists only of ANN in its original form and the input/output data are directly available. However, for the HANN1 model the target output for ANN-1 and ANN-2 are not directly available. Therefore, the ODEs equations in Table 3.1 are rearranged for r_B and r_G, as shown in equations 3.5 and 3.6.

$$r_B = -\frac{dC_S}{dz}\frac{v}{\pi R_1^2} + \frac{2L_P \Delta P_T C_S}{R_1}; \tag{3.5}$$

$$r_G = \frac{dC_G}{dz}\frac{v}{\pi R_1^2}; \tag{3.6}$$

where $\Delta P_T = P_F - P_P + (\frac{P_R - P_F}{L})z$

Then, these equations were used to calculate r_B and r_G using the smoothed process data and the values of the first derivative of cellulose and glucose at each segment for all of the training cases. A MATLAB script file (**hann1_data.m**) in Appendix B4 was used to generate the training data for the ANN-1 and ANN-2 using the feed conditions of the training cases in Table 3.3.

Two single-hidden-layer ANN were developed for HANN1 model. Both of them (ANN-1 and ANN-2) consist of one node in the output layer with the use of linear transfer function. Hyperbolic transfer function was used in the hidden layer for both ANN. The number of the nodes in the hidden layer was optimized during the training of both ANN using **trainbr.m** function. Table-3.5 and Table-3.6 summarized the training results of ANN-1 and ANN-2 obtained by using all training cases (1-20) and the interpolation cases for testing. Figures 3.12 and 3.13 illustrate the performance of different ANN strictures developed for ANN-1 ANN-2. As shown in these tables and figures the optimum number of nodes in the hidden layer is 6 for ANN-1, and 5 for ANN-2, because after this point, the effective number of parameters remain constant.

The trained ANN (ANN-1 and ANN-2) with optimum configurations were combined with the mass balance equations (ODEs) as shown in Figure 3.14. A MATLAB program, **hann1.m**, in Appendix B5 was developed to combine these two ANN with mass balance equations. As shown in Figure 3.14, ANN-1 and ANN-2 receive as inputs the normalized concentrations of cellulose, cellobios, and glucose in tube side and the feed flowrate and predicate as outputs the normalized rates of formation of cellobiose and glucose (r_B and r_G). The de-normalized ANN outputs serve as an input to the mass balance equations (ODEs), which produces as output the concentrations of cellulose, cellobiose, and glucose in tube and shell sides. This step was repeated iteratively until all the 20 pairs of input/output vectors for each training case is included. The combination of ANN and mass balance equations yields a complete HANN1 model for TMR. The performance of

the HANN1 model was evaluated by using recall, interpolation and extrapolation cases and the results of this evaluation are discussed in chapter 4.

Table-3.5 ANN-1 Development for HANN1 model

S	E_D	E_W	E_T	N	γ
1	0.4580140	3.6998	0.05113	6	5.74
2	0.0975496	8.5823	0.01627	11	10.2
3	0.0535833	17.1539	0.00825	16	15.1
4	0.0388297	19.3016	0.00479	21	19.1
5	0.0381351	18.8586	0.00437	26	23.3
*6	0.0338672	22.147	0.00356	31	27.1
8	0.0338355	21.5688	0.00380	41	27.8
10	0.0341173	21.2022	0.00400	51	27.6
15	0.0338322	22.6352	0.00338	76	27.8
20	0.0335925	21.1633	0.00393	101	28.5

*Optimum number of nodes in the hidden layer.

Table-3.6 ANN-2 Development for HANN1 model

S	E_D	E_W	E_T	N	γ
1	1.18114	23.600	0.44980	6	5.36
2	0.569646	35.463	0.06747	11	9.11
3	0.408939	39.313	0.05704	16	14.2
4	0.367445	39.588	0.05424	21	18.5
*5	0.353455	39.053	0.04781	26	22.7
6	0.354659	38.271	0.05030	31	22.6
7	0.354662	38.187	0.05030	36	22.6
10	0.354803	38.138	0.05050	51	22.8
15	0.354529	37.990	0.05019	76	22.8
20	0.354287	38.025	0.05080	101	22.8

*Optimum number of nodes in the hidden layer.

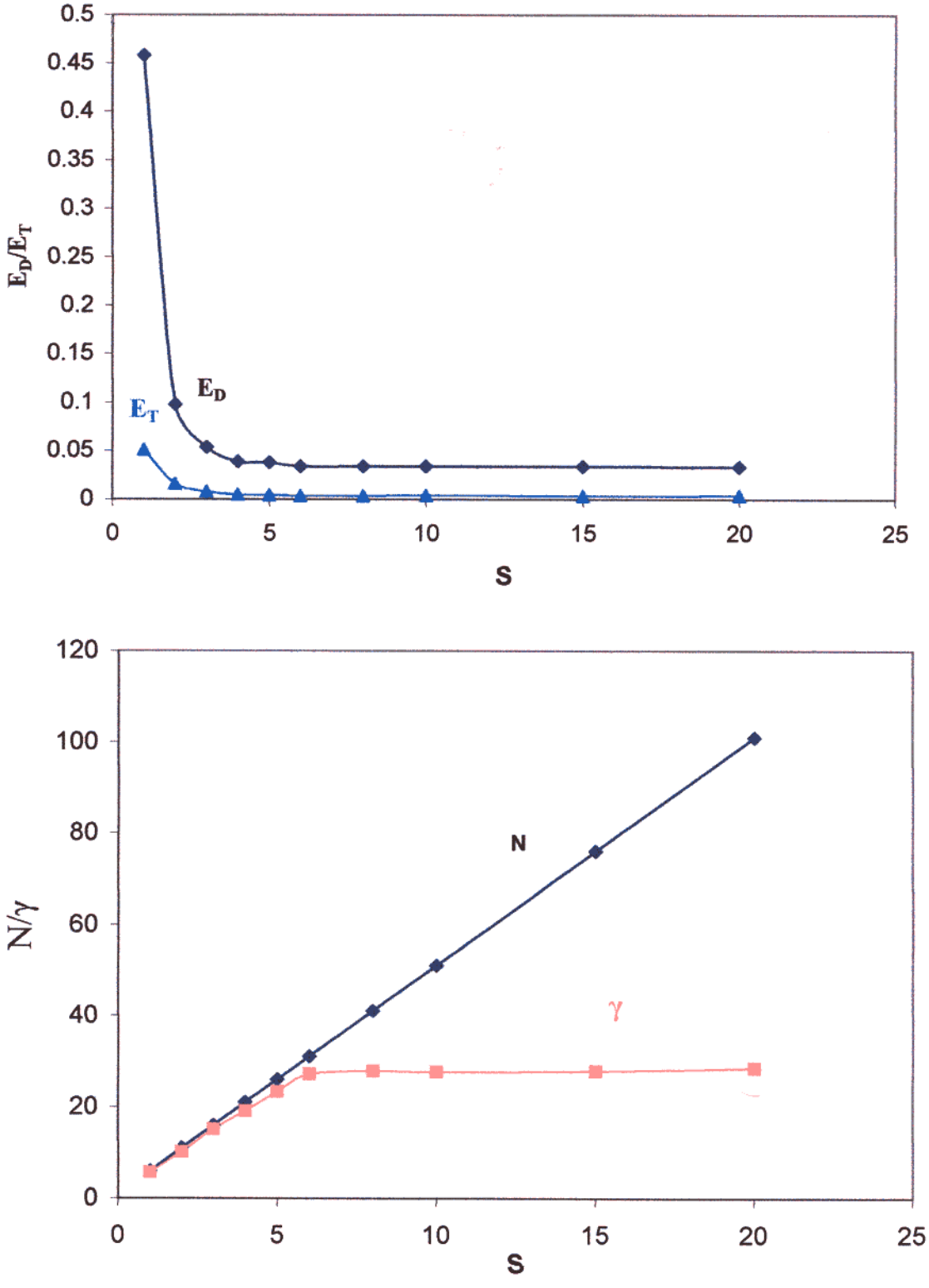

Figure 3.12 ANN-1 development for HANN

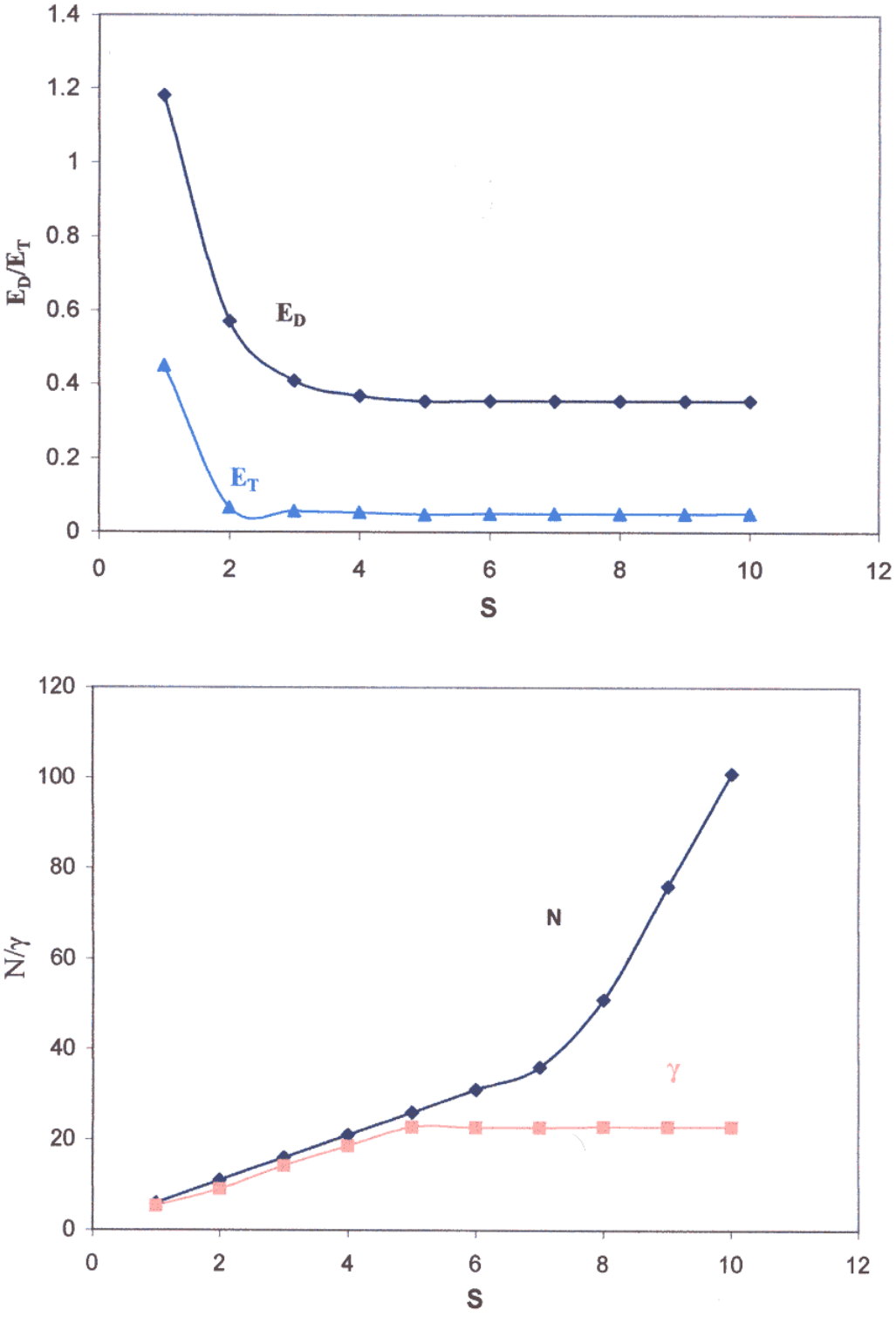

Figure 3.13 ANN-2 development for HANN

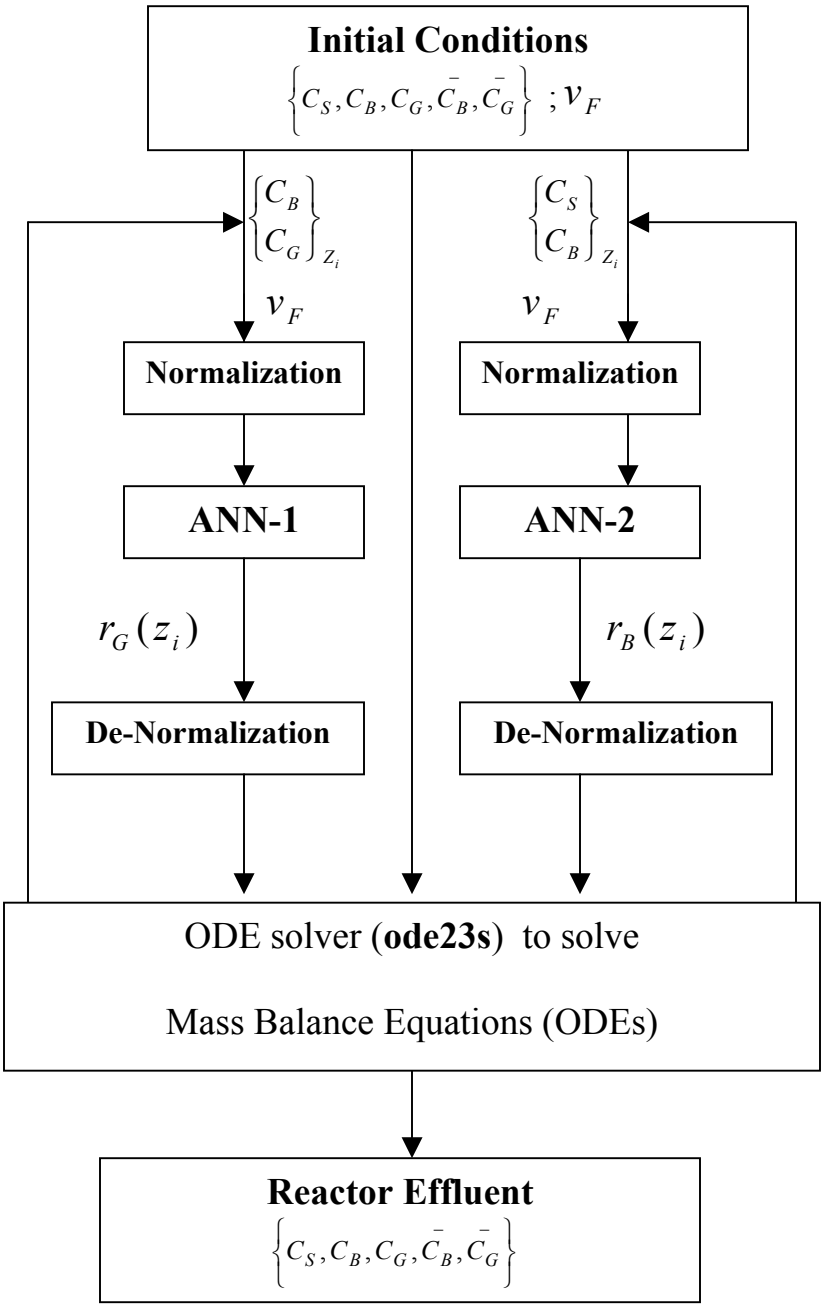

Figure 3.14 Combination of ANN with ODEs for HANN1 model

3.6 Modeling TMR using HANN2

The second hybrid approach (HANN2) in this project is an application of the new hybrid modeling technique developed by Kasprow [17]. In addition to the mass balances equations used in the previous hybrid approach (HANN1), this model also assumes two simple expressions for the reaction rates. However, rather than using constant values for all of kinetics parameters, some parameters will have to vary with the state variables in order for the hybrid model to emulate the true situation where rate expressions are not explicitly known. This acts to relax the constraints on the specific rates, in that they are not restricted to a certain assumed functional form and also are allowed to vary with variables not explicitly included in the assumed function.

In this work the kinetics of enzymatic hydrolysis of cellulose in the deterministic model has been described using a two-step competitive product inhibition rate expressions (equations 2.9 and 2.10). Instead of using theses two rate expressions, two simple rate expressions are assumed as follows:

$$r_B = \frac{r_m C_S}{K_m + C_S} \tag{3.8}$$

$$r_G = \frac{r_m^{'} C_B}{K_m^{'} + C_B} \tag{3.9}$$

In theses two simple rate expressions K_m and $K_m^{'}$ will be determined by ANN based on process data. Therefore, the ANN will be used to represent the variation in the rate model

parameters with $\{C_S,\ C_B,\ C_G,$ and $v_F\}$. A schematic diagram of this hybrid approach is shown in Figure 3.15.

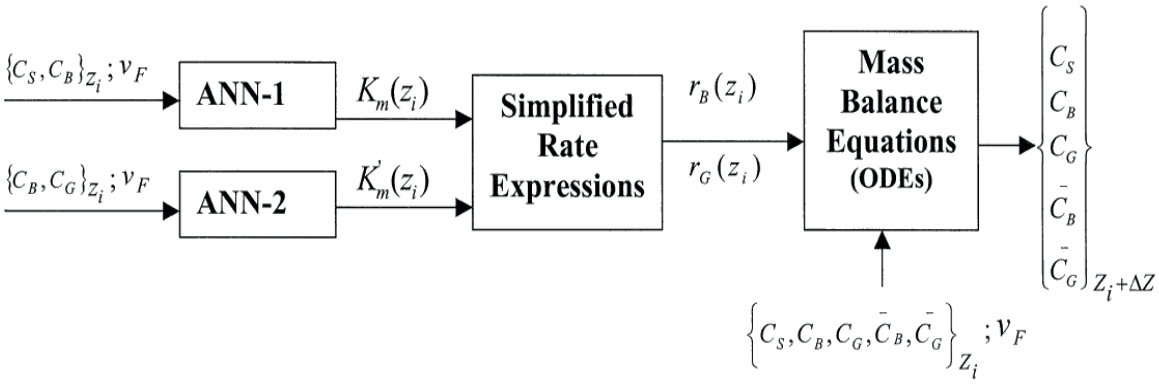

Figure 3.15 HANN2 Model for TMR

As shown in this figure, the two ANN (ANN-1 and ANN-2) will receive the state variables at z_i as inputs. Then, they will predict the values of K_m and $K_m^{'}$ at z_i. These values, along with the pre-chosen constant kinetic parameters, r_m and $r_m^{'}$, will be used to predict reaction rates, r_B and r_G, at z_i using the simplified rate expressions, once K_m and $K_m^{'}$ are determined by ANN. The calculated reaction rates along with state variables at z_i will then be used in the mass balance equations (ODEs) to predict the state variables at $z_i + \Delta z_i$.

Once the ANN variable parameters are chosen (K_m and $K_m^{'}$) the next step is to develop training data sets illustrating how these parameters vary with the state variables. In order to do that, the ODEs equations in Table 3.3 are rearranged for K_m and $K_m^{'}$, as shown in the following equations:

$$K_m = \frac{r_m C_S}{\frac{2L_P \Delta P_T C_S}{\pi R_1} - \frac{dC_S}{dz}\frac{v}{\pi R_1^2}} - C_S \tag{3.10}$$

$$K_m^{'} = \frac{r_m^{'} C_B}{\frac{dC_G}{dz}\frac{v}{\pi R_1^2}} - C_B \tag{3.11}$$

where $\Delta P_T = P_F - P_P + (\frac{P_R - P_F}{L})z$

Two ANN were developed for this model to represent the variation in K_m and $K_m^{'}$ with state variables. The first ANN (ANN-1) have three inputs variables; the first two are concentrations of cellulose and cellobiose in the tube side of the TMR. The second ANN (ANN-2) also have three inputs variables; the first two are the concentrations of cellobiose and glucose in the tube side of the TMR. In order to allow the two ANN to discriminate between the training cases, the inlet feed flow rate is included as input variable. The output variables of ANN-1 and ANN-2 are K_m and $K_m^{'}$ respectively. The trained ANN (ANN-1 and ANN-2) with optimum configurations are then combined with the mass balance equations (ODEs) and simplified rate expressions (Equations 3.8 and

3.9). As shown in Figure 3.20, ANN-1 and ANN-2 receive as inputs the normalized concentrations of cellulose, cellobiose, and glucose in tube side and the feed flowrate. Then, predicate as outputs the normalized K_m and $K_m^{'}$. The de-normalized values of K_m and $K_m^{'}$ serve as an input to the simplified rate expressions. The calculated rates of formation of cellobiose and glucose (r_B and r_G) serve as an input to the mass balance equations (ODEs), which produces as output the concentrations of cellulose, cellobiose, and glucose in tube and shell sides. This step is repeated iteratively until all the 20 pairs of input/output vectors for each training case is included.

3.6.1 HANN2a

Two hybrid models were developed using this approach. The two ANN (ANN-1 and ANN-2) for the first HANN2 (HANN2a) model were trained using non-smoothed training data (no smoothing spline applied) as inputs. Also, the non-smoothed state variables and smoothed first derivative values were used in equations 3.10 and 3.11 to determine K_m and $K_m^{'}$ at each position z_i. Therefore, the training data for this model can be considered as partially non-smoothed data. A MATLAB script file (**hann2a_data.m**) in Appendix B6 was used to generate the training data for the ANN part of this model.

The structure of both ANN (ANN-1 and ANN-2) consists of one node in the output layer with the use of linear transfer function. A hyperbolic transfer function was used in the

hidden layer for both ANN. The number of nodes in the hidden layer was optimized during the training of both ANN using **trainbr.m** function. Table-3.7 and Table-3.8 summarized the training results of ANN-1 and ANN-2 obtained by using all training cases (1-20) and the interpolation cases (21-23) for testing.

Figures 3.16 and 3.17 illustrate the performance of different ANN structure developed for HANN2a. As shown in these tables and figures, the optimum number of nodes in the hidden layer is 4 for ANN-1 and ANN-2, since the effective numbers of parameters remain constant after this point. The trained ANN (ANN-1 and ANN-2) with optimum configurations were combined with the mass balance equations (ODEs) and simplified rate expressions (Equations 3.8 and 3.9) as shown in Figure 3.20. A MATLAB program, **hann2.m**, in Appendix B8 was developed to perform the combination. The performance of the HANN2a model was evaluated by using recall, interpolation, and extrapolation cases and the results of this evaluation are discussed in chapter 4.

3.6.2 HANN2b

The second HANN2 model (HANN2b) is similar to the previous one (HANN2a), however, smoothed (by smoothing-spline) data were used to train ANN-1 and ANN-2 for this model in order to compare its prediction to the predictions of BANN and HANN1 models using the same training data. The smoothed process data were used as inputs and the targets (K_m and $K_m^{'}$) were calculated from equations 3.10 and 3.11 using smoothed

first derivative and smoothed process data. A MATLAB script file (**hann2b_data.m**) in Appendix B7 was used to generate the training data for ANN part of this model.

The structure of the two ANN used in this model is similar to the ANN developed for HANN2a. The number of the nodes in the hidden layer was optimized during the training of both ANN using **trainbr.m** function. Table-3.9 and Table-3.10 summarized the training results of ANN-1 and ANN-2 obtained by using all training cases (1-20) and the interpolation cases for testing. Figures 3.18 and 3.19 illustrate the performance of different ANN structure developed for ANN-1 ANN-2. As shown in these tables and figures, the optimum number of nodes in the hidden layer is 5 for ANN-1, and 3 for ANN-2, because after this point the effective number of parameters remains constant. A MATLAB program, **hann2.m**, in Appendix B8 was used to combine the train ANN with the mass balance equations (ODEs) and simplified rate expressions. The performance of this model was evaluated by using recall, interpolation, and extrapolation cases and the results of this evaluation are presented in chapter 4. The differences of the four models (BANN, HANN1, HANN2a, and HANN2b) are summarized in Table-3.11

Table-3.7 ANN-1 Development for HANN2a Model

S	E_D	E_W	E_T	N	γ
1	0.07631	8.18	0.00835	6	5.57
2	0.06581	4.11	0.00720	11	9.09
3	0.06446	3.30	0.00699	16	11.90
*4	0.06404	3.10	0.00707	21	13.22
5	0.06406	3.01	0.00700	26	13.16
6	0.06409	5.26	0.00707	31	13.21
10	0.06340	3.56	0.00701	51	13.36
15	0.06396	3.30	0.00706	76	13.41

*Optimum number of nodes in the hidden layer.

Table 3.8 ANN-2 Development for HANN2a Model

S	E_D	E_W	E_T	N	γ
1	0.02754	5.501	0.00949	6	5.50
2	0.02640	4.93	0.00849	11	9.36
3	0.02480	6.71	0.00801	16	13.38
*4	0.02378	6.82	0.00800	21	15.78
5	0.02373	6.97	0.00806	26	15.69
6	0.02370	6.59	0.00804	31	15.69
10	0.02377	5.62	0.00806	51	15.53
15	0.02375	6.16	0.00802	76	15.93

*Optimum number of nodes in the hidden layer.

Table-3.9 ANN-1 Development for HANN2b Model

S	E_D	E_W	E_T	N	γ
1	0.036633	25.51	0.00586	6	5.20
2	0.036279	7.94	0.00569	11	10.08
3	0.033573	10.82	0.00511	16	13.00
4	0.030091	16.86	0.00486	21	18.52
*5	0.027956	31.90	0.00393	26	21.82
6	0.027818	15.55	0.00393	31	21.81
7	0.027983	14.16	0.00392	36	21.31
10	0.027985	14.33	0.00398	51	21.61
15	0.027730	15.19	0.00399	76	21.61

*Optimum number of nodes in the hidden layer.

Table-3.10 ANN-2 Development for HANN2b Model

S	E_D	E_W	E_T	N	γ
1	0.008278	54.5130	0.00558	6	5.50
2	0.007944	5.47865	0.00509	11	9.15
*3	0.005757	7.80884	0.00286	16	13.13
4	0.005744	7.95868	0.00287	21	13.22
5	0.005755	7.81909	0.00281	26	13.49
10	0.005749	6.03844	0.00284	51	13.51
15	0.005818	5.09061	0.00287	76	13.73

*Optimum number of nodes in the hidden layer.

Table-3.11 Models Summary

	Training Data		First-Principle relations used	
	Data Smoothed	First-Derivatives Smoothed	Mass Balance Equations	Simplified Rate Expressions
BANN	Yes	Not Applicable	Not Applicable	Not Applicable
HANN1	Yes	Yes	Yes	Not Applicable
HANN2a	No	Yes	Yes	Yes
HANN2b	Yes	Yes	Yes	Yes

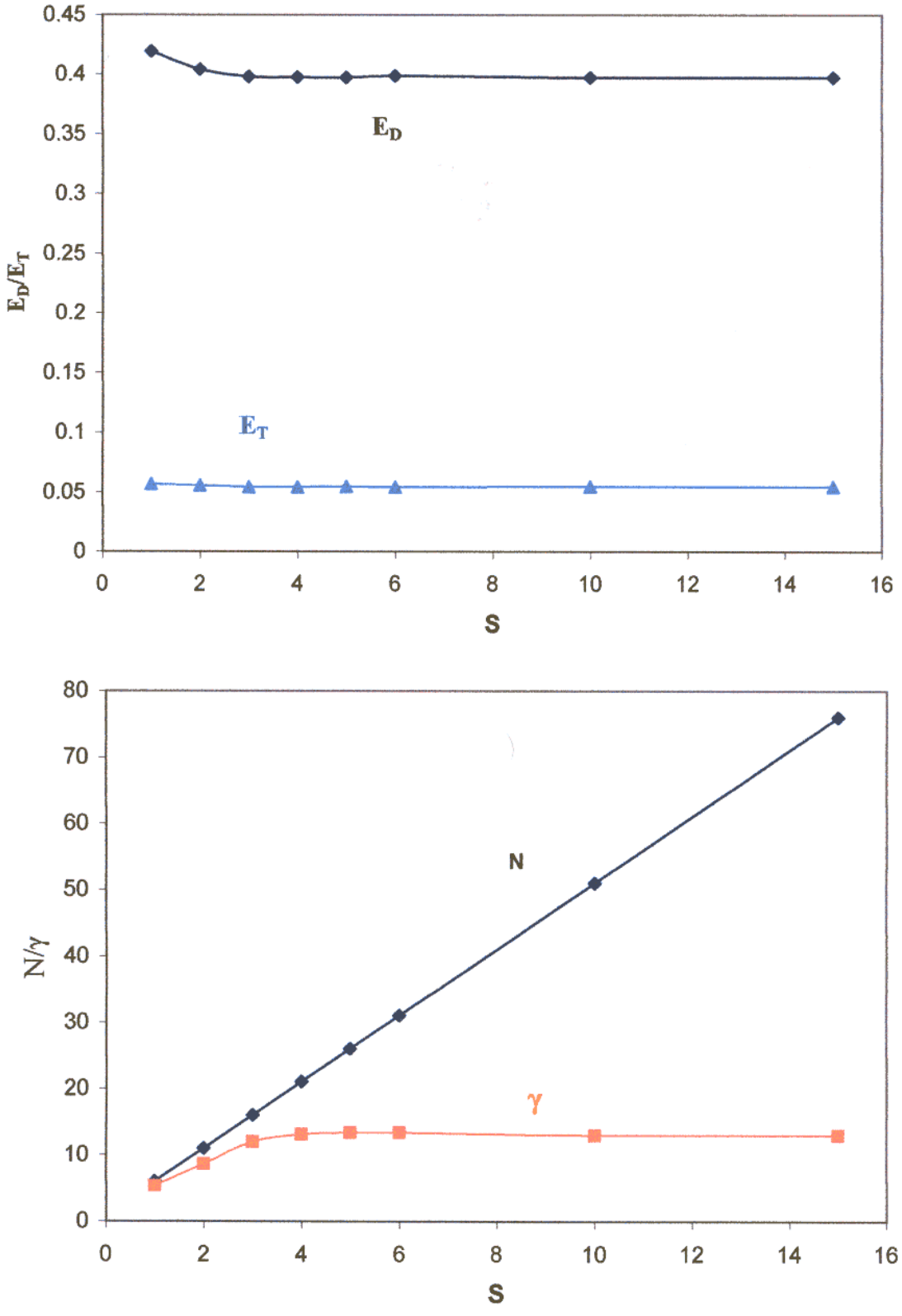

Figure 3.16 ANN-1 development for HANN2a

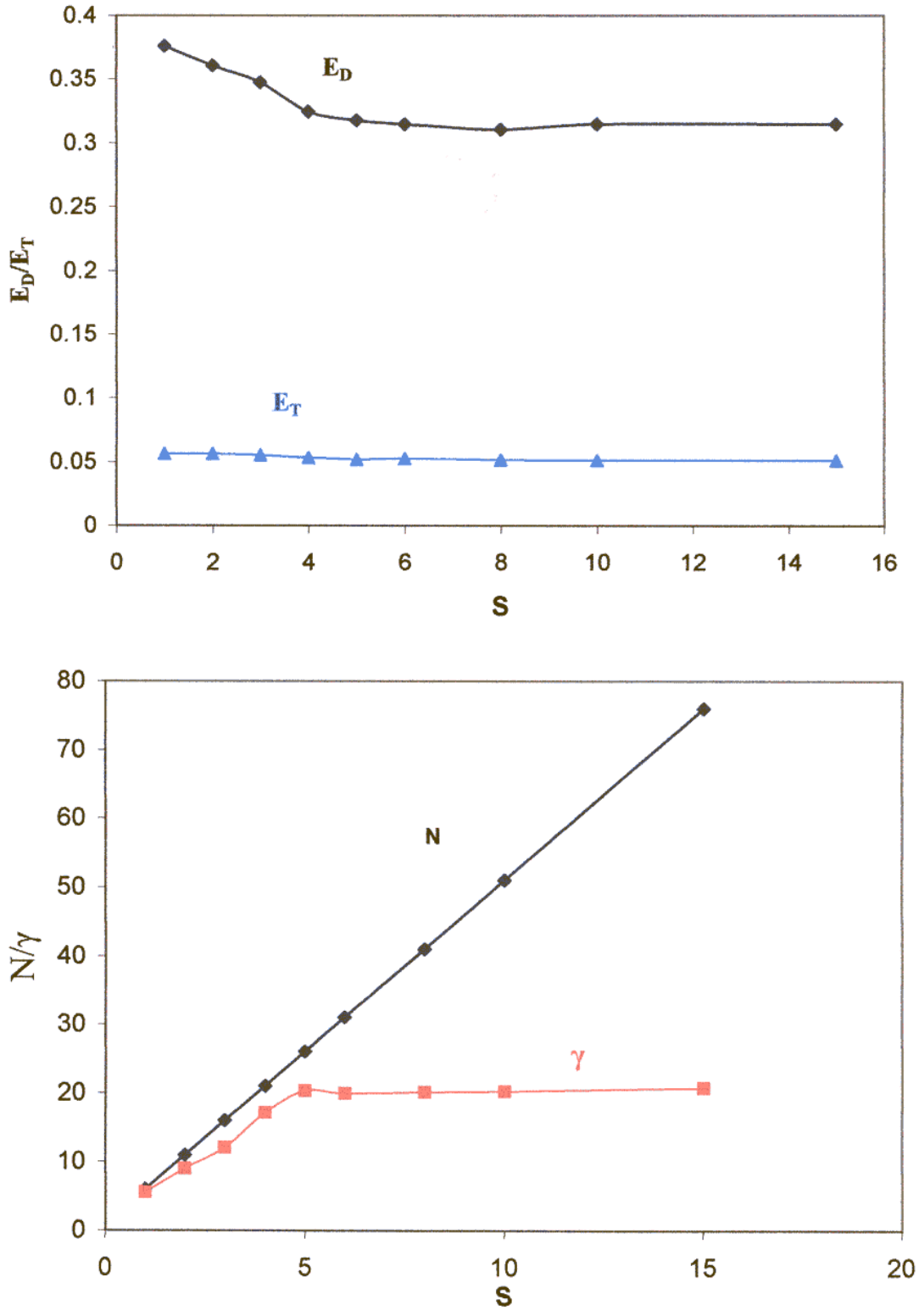

Figure 3.17 ANN-2 development for HANN2a

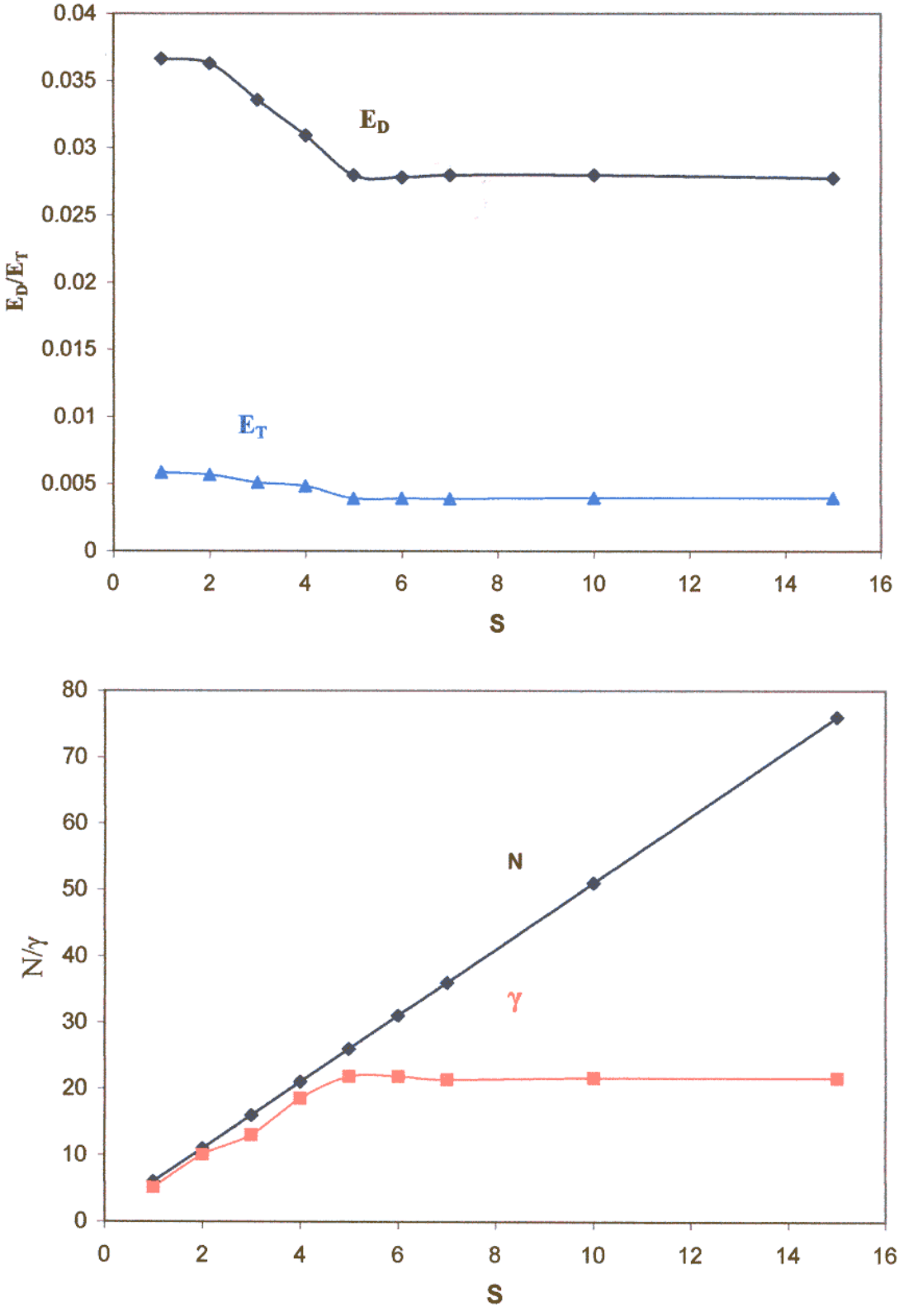

Figure 3.18 ANN-1 development for HANN2b

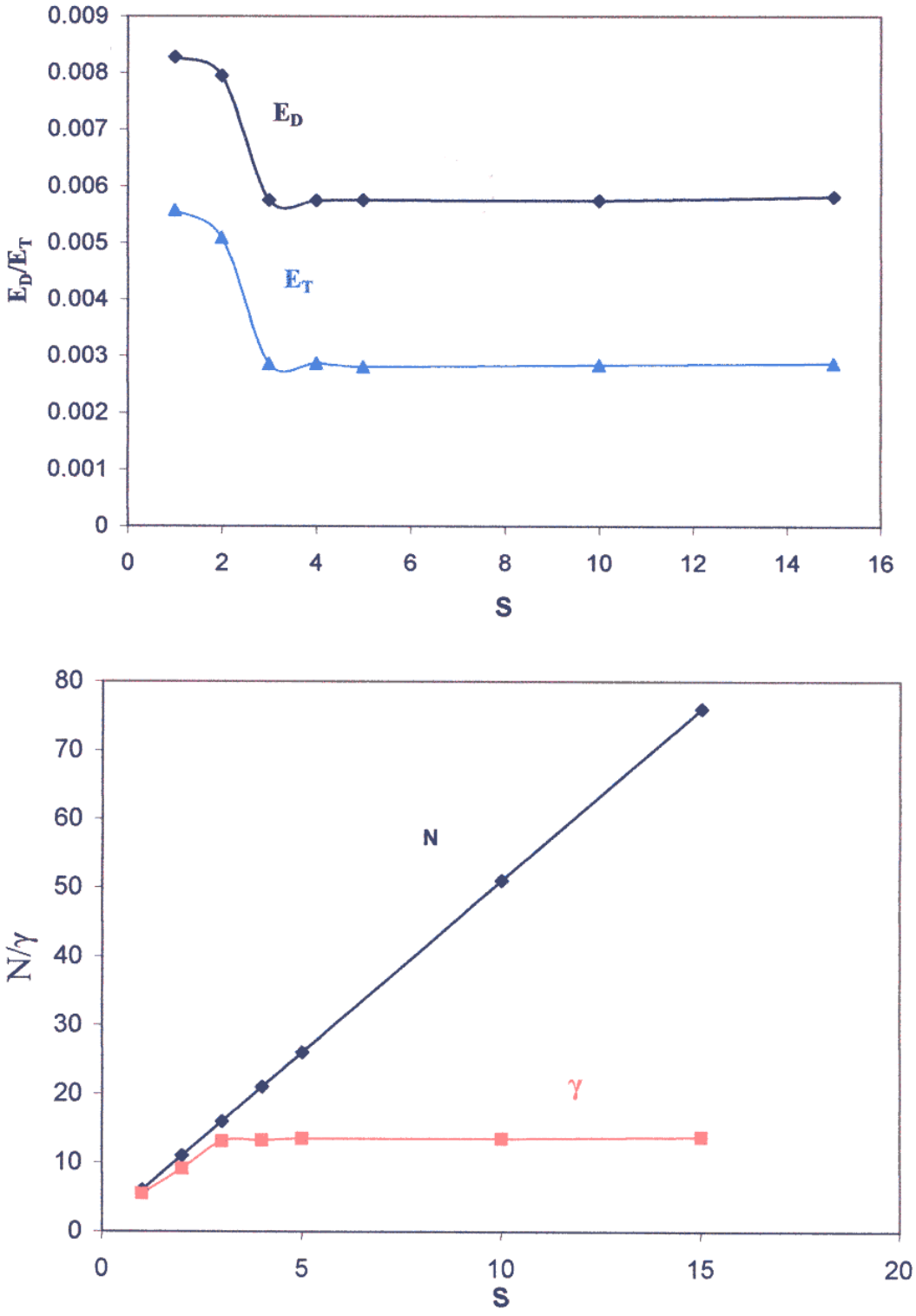

Figure 3.19 ANN-2 development for HANN2b

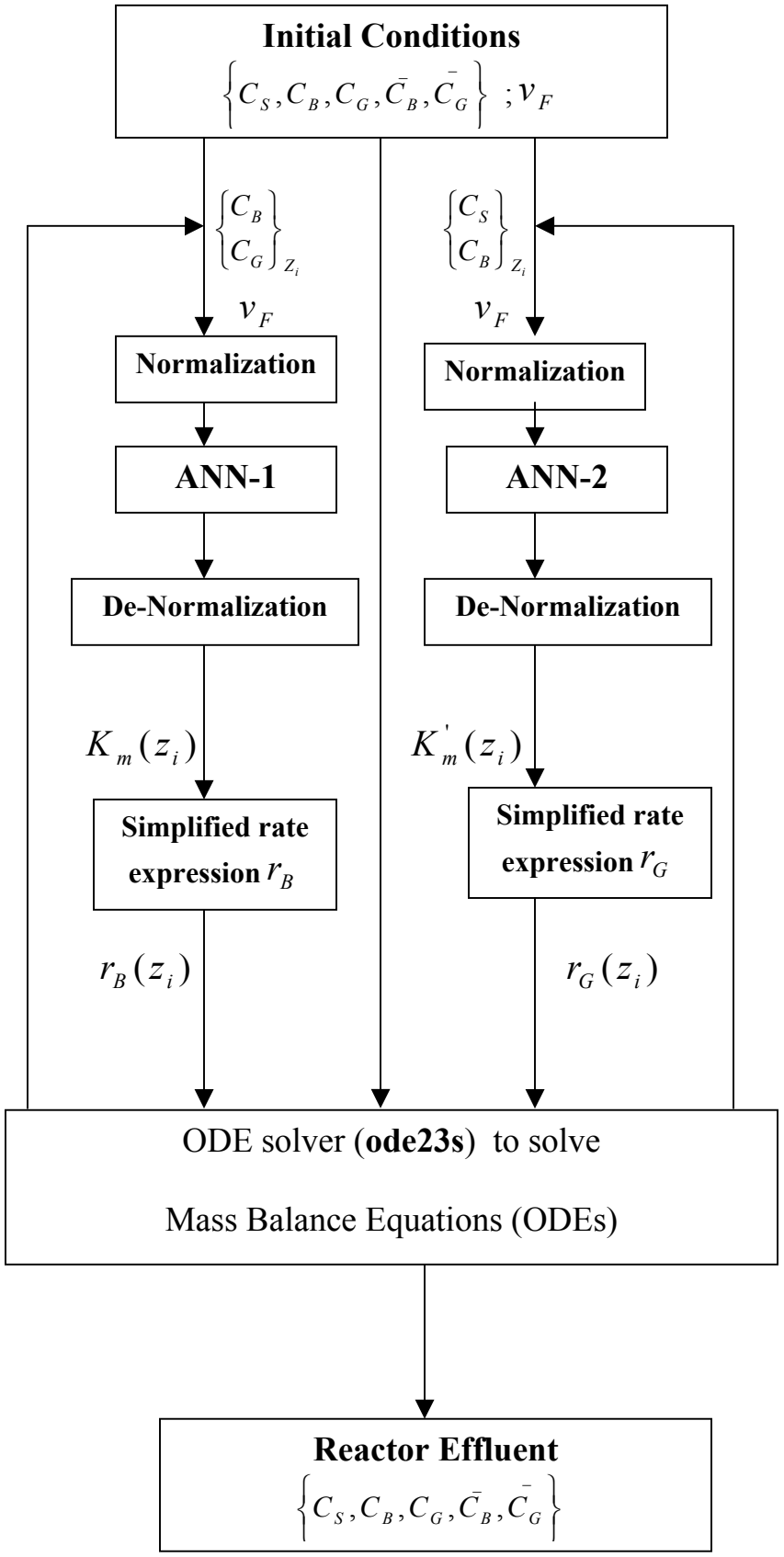

Figure 3.20 Combination of ANN with ODEs for HANN2 model.

4 Performance Comparisons of BANN, HANN1, and HANN2

One of the most important aspects in developing ANN is to determine how well the ANN performs once training is complete. Checking the performance of a trained ANN involves two steps: (1) How well the ANN "recall" the predicted responses (output vector) from the same data sets used to train the ANN; and (2) How well the ANN predicts responses from data sets that were not used in training. This usually involves "interpolation", if the data sets used in this step is within the range of the training data sets, or "extrapolation", if otherwise. Case 1 from Table 3.3 was selected to test the ability of the ANN to recall the training data since it was the first case used to train the ANN for both modeling approaches. The last six cases in Table 3.3 were used for generalization step, three cases (21-23) for interpolations, and three cases (24-26) for extrapolations. The interpolation cases were selected, as shown in Figure 3.3, from different operation conditions within the training data. The extrapolation cases were selected to be faraway from the training conditions and also to represent different operating conditions (see Figure 3.1). Qualitative and quantitative comparisons of BANN, HANN1, HANN2a, and HANN2b were performed and the results are presented in the next two sections.

4.1 Qualitative Comparison of BANN, HANN1, and HANN2

The qualitative comparison was performed by plotting predictions of BANN, HANN1, HANN2a, and HANN2b models versus process data of recall, interpolation and extrapolation cases. One plot is given as a sample from each of the testing regimes: recall

(case1), interpolation (case21), and extrapolation (case25). Each plot consists of the predicted and process profiles for each of the state variables.

Figures 4.1 - 4.3 present a sample of the results for BANN model. As shown in these Figures, the results are in a good agreement with process data in recall and interpolation cases (Figures 4.1 and 4.2). The good modeling performance on these cases indicates that the ANN was trained properly. However, The BANN model was seen to perform poorly on extrapolation case (Figure 4.3). The poor model predictions of BANN on the extrapolation case may indicate that the reaction system is too complex to be adequately modeled using this BANN model. Because it is a difficult modeling task for ANN, requiring the determination of mass balance and reaction rates based only on feed conditions of TMR. Also, It is a very challenging test, since any errors made near the entrance region of the TMR will propagate through the entire reactor.

Figures 4.4 – 4.6 show a selection of the results obtained using the HANN1 model. Similar to BANN model, the predictions of this model in recall and interpolation cases (Figures 4.4 and 4.5) are in a very good agreement with process data. However, the over all prediction is less accurate in the extrapolation case although the gross trends are correct. For example, as shown in Figure 4.6, HANN1 over-predict the concentration profiles of cellobiose and glucose in the shell side. This seems to be due to an over prediction of rate of productions of cellobiose and glucose in this case. This problem can occur because in this modeling approach, there is no inclusion of biochemical knowledge.

Figures 4.7 - 4.9 present typical results obtained using HANN2a model. The predicted profiles of this model are seen to be very close to the process data in almost all cases. It was initially expected that the prediction of this models would have larger errors, compared with HANN2b, because the presence of noise in the training data (partially non-smoothed) will degrade the interpolation and extrapolation abilities of this model. However, including basic biochemical knowledge, in the form of simplified rate expressions, allowed this model to be accurately interpolated and extrapolated.

Figures 4.10 – 4.12 show a selection of the results obtained using the HANN2b model. The performance of this hybrid model is excellent for recall, interpolation, and extrapolation cases. The predicted profiles are very close to the deterministic model profiles. The modeling performance in the extrapolation cases (Figure 4.12) is surprisingly good; it is a direct result of the contribution of the first-principle parts. Qualitatively, the second hybrid scheme (HANN2a and HANN2b) is the best of the three modeling schemes in terms of overall predictive ability. Quantitative comparisons are presented in the next section.

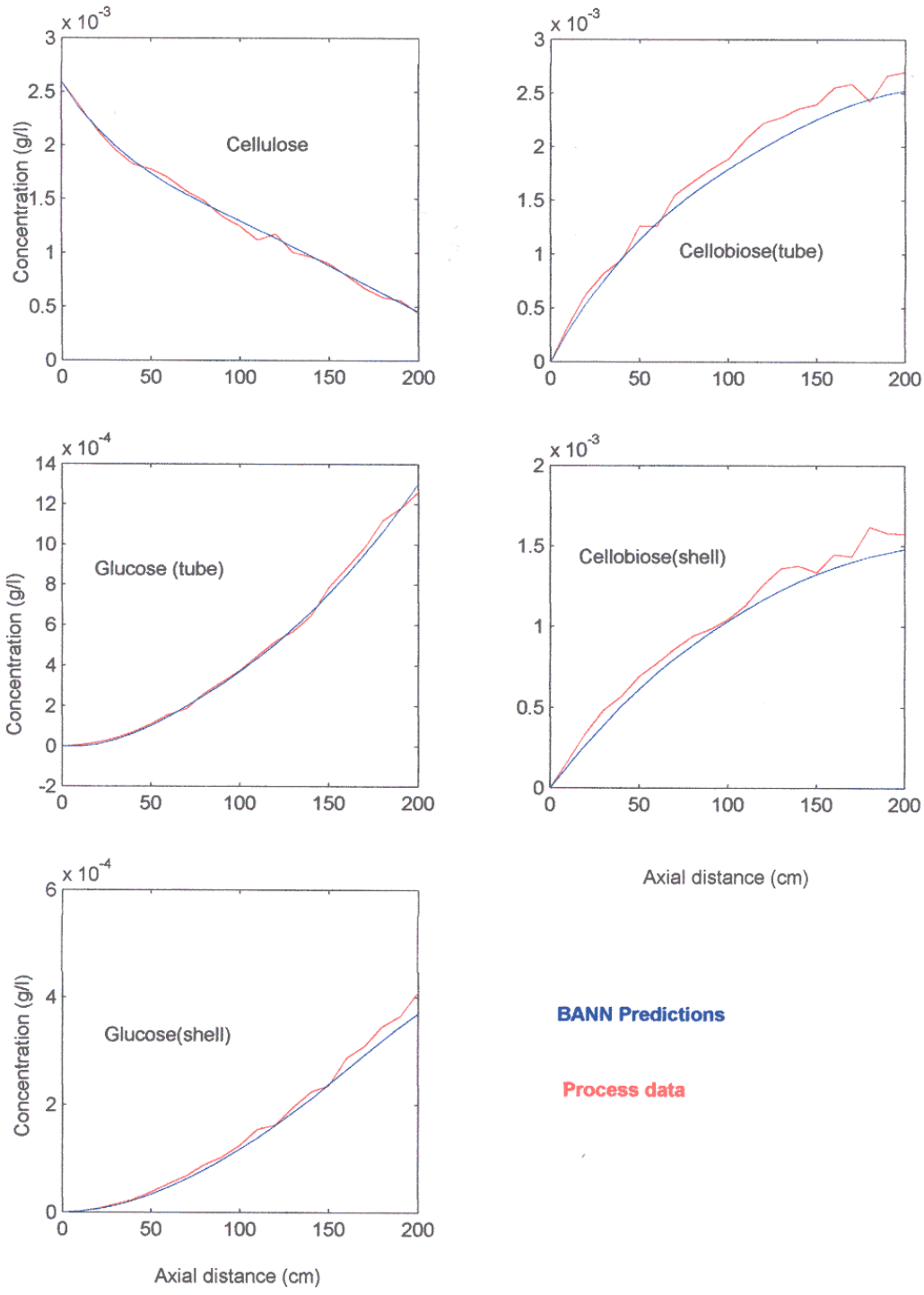

Figure 4.1 Comparison between process data and BANN predictions for recall case (case 1)

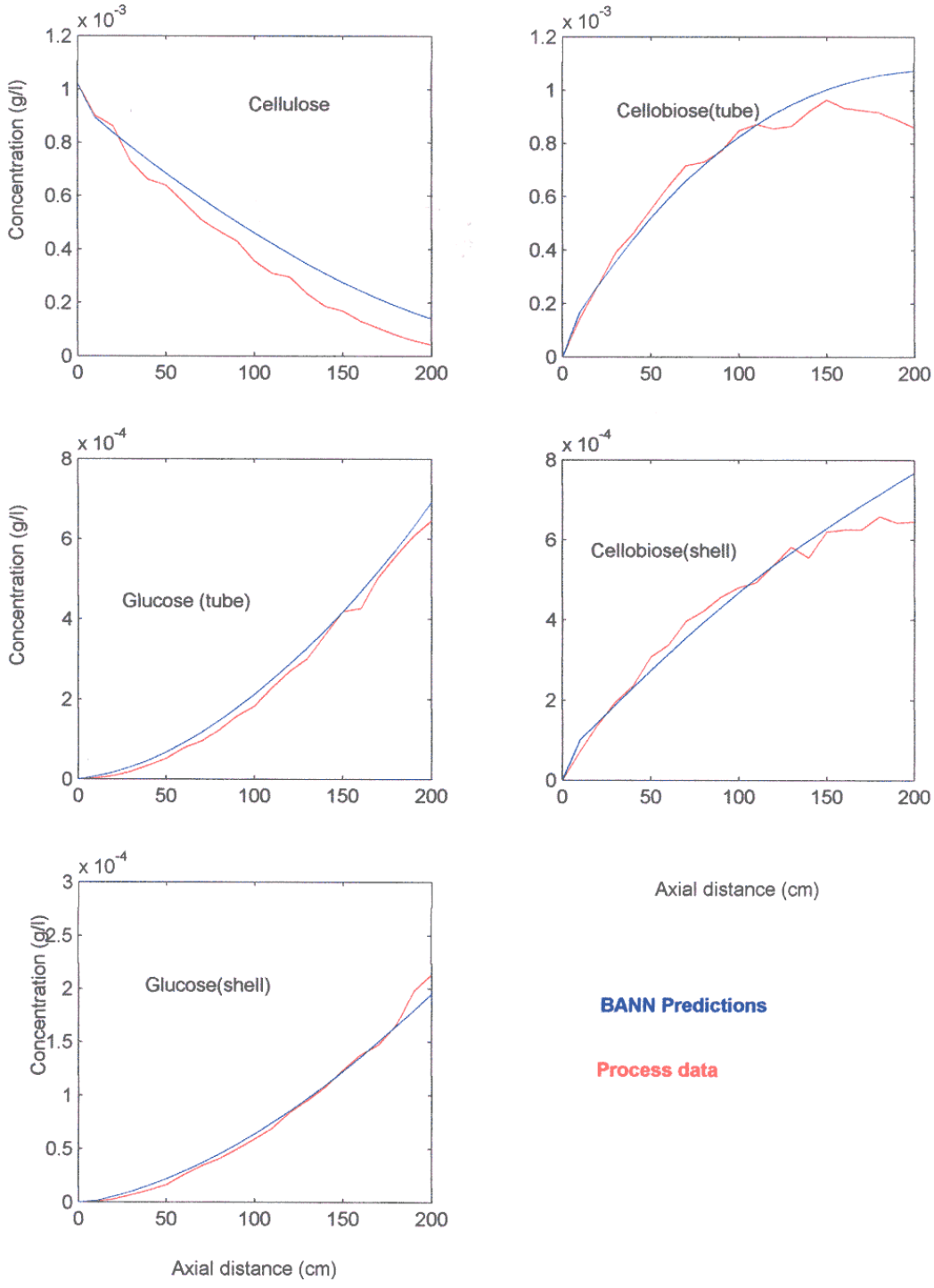

Figure 4.2 Comparison between process data and BANN predictions for interpolation case (case 21)

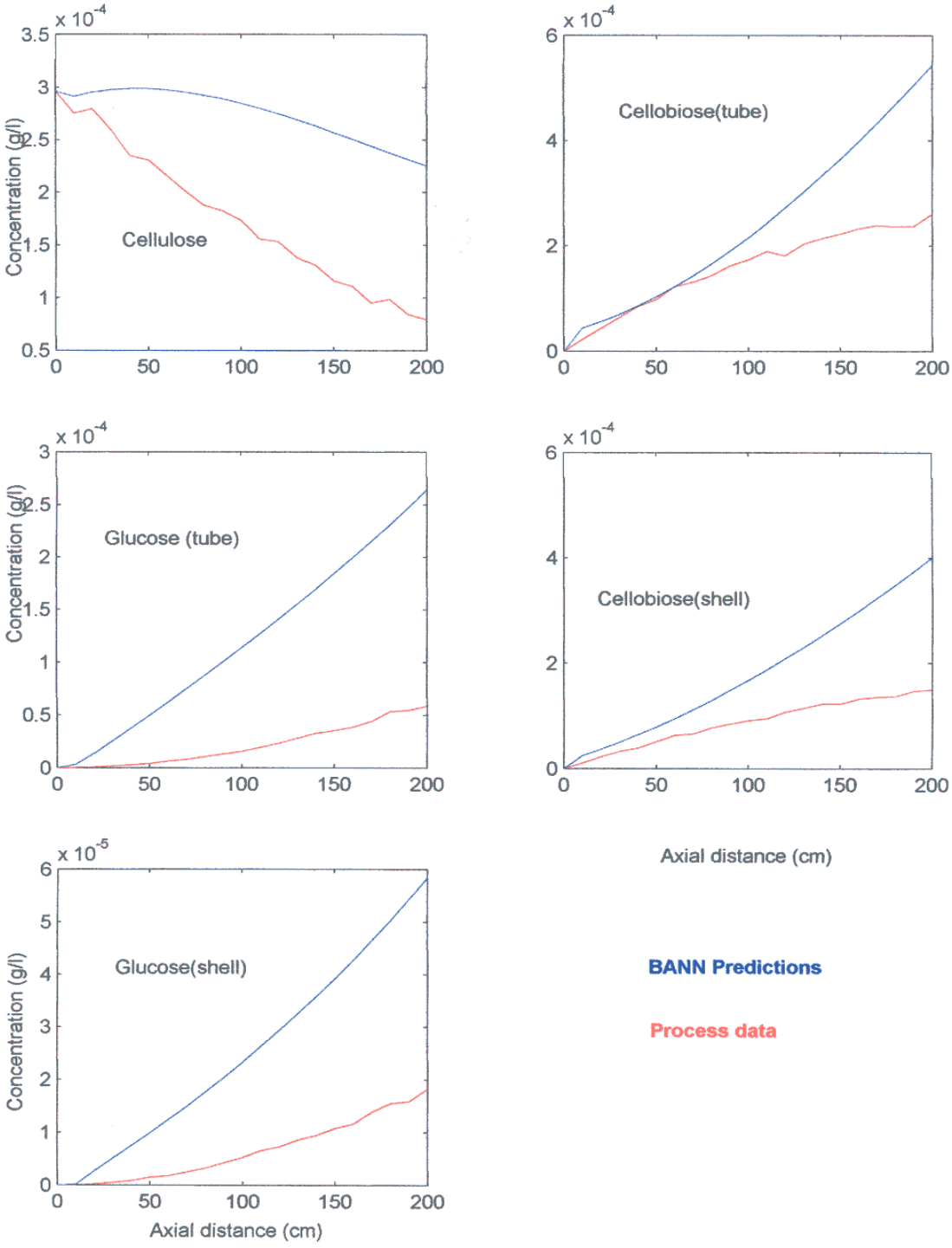

Figure 4.3 Comparison between process data and BANN predictions for extrapolation case (case 25)

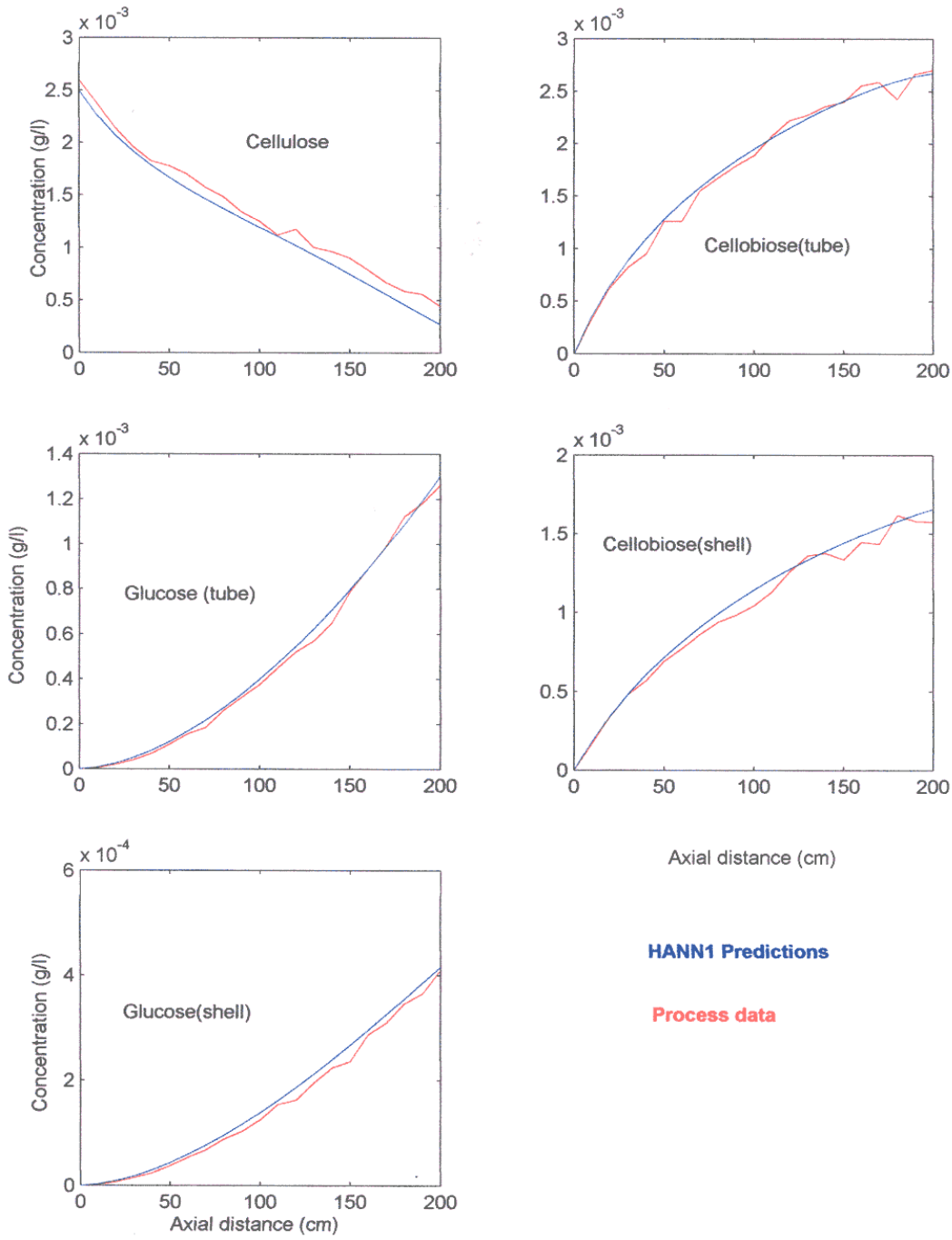

Figure 4.4 Comparison between process data and HANN1 predictions for recall case (case 1)

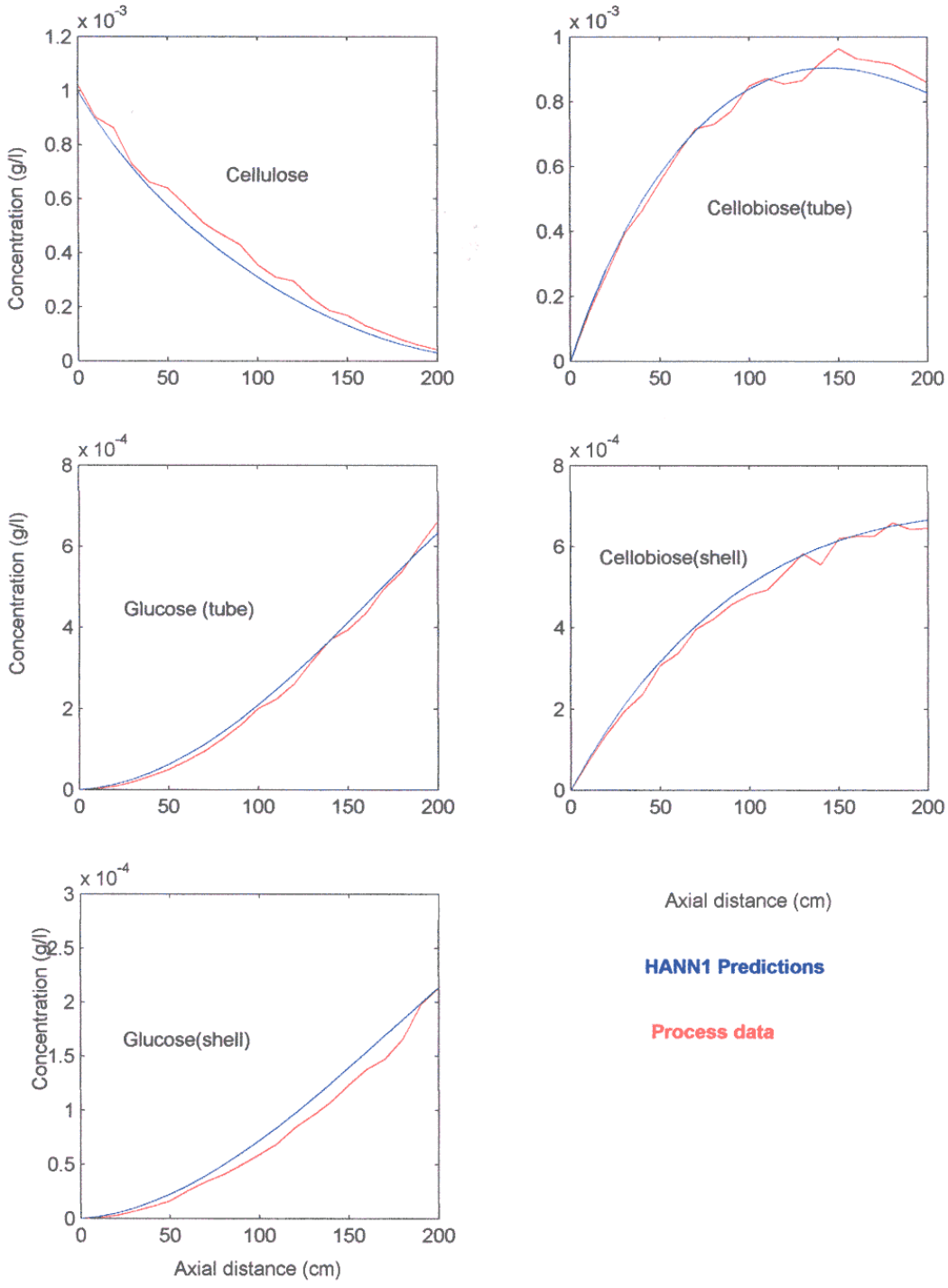

Figure 4.5 Comparison between process data and HANN1 predictions for interpolation case (case 21)

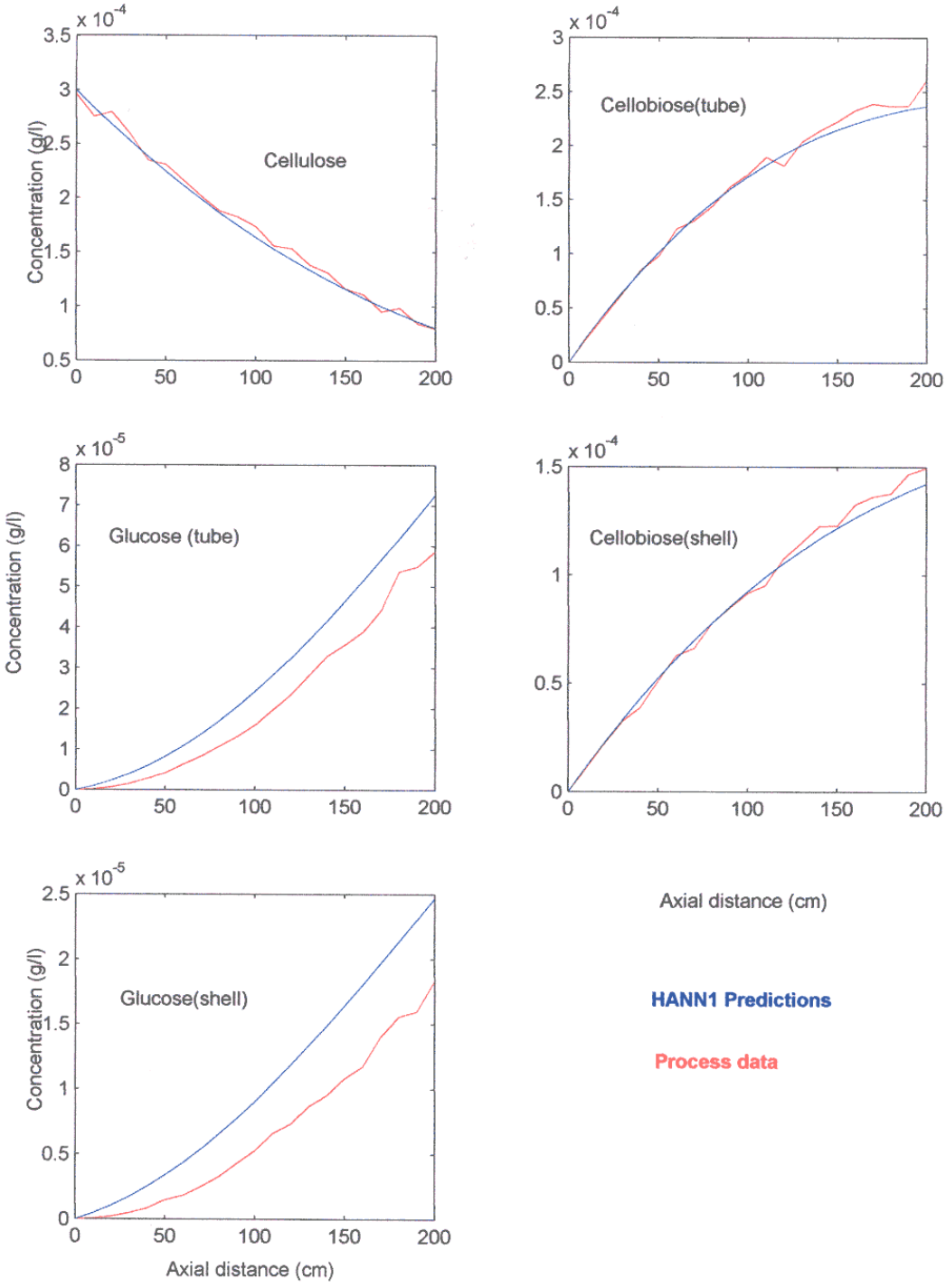

Figure 4.6 Comparison between process data and HANN1 predictions for extrapolation case (case 25)

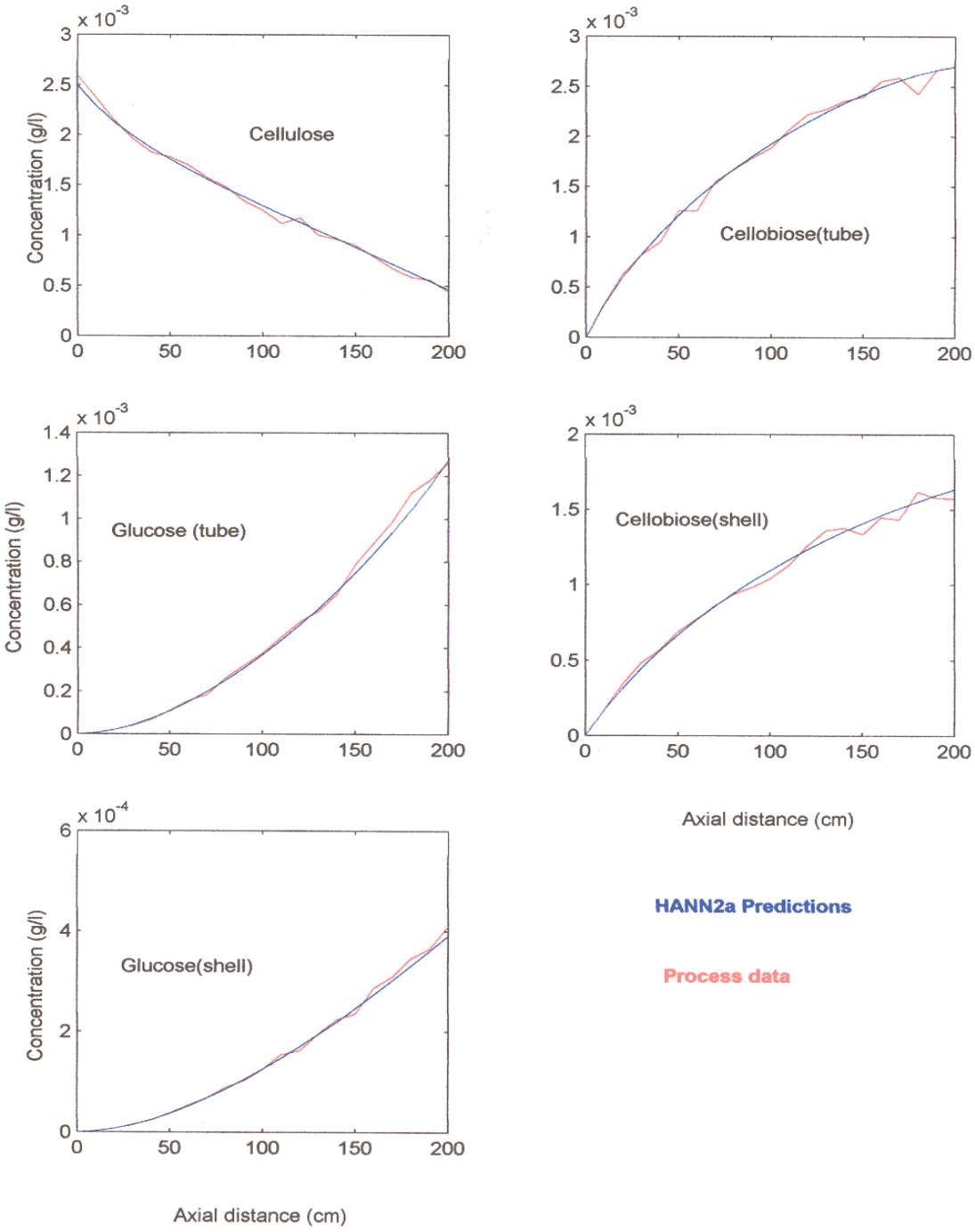

Figure 4.7 Comparison between process data and HANN2a predictions for recall case (case 21)

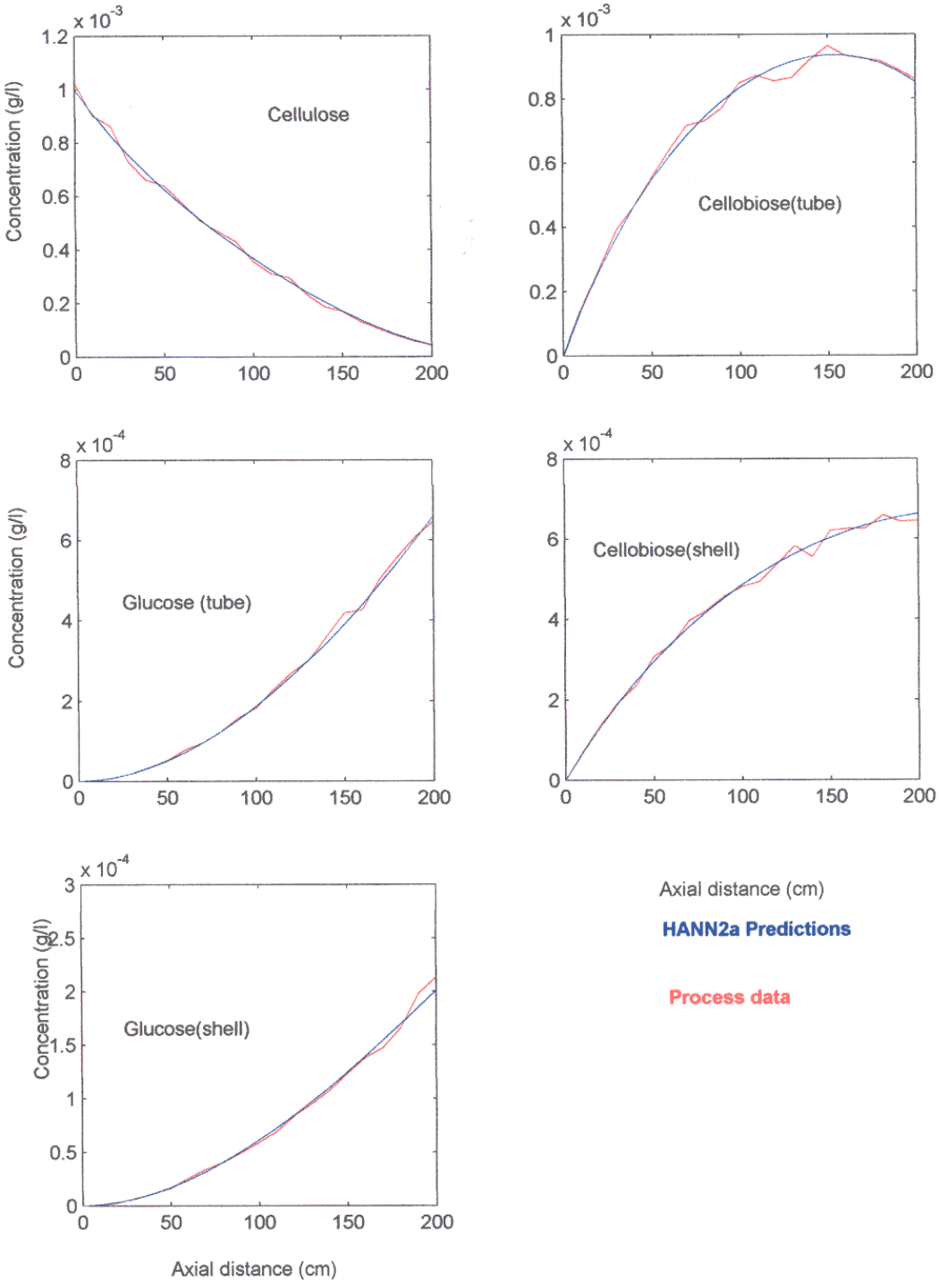

Figure 4.8 Comparison between process data and HANN2a predictions for interpolation case (case 21)

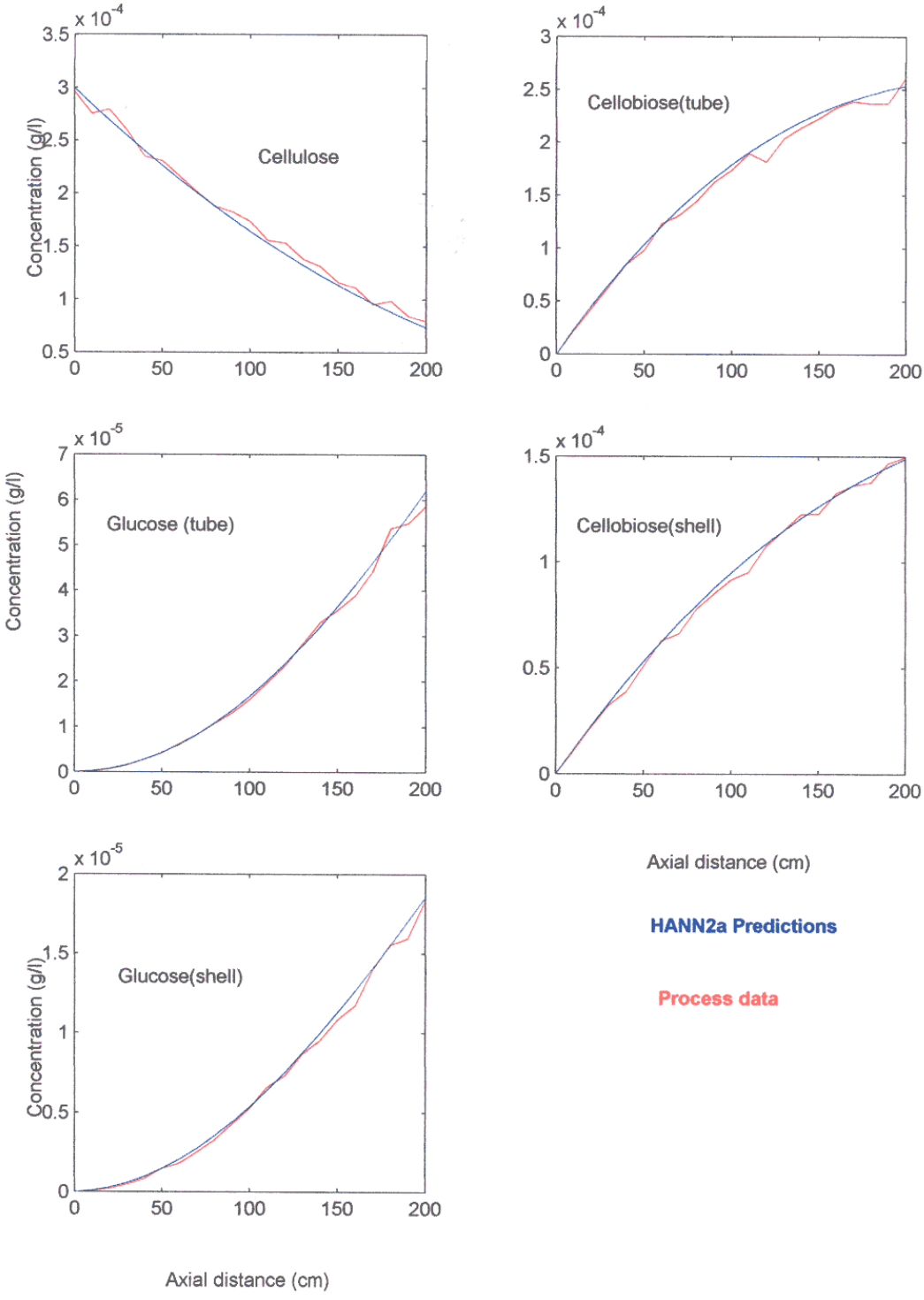

Figure 4.9 Comparison between process data and HANN2a predictions for extrapolation case (case 25)

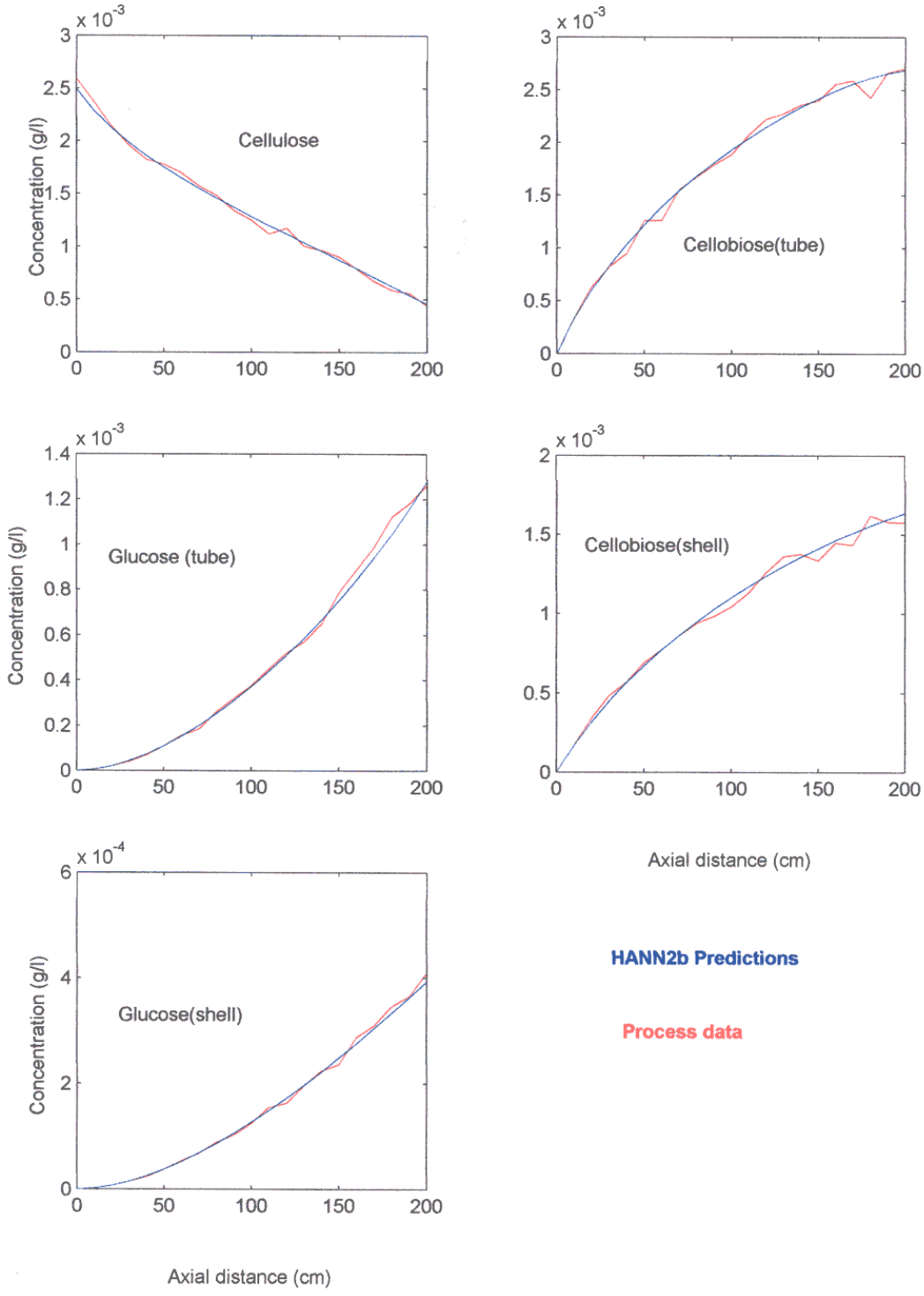

Figure 4.10 Comparison between process data and HANN2b predictions for recall case (case 1)

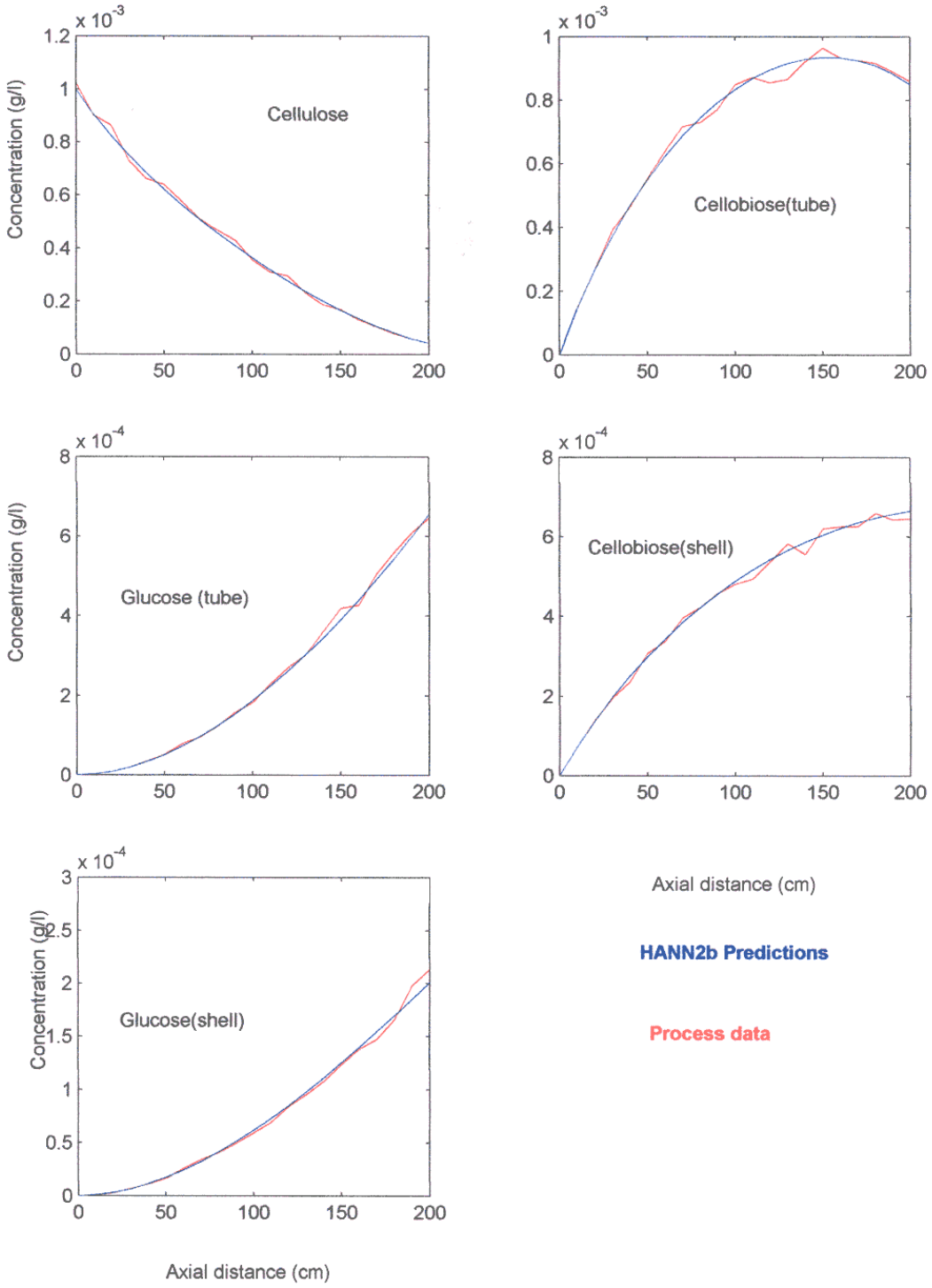

Figure 4.11 Comparison between process data and HANN2b predictions for interpolation case (case 21)

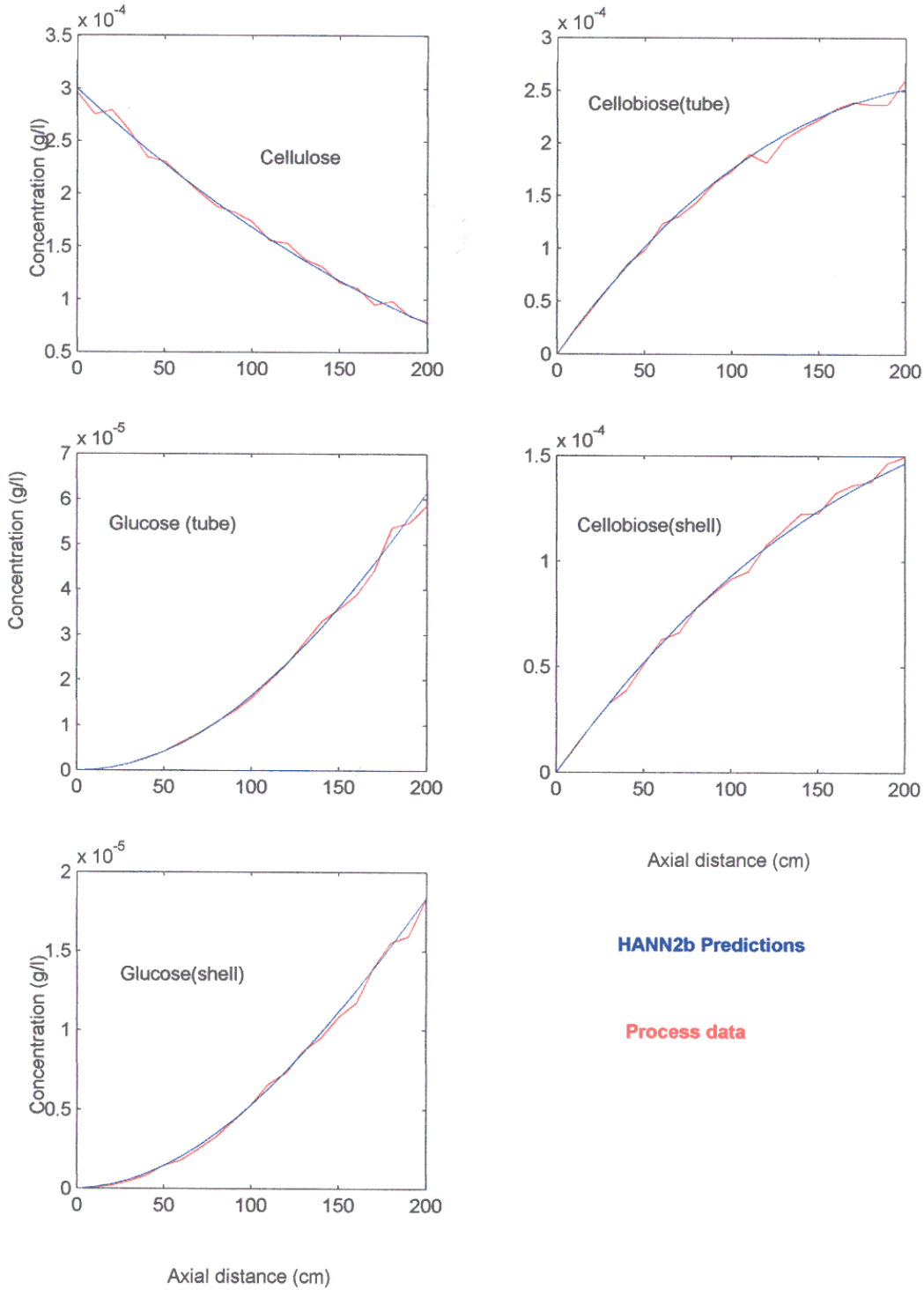

Figure 4.12 Comparison between process data and HANN2b predictions for extrapolation case (case 25)

4.2 Quantitative Comparison of BANN HANN1, and HANN2

In order to compare the performance of the three modeling schemes, a quantitative measure of their accuracy is needed. This was done first by performing regression analysis between process data and ANN predictions for all models. This analysis was done by using **postrg.m,** a MATLAB build-in function. Samples of these analyses for recall, interpolation, and extrapolation cases for glucose in the shell side are shown in Figures 4.13 - 4.16. As shown in these Figures, all models give a very good agreement with process data for recall and interpolation cases. However, Figures 4.13 and 4.14 show very poor model predictions of glucose concentration profiles in the shell side using BANN and HANN1 models in the extrapolation case (case 25). On the other hand, the prediction quality of both HANN2a and HANN2b models remain roughly the same when the models were used for interpolation and extrapolation (Figures 4.15 - 4.16); this is a benefit of including the first-principle parts. The assumed rate expressions clearly allow the ANN in the second hybrid scheme part to successfully emulate the TMR system.

Normalized root mean square error (NSM) associated with the predictions of the four models was also used to compare the performance of the three modeling schemes. NSM was calculated for each variable using the following equation [17].

$$NSM = \frac{\left[\dfrac{\sum_{i=1}^{N}(t_i - b_i)^2}{N}\right]^{1/2}}{\dfrac{\sum_{i=1}^{N} t_i}{N}} \qquad (4.1)$$

where N in this equation is the number of data points for each variable in each case (training, interpolation, and extrapolation), t_i , is the desired output, and b_i, is the output calculated via HANN1, HANN2 or BANN model.

The NSM is more strongly influenced by errors when the values of the state variables are relatively large in magnitude. Therefore, the median percent error (MPE) was used as a second measure of the performance of the three schemes. The MPE is less popular as a measure of ANN performance than NSM. However, when compared with NSM, it is less susceptible to being dominated by one or two terms with a large error [5,17]. MPE was calculated for each variable using the following equation,

$$MPE = median\left(\frac{\left|t_i - b_i\right|}{b_i}\right) \qquad (4.2)$$

Once both of these error measurements were calculated for all of the 26 cases, their averages were determined for each state variable for each testing regime (recall,

interpolation, and extrapolation). The results of the averages of two different error calculations, NSM and MPE respectively, for concentrations of cellulose, cellobiose, and glucose in tube and shell sides over twenty cases (1-20) for recall, three cases (21-23) for interpolation, and three cases (24-26) for extrapolation, are illustrated in Figures 4.17 - 4.26.

As shown in these figures, the average errors (NSM and MPE) associated with predictions of all models were relatively low in recall and interpolation cases, and the HANN2b model was always the lowest. The BANN model had problem with prediction of the extrapolation cases. This caused the average NSM and MPE for this model to be very large compare to the hybrid models. On the other hand, HANN1 model had low average errors compared to the BANN model in all extrapolation cases. This may indicate that the first principle part (mass balance) of this model has allowed the ANN part to capture the underlying behavior. Therefore, the HANN1 model is expected to perform much better than the BANN model in the extrapolation cases since the prediction is only in the kinetic parts of the process, while the mass balance remains unchanged.

It can be seen from these figures that the average errors associated with the predictions of HANN2a were always lower than the error associated with BANN and HANN1, even though the training data used to develop ANN for this model were partially non-smoothed. As mentioned before, it was initially expected that this model would have larger average errors when used for extrapolation. This was not seen for all cases in this investigation, which confirms the superiority and capability of this modeling scheme. Also, HANN2a performs well for all state variables, for all testing regimes. In fact, on the

basis of average NSM and MPE errors, predictions using this model have the lowest average error on all state variables. These results show clearly that the inclusion of first principles and basic biochemical knowledge, in the form of mass balances and the simplified rate expressions, have allowed the HANN2 scheme to perform consistently well on all predictions. Therefore, the quantitative comparisons support the conclusions drawn from qualitative comparisons described in section 4.1.

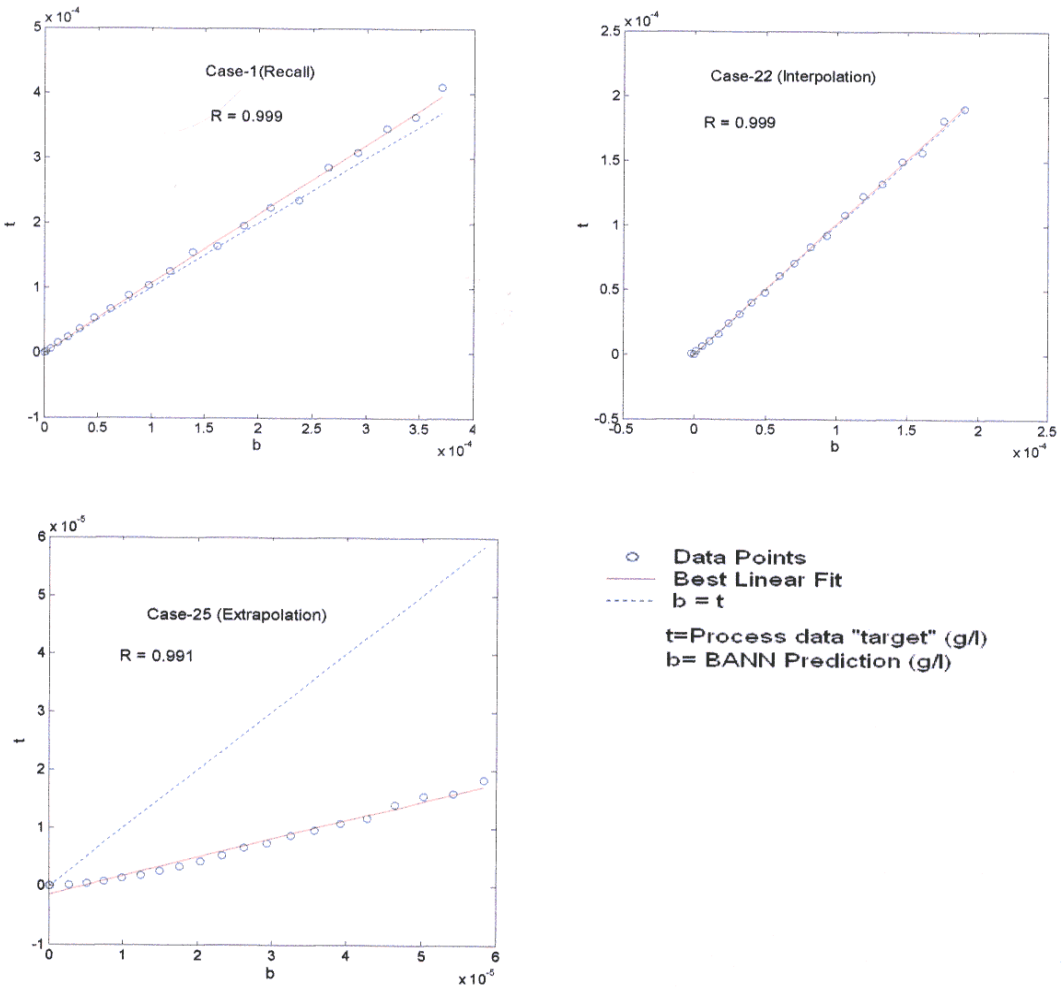

Figure 4.13 Regression results for glucose in shell side using BANN model

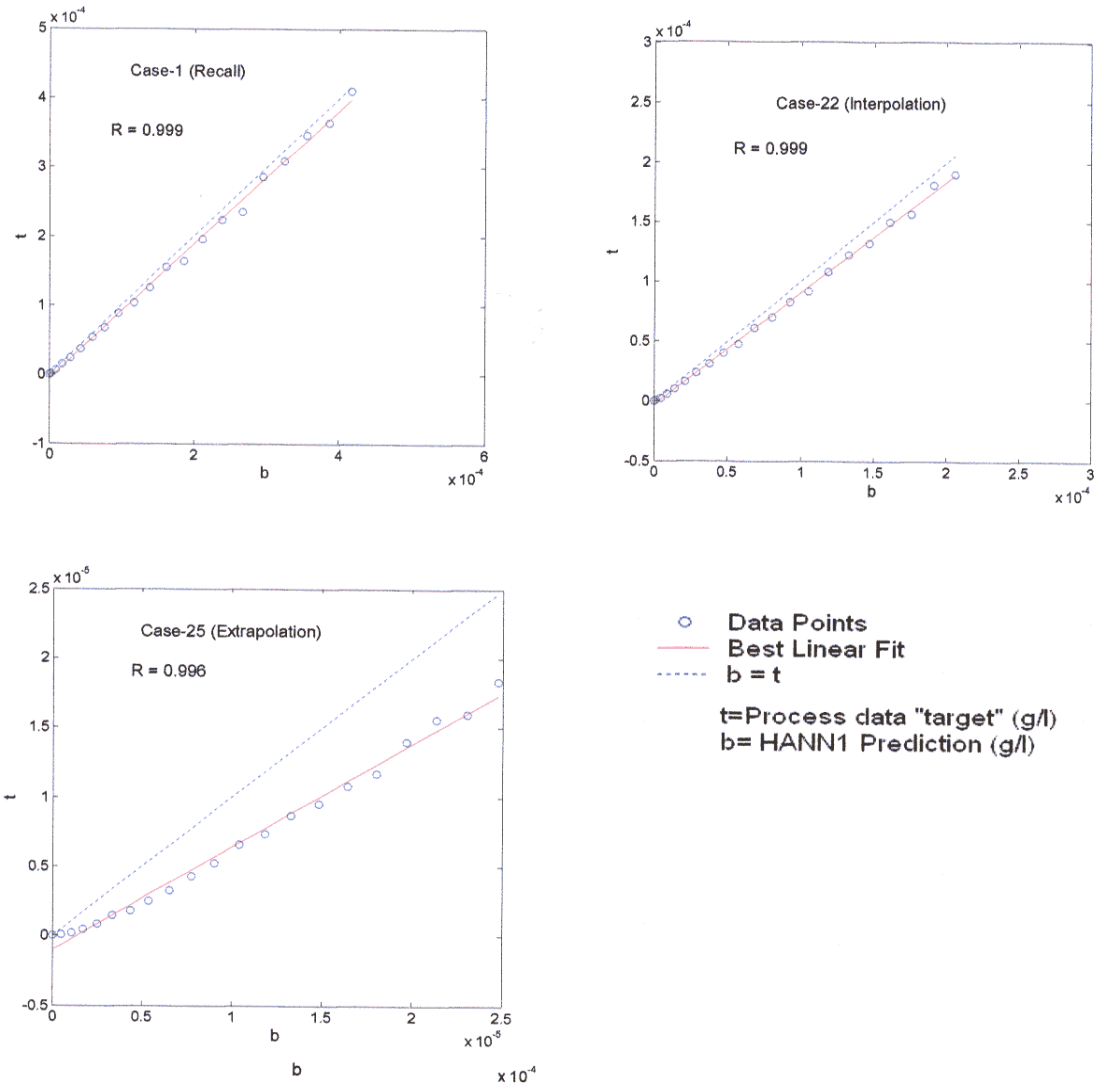

Figure 4.14 Regression results for glucose in shell using HANN1 model

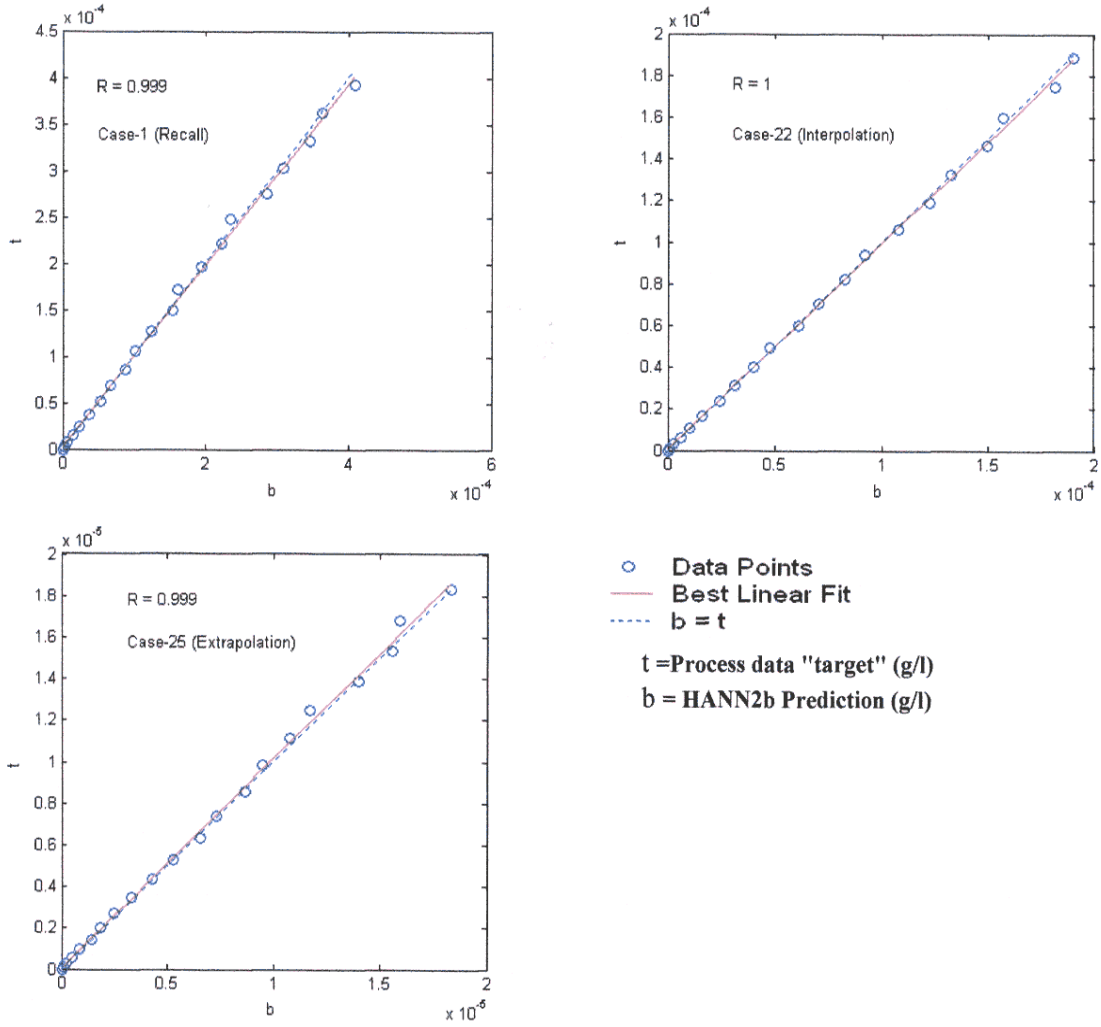

Figure 4.16 Regression results for glucose in shell using HANN2b model.

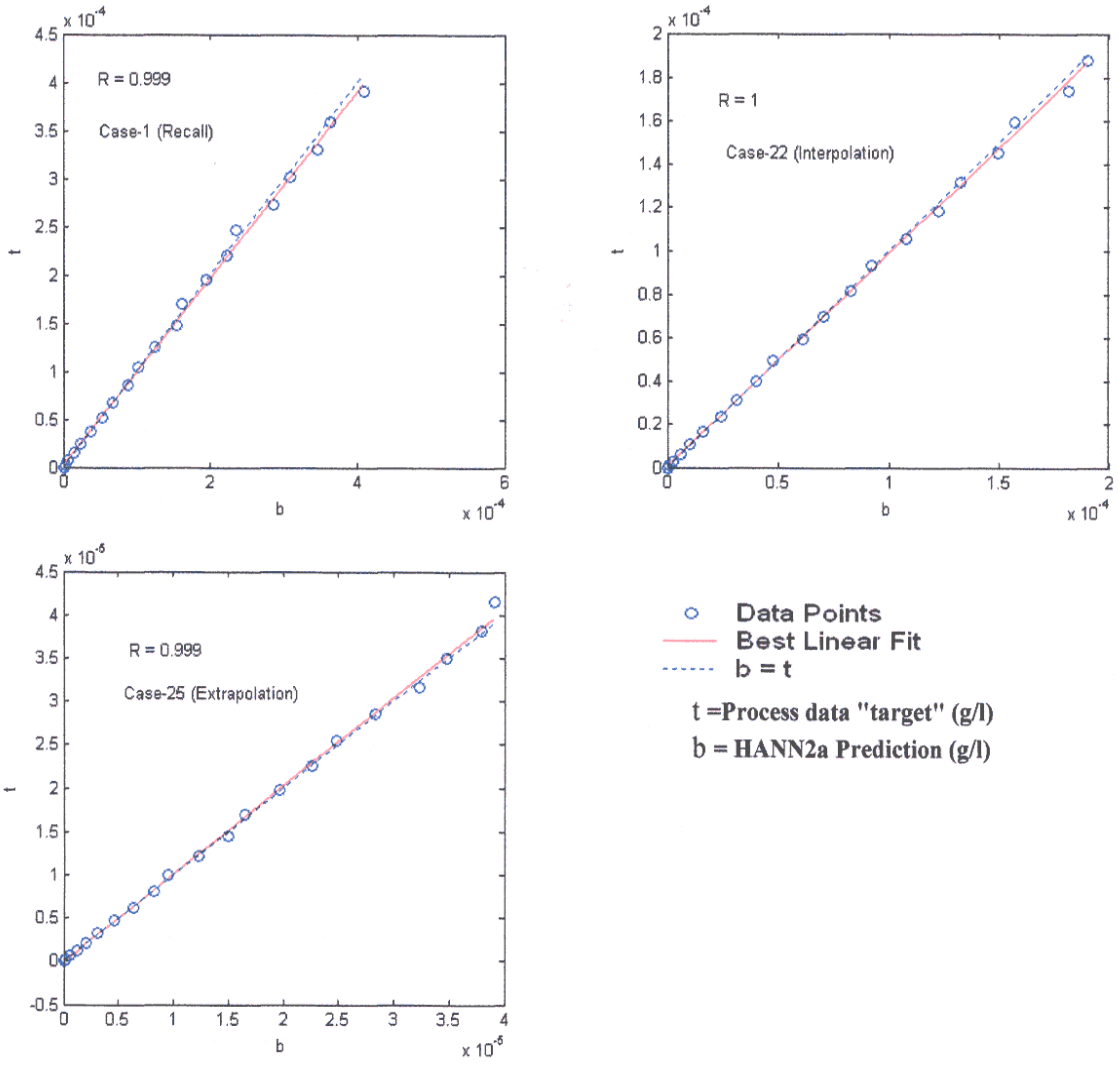

Figure 4.15 Regression results for glucose in shell using HANN2a model.

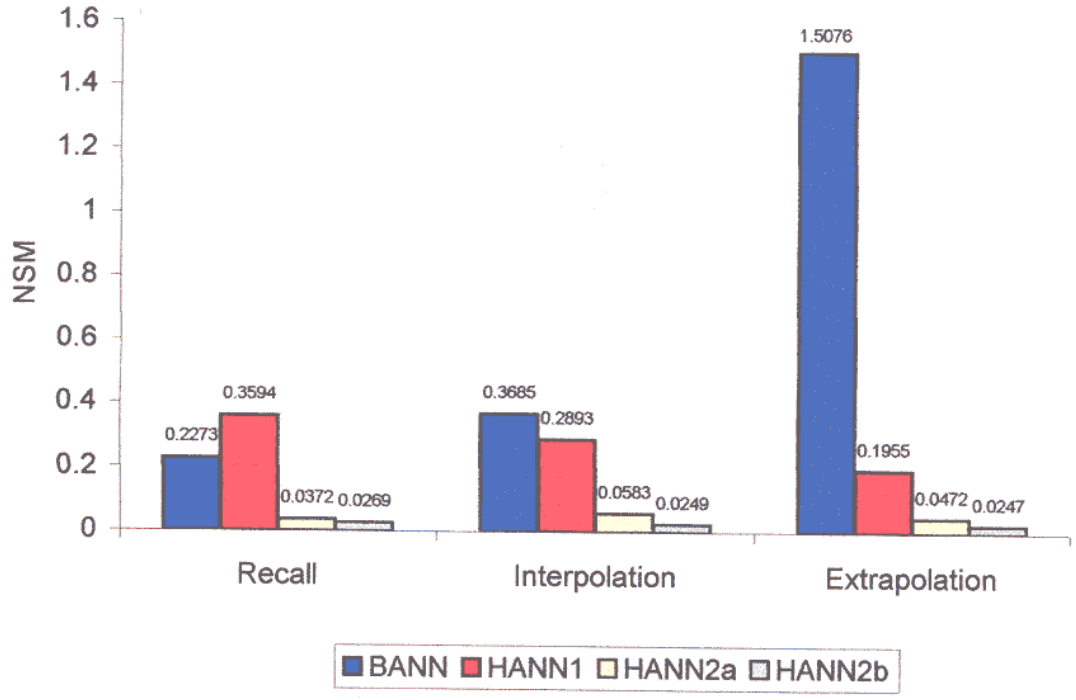

Figure 4.17 Average NSM for cellulose concentration predictions

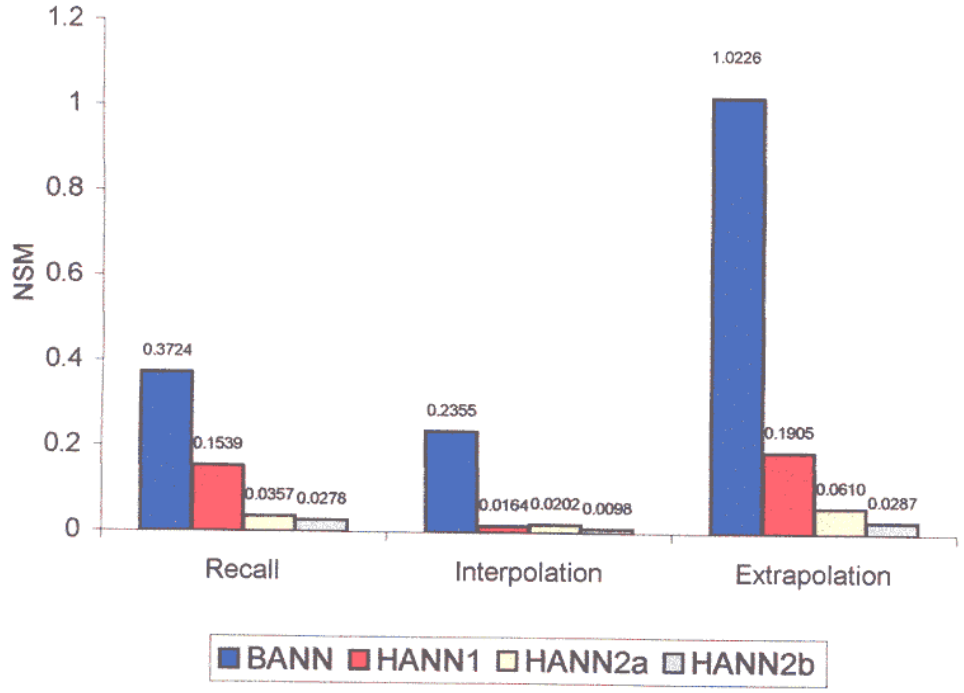

Figure 4.18 Average NSM for cellobiose (tube) concentration predictions

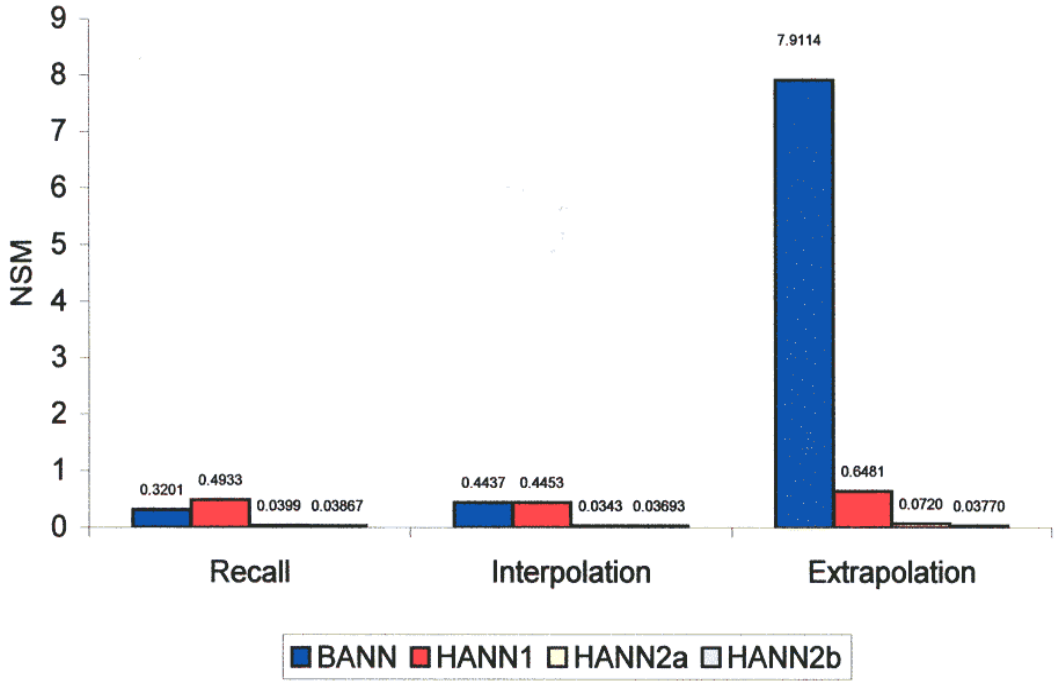

Figure 4.19 Average NSM for glucose (tube) concentration predictions

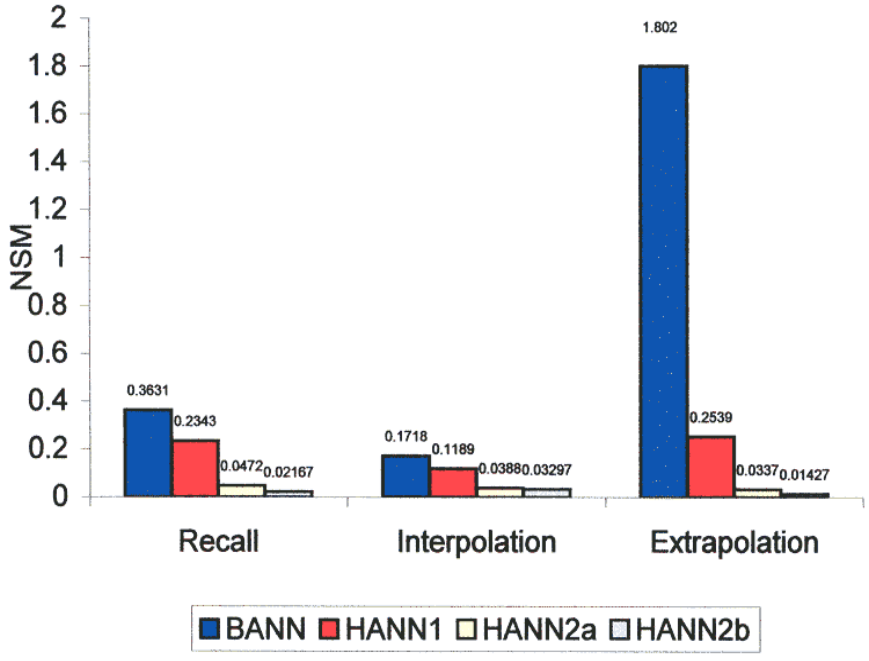

Figure 4.20 Average NSM for cellobiose (shell) concentration predictions

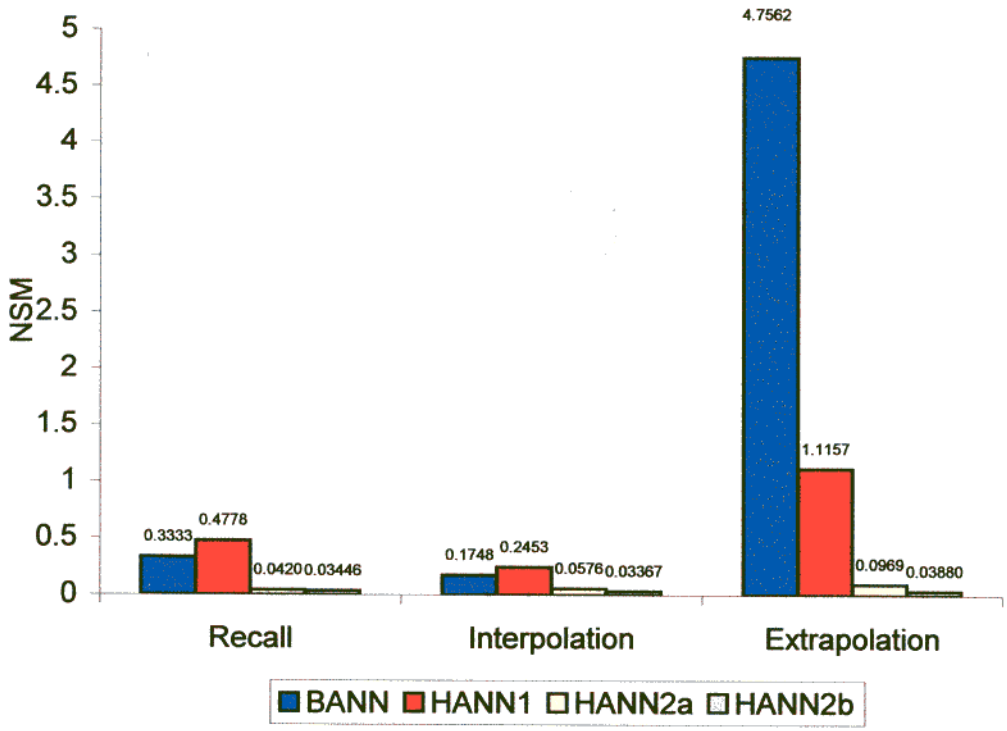

Figure 4.21 Average NSM for glucose (shell) concentration predictions

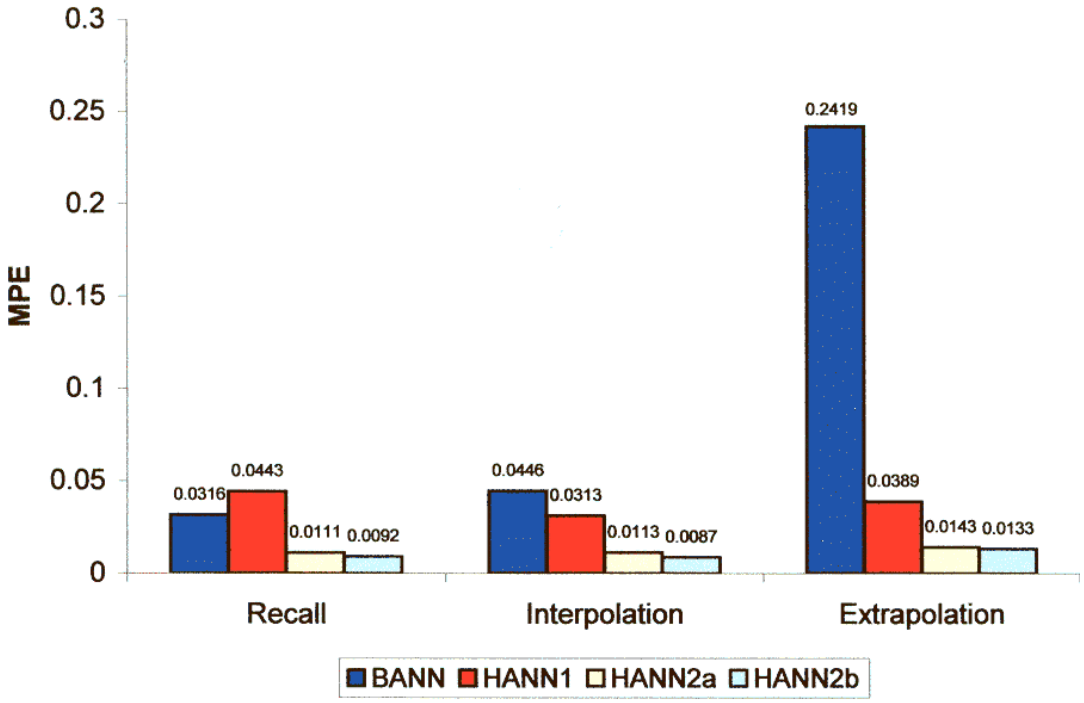

Figure 4.22 Average MPE for cellulose concentration predictions

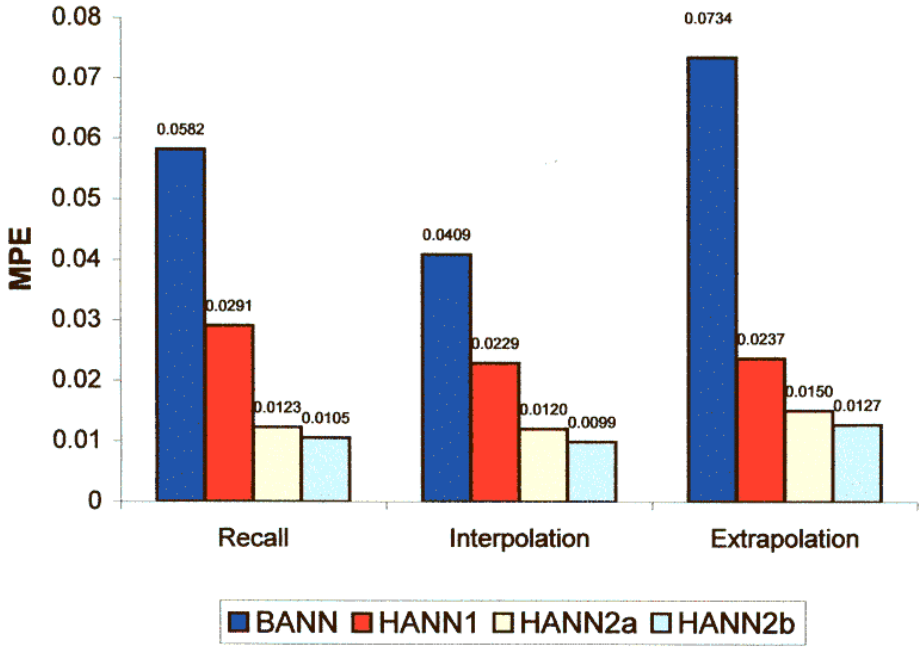

Figure 4.23 Average MPE for cellobiose (tube) concentration predictions

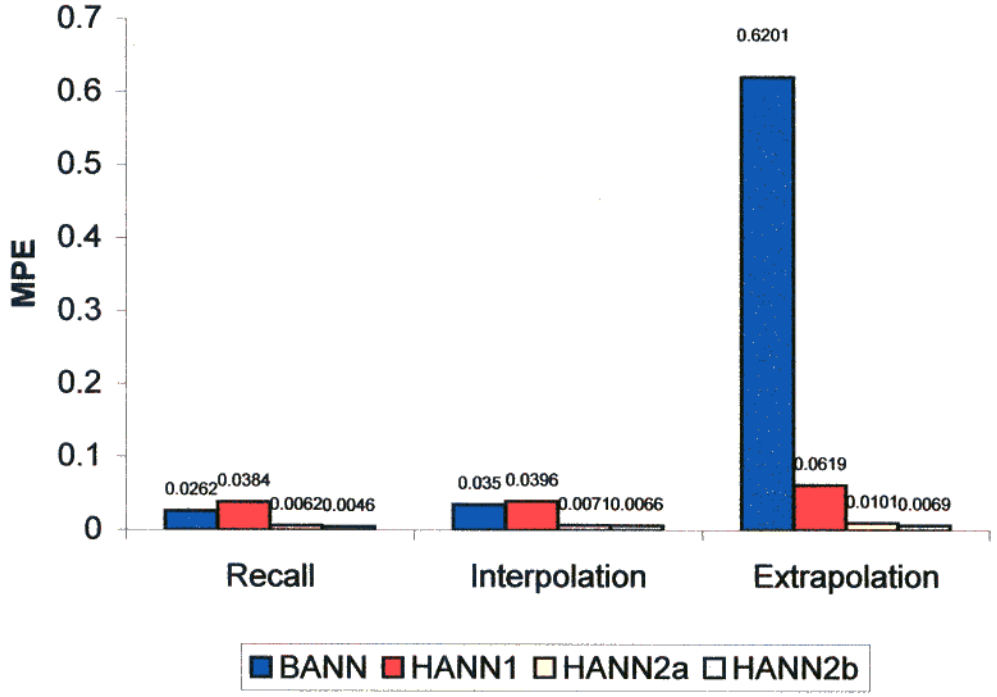

Figure 4.24 Average MPE for glucose (tube) concentration predictions

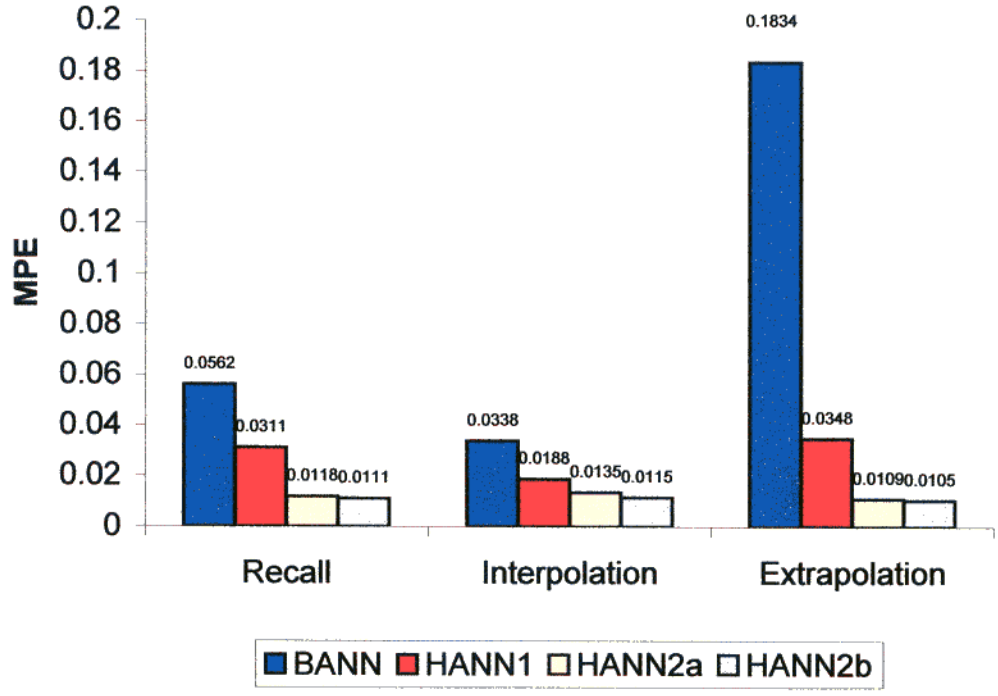

Figure 4.25 Average MPE for cellobiose (shell) concentration predictions

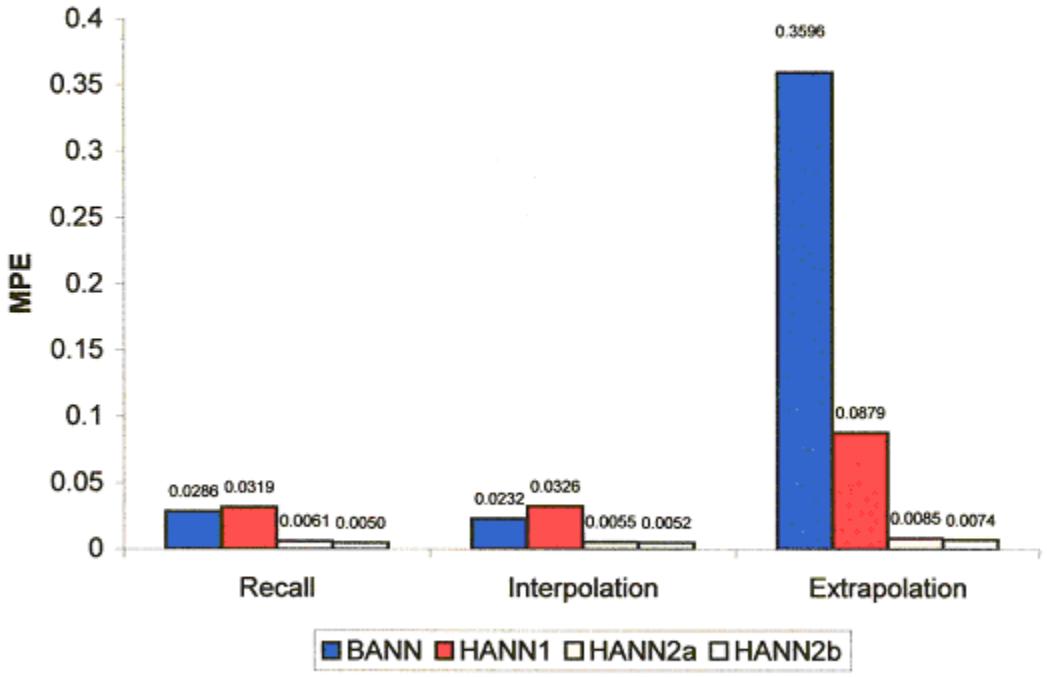

Figure 4.26 Average MPE for glucose (shell) concentration predictions

5. Conclusions

Three modeling schemes were developed to model a steady-state space-dependent enzymatic tubular membrane reactor (TMR). At first, a fully black-box model (BANN), based on ANN technique, was developed using only the process data. No information about the process was included in this model. Then, first-principle information of mass balances equations (ODEs) was introduced separately into the black-box model to generate the first hybrid model (HANN1). After that, a new hybrid scheme, combining ANN with mass balances and assumed rate expressions, was used to develop the second hybrid model (HANN2) using smoothed and non-smoothed data. The second hybrid scheme, developed for a space-dependent steady-state enzymatic reactor, is similar to that developed by Kasprow for a fed-batch microbial reactor that is space-independent and time-dependent.

Qualitative and quantitative comparisons of the predicted profiles of the three modeling schemes (BANN, HANN1, and HANN2) indicated that the second hybrid scheme (HANN2) performed better than the other two schemes (BANN and HANN1). Because the inclusion of engineering first principles and basic biochemical knowledge, in the form of mass balances and the simplified rate expressions, have allowed the HANN2 scheme to perform consistently well on all testing regimes (recall, interpolation, and extrapolation). It is also worthwhile to note that HANN1 model significantly outperforms the BANN model in the extrapolation cases, while the differences in outcomes from HANN2a and HANN2b are not significant.

References

1. Azevedo, S. F., Dahm, B., and Oliveira, F. R., "Hybrid Modeling of Biochemical Process: A comparison With Conventional Approach," Computers and Chemical Engineering, **21**, S751 (1997).
2. Bahat, N. and McAvoy, T., "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process System," Computers and Chemical Engineering, **14**, 573 (1990).
3. Baughman, D. R. and Liu, Y. A., Neural Networks in Bioprocessing and Chemical Engineering, Academic Press, 1995.
4. Boor, C., Spline Toolbox User's Guide, The MathWorks Inc., 2001.
5. Borse, G. J.,Numerical Methods with MATLAB, PWS Publishing Company, 1995.
6. Chitra, S. P., "Use of Neural Networks for Problem Solving" Chemical Engineering Progress, April, 44 (1993).
7. Costa, A. C., Alves, T. L. M, Henriques, A. W. S., Filho, R. M., and Lima, E. L., "A Hybrid Neural Model for the Optimization of Fed Batch Fermentations," Brazilian Journal of Chemical Engineering, **16**, 53 (1999).
8. Demuth, H. and Beale, M., Neural Networks Toolbox User's Guide, The MathWorks Inc., 2001.
9. Foresee, F. D., "Generalization and Neural Networks", Ph.D. Dissertation, Oklahoma Stat University,1996.

10. Foresee, F. D., and Hagan, M.T., "Guass-Newton approximation to Bayesian regulation", Proceedings of the 1997 International Joint Conference on Neural Networks, pages 1930-1935,1997.
11. Gauba, G., " Enzymatic Saccharification of Cellulose in Membrane Bioreactor," M.S. Thesis, WVU, (Research advisor: R. Y. K. Yang), 1993.
12. Harada, H.P., Costa, A. C., and Filho, R. M., "Hybrid Neural Modeling of Bioprocesses Using Function Link Networks," Applied Biochemistry and Biotechnology, **98**, 1009 (2002).
13. Hagan, M. T. and Demuth, H. B., Neural Networks Design, PWS, 1996.
14. Henrigue, H. and Lima, E. L., " Model Structure determination in Neural Network Models," Chemical Engineering Scince, **55**, 5457 (2000).
15. Hornik, K., Stinchcomb, M., and White, H., " Multilayer Feedforward Networks are Universal Approximations," Neural Networks, **2**, 359, (1989).
16. Hassoun, M.H., " Fundamentals of Artificial Neural Networks" Cambridge, MIT Press, 1995.
17. Kasprow, R. K, " Hybrid Modeling (Neural Networks and First Principles) of Fermentation: Combining Biochemical Engineering Fundamentals and Process data" PhD. Dissertation, University of Virginia, (Research advisor: D. Kirwan), 2000.
18. Layton, C., " Enzymatic Hydrolysis of Cellulose in Hollow-Fiber Membrane Reactor," M.S. Thesis, WVU, (Research advisor: R. Y. K. Yang), 1991.
19. Psichogios, D. C. and Ungar, L. H., " A Hybrid Neural Network First Principles Approach to Process Modeling," AIChE Journal, **38**, 1499(1992).

20. Polking, J. P. and Arnold, D., Ordinary Differential Equation Using Matlab, 2nd Ed. , Prentice Hall , 1999.

21. Qi, H., Zhou, X., Liu, L. and Yuan, W., "A Hybrid Neural Networks First Principles for Fixed bed Reactor," Chemical Engineering Science, **54**, 2521, (1999).

22. Thompson, M. L., and Kramer, M. R. " Modeling Chemical Process Using Prior Knowledge and Neural Networks," AIChE Journal, **40**, 1328 (1994).

23. Tholudur, A., and Ramirez, W. F. "Optimization of Fed-Batch Bioreactor using Neural Network Parameter Function Models", Biotechnology Progress, **38**, 302, (1996).

24. Wilson, J. A., and Zorzetto, L. F., " A Generalized Approach to Process State Estimation Using Hybrid Artificial Neural Networks/Mechanistic Models," Computers and Chemical Engineering, **21**, 951 (1997).

25. Zorzetto, L. F., Fiho, R. M., and Wolf, M. R., " Process Modeling Development Through Artificial Neural Networks and Hybrid Models" Computers and Chemical Engineering, **24**, 1355 (2000).

Appendix A-Programs output

A1 Sample of output generated from Fortran program developed by G. Gauba [11] to solve the initial-value type ordinary differential equations (ODEs) using Livermore solver for ODE (LSODE).

A2 Sample of output generated from Matlab function "**tmr.m**" to solve the initial-value type ordinary differential equations (ODEs) using Matlab solver "**ode23S**".

A1

C_{so}= 0.0025 g/l, V_f= 0.6 ml/min, L= 200 cm

Z (cm)	C_s (g/l)	C_B (g/l)
1.440997973597329e+002	9.204938365251831e-004	2.357510673193909e-003
1.450997973344707e+002	9.122489493962936e-004	2.365588556109517e-003
1.460997973092086e+002	9.040018343372537e-004	2.373586021969356e-003
1.470997972839465e+002	8.957522946650913e-004	2.381502537185116e-003
1.480997972586844e+002	8.875001430634884e-004	2.389337540244296e-003
1.490997972334222e+002	8.792452041309137e-004	2.397090436306075e-003
1.500997972081601e+002	8.709873183088079e-004	2.404760588957872e-003
1.510997971828980e+002	8.627263248866492e-004	2.412347355019059e-003
1.520997971576359e+002	8.544620775941547e-004	2.419850052300013e-003
1.530997971323737e+002	8.461944402158041e-004	2.427267968463170e-003
1.540997971071116e+002	8.379232866509205e-004	2.434600360737810e-003
1.550997970818495e+002	8.296485107204836e-004	2.441846434464746e-003
1.560997970565874e+002	8.213700068241208e-004	2.449005384954997e-003
1.570997970313252e+002	8.130876816930559e-004	2.456076371807432e-003
1.580997970060631e+002	8.048014541245818e-004	2.463058519330781e-003
1.590997969808010e+002	7.965112539895258e-004	2.469950918544718e-003
1.600997969555389e+002	7.882170271821136e-004	2.476752616000304e-003
1.610997969302767e+002	7.799187338213712e-004	2.483462617639613e-003
1.620997969050146e+002	7.716163368493236e-004	2.490079914056462e-003
1.630997968797525e+002	7.633098155222240e-004	2.496603450282101e-003
1.640997968544904e+002	7.549991614260206e-004	2.503032134516285e-003
1.650997968292282e+002	7.466843789018208e-004	2.509364837017601e-003
1.660997968039661e+002	7.383654998002783e-004	2.515600356359604e-003
1.670997967787040e+002	7.300425502298069e-004	2.521737495021252e-003
1.680997967534419e+002	7.217155770186947e-004	2.527774998978641e-003
1.690997967281797e+002	7.133846409351382e-004	2.533711573005726e-003
1.700997967029176e+002	7.050498169114886e-004	2.539545880022936e-003
1.710997966776555e+002	6.967112036017983e-004	2.545276518544461e-003
1.720997966523934e+002	6.883689116275672e-004	2.550902050135332e-003
1.730997966271312e+002	6.800230601916410e-004	2.556421007231990e-003
1.740997966018691e+002	6.716737885828502e-004	2.561831866020575e-003
1.750997965766070e+002	6.633212523339036e-004	2.567133055327538e-003
1.760997965513449e+002	6.549656263985318e-004	2.572322948809393e-003
1.770997965260827e+002	6.466071168821344e-004	2.577399836389321e-003
1.780997965008206e+002	6.382459264189027e-004	2.582362008160128e-003
1.790997964755585e+002	6.298822843366952e-004	2.587207681094362e-003
1.800997964502964e+002	6.215164387574542e-004	2.591935018109236e-003
1.810997964250342e+002	6.131486569750993e-004	2.596542127048122e-003
1.820997963997721e+002	6.047792429735273e-004	2.601027016677437e-003
1.830997963745100e+002	5.964085070227625e-004	2.605387672863769e-003
1.840997963492479e+002	5.880367791189517e-004	2.609622024797741e-003
1.850997963239857e+002	5.796644146670154e-004	2.613727930672027e-003
1.860997962987236e+002	5.712917914552784e-004	2.617703185197685e-003
1.870997962734615e+002	5.629193156430315e-004	2.621545504015519e-003
1.880997962481994e+002	5.545474267889907e-004	2.625252510600056e-003
1.890997962229372e+002	5.461765738942988e-004	2.628821798493900e-003
1.900997961976751e+002	5.378072382346145e-004	2.632250871941818e-003
1.910997961724130e+002	5.294399278179517e-004	2.635537160266717e-003
1.920997961471509e+002	5.210751905220844e-004	2.638677981811962e-003
1.930997961218887e+002	5.127135900285188e-004	2.641670610195530e-003
1.940997960966266e+002	5.043557235603400e-004	2.644512225600809e-003
1.950997960713645e+002	4.960022192752868e-004	2.647199922016836e-003
1.960997960461024e+002	4.876537422304381e-004	2.649730689531475e-003
1.970997960208402e+002	4.793109906861528e-004	2.652101425669553e-003
1.980997959955781e+002	4.709746908936624e-004	2.654308951275732e-003
1.990997959703160e+002	4.626456065467428e-004	2.656349982446680e-003

A2

C_{so}= 0.0025 g/l, V_f= 0.6 ml/min, L= 200 cm

Z (cm)	C_s (g/l)	C_B (g/l)
1.440997000000000e+002	9.206133393034483e-004	2.357028029533870e-003
1.450997000000000e+002	9.123655622921875e-004	2.365109414402626e-003
1.460997000000000e+002	9.041156053036204e-004	2.373110289821769e-003
1.470997000000000e+002	8.958632684994322e-004	2.381030128764720e-003
1.480997000000000e+002	8.876083605569493e-004	2.388868378888446e-003
1.490997000000000e+002	8.793506986691386e-004	2.396624462533466e-003
1.500997000000000e+002	8.710901085446079e-004	2.404297776723846e-003
1.510997000000000e+002	8.628264244076063e-004	2.411887693167202e-003
1.520997000000000e+002	8.545594903611551e-004	2.419393553913130e-003
1.530997000000000e+002	8.462891768843841e-004	2.426814623843403e-003
1.540997000000000e+002	8.380153650458658e-004	2.434150145634370e-003
1.550997000000000e+002	8.297379449551726e-004	2.441399335903717e-003
1.560997000000000e+002	8.214568182828716e-004	2.448561376181516e-003
1.570997000000000e+002	8.131718982605244e-004	2.455635412910230e-003
1.580997000000000e+002	8.048831096806872e-004	2.462620557444711e-003
1.590997000000000e+002	7.965903888969110e-004	2.469515886052200e-003
1.600997000000000e+002	7.882936838237414e-004	2.476320439912326e-003
1.610997000000000e+002	7.799929539367183e-004	2.483033225117107e-003
1.620997000000000e+002	7.716881702723768e-004	2.489653212670951e-003
1.630997000000000e+002	7.633793154282460e-004	2.496179338490653e-003
1.640997000000000e+002	7.550663835628502e-004	2.502610503405400e-003
1.650997000000000e+002	7.467493803957078e-004	2.508945573156765e-003
1.660997000000000e+002	7.384283232073321e-004	2.515183378398710e-003
1.670997000000000e+002	7.301032408392310e-004	2.521322714697589e-003
1.680997000000000e+002	7.217741736939070e-004	2.527362342532141e-003
1.690997000000000e+002	7.134411737348573e-004	2.533300987293496e-003
1.700997000000000e+002	7.051043044865736e-004	2.539137339285173e-003
1.710997000000000e+002	6.967636410345423e-004	2.544870053723080e-003
1.720997000000000e+002	6.884192742111839e-004	2.550497738133260e-003
1.730997000000000e+002	6.800713398740353e-004	2.556018873673755e-003
1.740997000000000e+002	6.717199847361660e-004	2.561431924783719e-003
1.750997000000000e+002	6.633653713434626e-004	2.566735311088073e-003
1.760997000000000e+002	6.550076817271672e-004	2.571927395297550e-003
1.770997000000000e+002	6.466471174038767e-004	2.577006483208690e-003
1.780997000000000e+002	6.382838993755434e-004	2.581970823703847e-003
1.790997000000000e+002	6.299182681294742e-004	2.586818608751186e-003
1.800997000000000e+002	6.215504836383314e-004	2.591547973404682e-003
1.810997000000000e+002	6.131808253601325e-004	2.596156995804122e-003
1.820997000000000e+002	6.048095922382500e-004	2.600643697175102e-003
1.830997000000000e+002	5.964371027014111e-004	2.605006041829032e-003
1.840997000000000e+002	5.880636946636986e-004	2.609241937163131e-003
1.850997000000000e+002	5.796897255245504e-004	2.613349233660430e-003
1.860997000000000e+002	5.713155721687589e-004	2.617325724889770e-003
1.870997000000000e+002	5.629416309664723e-004	2.621169147505805e-003
1.880997000000000e+002	5.545683177731934e-004	2.624877181248997e-003
1.890997000000000e+002	5.461960679297804e-004	2.628447448945622e-003
1.900997000000000e+002	5.378253362624463e-004	2.631877516507765e-003
1.910997000000000e+002	5.294565970827594e-004	2.635164892933323e-003
1.920997000000000e+002	5.210903683506394e-004	2.638306973673227e-003
1.930997000000000e+002	5.127272932708564e-004	2.641300847866090e-003
1.940997000000000e+002	5.043679809772931e-004	2.644143669076694e-003
1.950997000000000e+002	4.960130624948722e-004	2.646832524366862e-003
1.960997000000000e+002	4.876632015729075e-004	2.649364408953359e-003
1.970997000000000e+002	4.793190946851038e-004	2.651736226207884e-003
1.980997000000000e+002	4.709814710295566e-004	2.653944787657077e-003
1.990997000000000e+002	4.626510925287525e-004	2.655986812982515e-003

Appendix B-Sample programs

B1 "**tmr.m**" : A MATLAB version of deterministic model TMR program.

B2 "**noise.m**": A MATLAB function to generate random noise.

B3 "**bann_data.m**": A MATLAB script file to generate training and testing data for BANN model.

B4 "**hann1_data.m**": A MATLAB script file to generate training and testing data for HANN1 model.

B5 "**hann1.m**": A MATLAB function for the HANN1 model.

B6 "**hann2a_data.m**": A MATLAB script file to generate training and testing data for HANN2b model.

B7 "**hann2b_data.m**": A MATLAB script file to generate training and testing data for HANN2b model.

B8 "**hann2.m**": A MATLAB function for the HANN2 model.

B1- tmr.m

```
function [cdot]=tmr(z,c,flag,f);
% This functions contains  5 ODEs which describes enzymatic
% Saccharification of Cellulose in Hollow Fiber Bioreactor.
% This function will be called by a selected ODE solver to
% calculate the concentration profiles of Cellulose,
% Cellobiose and Gulocose in the bioreactor.
% Last update 6/11/03
%Operating Parameters
pp=1033.82;        % pressure on the shell side, g/cm.cm
pf=1100;           % pressure at the entrance of the modul(tube side),
                   % g/cm.cm
pr=1070;           % pressure at the exit of the module,g/cm.cm
l =200;            % tube length, cm
r1=0.3;            % tube radius, cm
vf=f;              % volumetric flow rate at the entrance of the
                   % reactor, cc/min
lp=2.5e-5;         % hydraulic permeability, cc/[(cm.cm).min.(g/cm.cm)]
rm=1.39e-3;        % maximum reaction rate (cellobiose), g/(cc.min)
rprimem=1.22e-3;   % maximum reaction rate (glucose)g/(cc.min)
km=42.18e-3;       % Michaelis-Menten constant (cellobiose), g/cc
kprimem=198.34e-3; % product inhibition constant (cellobiose), g/cc
ki=1.89e-3;        % product inhibition constant (cellobiose), g/cc
kprimei=0.66e-3;   % product inhibition constant (glucose), g/cc

pt=(pf-pp)+(pr-pf)*z/l;  % transmembrane pressure drop

% v volumetric flow rates at a distance z from the entrance of the
% reactor on the tube side
 v=-(2.0*pi*r1*lp)*((pf-pp)*z+(pr-pf)*z^2/l/2.0)+vf;

% vbar volumetric flow rates at a distance z from the entrance of the
% reactor on the shell side
     vbar= vf-v+1.0e-15;
     dvdz=-(2.0d0*pi*r1*lp)*((pf-pp)+(pr-pf)*z/l);

    % rb are the rate of formation of cellobiose
    rb=rm*c(1)/(km+c(1)+(km*c(2)/ki));

    % rg are the rate of formation of glucose
      rg=rprimem*c(2)/(kprimem+c(2)+(kprimem*c(3)/kprimei));

  % 5 ODEs to calculate the concentration profiles of Cellulose
  % c(1),Cellobiose c(2) and Glucose c(3) on the tube side.
  % Cellobiose c(4) and Glucose c(5) on the shell side.
      cdot=[(pi*r1^2/v)*(-rb-c(1)*dvdz/pi/r1^2);(pi*r1^2/v)*(-rg+rb-...
      (c(2)*dvdz/pi/r1^2)-(2.0d0*lp*pt/r1)*c(2));(pi*r1^2/v)*...
      (rg-(c(3)*dvdz/pi/r1^2)-(2.0d0*lp*pt/r1)*c(3));(pi*r1^2/vbar)*...
      ((c(4)*dvdz/pi/r1^2)+(2.0*lp*pt/r1)*c(2));(pi*r1^2/vbar)...
      *((c(5)*dvdz/pi/r1^2)+(2.0*lp*pt/r1)*c(3))];
```

B2- noise.m

```
function[Cn]=noise(C,per)
% This function generate noise and add  it to
% the mathematical model prediction "C"in order
% to generat process data "Cn"
% Noise value "Vnoise" is determined at each point by
% sampling from a normal distribution having zero mean and
% a standard deviation equal to "per" of the state variable value
% Last update 4/1/03

Vnoise=ones(size(C))+[per/100*randn(length(C),5)];
Cn=C.*Vnoise;
```

B3- bann_data.m

```
%This script file "bann_data" is used to prepare training,
%interpolation and extrapolation cases for BANN model
% Last update 15/6/03
% In this section the function of mass balance equations "tmr" is
% called ode23s solver
%for different initial conditions in order to generate training,
interpolation and extrapolation
[z,c1]=ode23s('tmr',[0:10:200],[0.0025;0;0;0;0],[],0.6);
[z,c2]=ode23s('tmr',[0:10:200],[0.0016;0;0;0;0],[],0.62);
[z,c3]=ode23s('tmr',[0:10:200],[0.0018;0;0;0;0],[],0.66);
[z,c4]=ode23s('tmr',[0:10:200],[0.0013;0;0;0;0],[],0.67);
[z,c5]=ode23s('tmr',[0:10:200],[0.0009;0;0;0;0],[],0.70);
[z,c6]=ode23s('tmr',[0:10:200],[0.0014;0;0;0;0],[],0.71);
[z,c7]=ode23s('tmr',[0:10:200],[0.0017;0;0;0;0],[],0.75);
[z,c8]=ode23s('tmr',[0:10:200],[0.0005;0;0;0;0],[],0.79);
[z,c9]=ode23s('tmr',[0:10:200],[0.0019;0;0;0;0],[],0.82);
[z,c10]=ode23s('tmr',[0:10:200],[0.0024;0;0;0;0],[],0.83);
[z,c11]=ode23s('tmr',[0:10:200],[0.0007;0;0;0;0],[],0.86);
[z,c12]=ode23s('tmr',[0:10:200],[0.0009;0;0;0;0],[],0.89);
[z,c13]=ode23s('tmr',[0:10:200],[0.0026;0;0;0;0],[],0.91);
[z,c14]=ode23s('tmr',[0:10:200],[0.0011;0;0;0;0],[],0.93);
[z,c15]=ode23s('tmr',[0:10:200],[0.001;0;0;0;0],[],0.95);
[z,c16]=ode23s('tmr',[0:10:200],[0.0006;0;0;0;0],[],0.97);
[z,c17]=ode23s('tmr',[0:10:200],[0.0021;0;0;0;0],[],0.99);
[z,c18]=ode23s('tmr',[0:10:200],[0.0004;0;0;0;0],[],1.01);
[z,c19]=ode23s('tmr',[0:10:200],[0.0021;0;0;0;0],[],1.03);
[z,c20]=ode23s('tmr',[0:10:200],[0.0018;0;0;0;0],[],1.05);
% interpolation cases (21-23)
[z,c21]=ode45('tmr',[0:10:200],[0.001;0;0;0;0],[],0.6);
[z,c22]=ode45('tmr',[0:10:200],[0.0015;0;0;0;0],[],0.73);
[z,c23]=ode45('tmr',[0:10:200],[0.0005;0;0;0;0],[],1.0);
% extrapolation cases (24-24)
[z,c24]=ode45('tmr',[0:10:200],[0.0045;0;0;0;0],[],0.65);
[z,c25]=ode45('tmr',[0:10:200],[0.0003;0;0;0;0],[],1.20);
[z,c26]=ode45('tmr',[0:10:200],[0.0035;0;0;0;0],[],1.30);
% "c" Matrix for whole cases
```

```
c=[c1;c2;c3;c4;c5;c6;c7;c8;c9;c10;c11;c12;c13;c14;c15;c16;c17;c18;c19;c
20;c21;c22;c23;c24;c25;c26];
% Generate noise (3%) for all cases (1-26) "cs" using noise function
cs=noise(c,3);
x=0:10:200;
% Appling smoothing spline using spaps for cellulose (tube) data points
for i=1:21:546
  y1=cs(i:i+20,1);
  ys1=spaps(x,y1,1e-7);
  ps1=fnval(ys1,x);
p1(i:i+20)=ps1;
end
% Appling smoothing spline using spaps for cellobiose (tube) data
% points
for i=1:21:546
   y2=cs(i:i+20,2);
   ys2=spaps(x,y2,1e-7);
   ps2=fnval(ys2,x);
   p2(i:i+20)=ps2;
end
% Appling smoothing spline using spaps for glucose (tube) data points
for i=1:21:546
   y3=cs(i:i+20,3);
   ys3=spaps(x,y3,1e-7);
   ps3=fnval(ys3,x);
   p3(i:i+20)=ps3;
end
% Appling smoothing spline using spaps for cellobiose (shell) data
% points
for i=1:21:546
   y4=cs(i:i+20,4);
   ys4=spaps(x,y4,1e-7);
   ps4=fnval(ys4,x);
   p4(i:i+20)=ps4;
end
% Appling smoothing spline using spaps for glucose (shell) data points
for i=1:21:546
   y5=cs(i:i+20,5);
   ys5=spaps(x,y5,1e-7);
   ps5=fnval(ys5,x);
   p5(i:i+20)=ps5;
end
p1=p1';p2=p2';p3=p3';p4=p4';p5=p5';
csm=[p1 p2 p3 p4 p5];  % smoothed process data
% Normalize training data between (-1 and 1)using premnmx function

[pn1,minp1,maxp1]=premnmx(p1);[pn2,minp2,maxp2]=premnmx(p2);[pn3,minp3,
maxp3]=premnmx(p3);
 [pn4,minp4,maxp4]=premnmx(p4);[pn5,minp5,maxp5]=premnmx(p5);
 % Normalize flowrates "vf" between (-1 and 1)using premnmx function
vf=[0.6;0.62;0.66;0.67;0.70;0.71;0.75;0.79;0.82;0.83;0.86;0.89;0.91;...
   0.95;0.97;0.99;1.01;1.03;1.05;0.6;0.73;1.0;0.65;1.2;1.3];
[vfn,minvf,maxvf]=premnmx(vf);
% prepare feed flowrate "vf" as a fector for each cases
vfn1=ones(21,1)*vfn(1);vfn2=ones(21,1)*vfn(2);vfn3=ones(21,1)*vfn(3);vf
n4=ones(21,1)*vfn(4);vfn5=ones(21,1)*vfn(5);vfn6=ones(21,1)*vfn(6);vfn7
=ones(21,1)*vfn(7);vfn8=ones(21,1)*vfn(8);vfn9=ones(21,1)*vfn(9);vfn10=
```

```
ones(21,1)*vfn(10);vfn11=ones(21,1)*vfn(11);vfn12=ones(21,1)*vfn(12);vf
n13=ones(21,1)*vfn(13);vfn14=ones(21,1)*vfn(14);vfn15=ones(21,1)*vfn(15
);vfn16=ones(21,1)*vfn(16);vfn17=ones(21,1)*vfn(17);vfn18=ones(21,1)*vf
n(18);vfn19=ones(21,1)*vfn(19);vfn20=ones(21,1)*vfn(20);vfn21=ones(21,1
)*vfn(21);vfn22=ones(21,1)*vfn(22);vfn23=ones(21,1)*vfn(23);vfn24=ones(
21,1)*vfn(24);vfn25=ones(21,1)*vfn(25);vfn26=ones(21,1)*vfn(26);
vfnt=[vfn1;vfn2;vfn3;vfn4;vfn5;vfn6;vfn7;vfn8;vfn9;vfn10;vfn11;vfn12;..
.vfn13;vfn14;vfn15;vfn16;vfn17;vfn18;vfn19;vfn20;vfn21;vfn22;vfn23;vfn2
4;vfn25;vfn26];
%Prepare all cases as "input" and  "target" vectors for all cases
%(training, interpolation and extrapolation)
input=[pn1 pn2 pn3 pn4 pn5 vfnt];
target=[pn1 pn2 pn3 pn4 pn5];
% each case consist of 20 input vectors and 20 traget vectors
% Inputs "Ptrb" and targets "Ttrb" for training cases (1-20)
input1=input(1:20,:);target1=target(2:21,:);
input2=input(22:41,:);target2=target(23:42,:);
input3=input(43:62,:);target3=target(44:63,:);
input4=input(64:83,:);target4=target(65:84,:);
input5=input(85:104,:);target5=target(86:105,:);
input6=input(106:125,:);target6=target(107:126,:);
input7=input(127:146,:);target7=target(128:147,:);
input8=input(148:167,:);target8=target(149:168,:);
input9=input(169:188,:);target9=target(170:189,:);
input10=input(190:209,:);target10=target(191:210,:);
input11=input(211:230,:);target11=target(212:231,:);
input12=input(232:251,:);target12=target(233:252,:);
input13=input(253:272,:);target13=target(254:273,:);
input14=input(274:293,:);target14=target(275:294,:);
input15=input(295:314,:);target15=target(296:315,:);
input16=input(316:335,:);target16=target(317:336,:);
input17=input(337:356,:);target17=target(338:357,:);
input18=input(358:377,:);target18=target(359:378,:);
input19=input(379:398,:);target19=target(380:399,:);
input20=input(400:419,:);target20=target(401:420,:);
Ptr_bn=[input1;input2;input3;input4;input5;input6;input7;input8;input9;
input10;input11;input12;...
  input13;input14;input15;input16;input17;input18;input19;input20]';
Ttr_bn=[target1;target2;target3;target4;target5;target6;target7;target8
;target9;target10;target11;...

target12;target13;target14;target15;target16;target17;target18;target19
;target20]';
%Inputs "Pin_bn" and targets "Tin_bn" for interpolation cases (21-23)
input21=input(421:440,:);target21=target(422:441,:);
input22=input(442:461,:);target22=target(443:462,:);
input23=input(463:482,:);target23=target(464:483,:);
Pin_bn=[input21;input22;input23]';
Tin_bn=[target21;target22;target23]';
%Inputs "Pex_bn" and targets "Tex_bn" for extrapolation cases (24-26)
input24=input(484:503,:);target24=target(485:504,:);
input25=input(505:524,:);target25=target(506:525,:);
input26=input(526:545,:);target26=target(527:546,:);
Pex_bn=[input24;input25;input26]';
Tex_bn=[target24;target25;target26]';
```

B4- hann1_data.m

```
%This script file "hann_data" is used to prepare training,
%interpolation and extrapolation cases for HANN-1 model
% Last update 6/11/03
% In this section the function of mass balance equations "tmr" is
% called by
% ode23s solver for different initial conditions in order to generate
training (1-20),
%interpolation (21-23) and extrapolation (24-26)
[z,c1]=ode23s('tmr',[0:10:200],[0.0025;0;0;0;0],[],0.6);
[z,c2]=ode23s('tmr',[0:10:200],[0.0016;0;0;0;0],[],0.62);
[z,c3]=ode23s('tmr',[0:10:200],[0.0018;0;0;0;0],[],0.66);
[z,c4]=ode23s('tmr',[0:10:200],[0.0013;0;0;0;0],[],0.67);
[z,c5]=ode23s('tmr',[0:10:200],[0.0009;0;0;0;0],[],0.70);
[z,c6]=ode23s('tmr',[0:10:200],[0.0014;0;0;0;0],[],0.71);
[z,c7]=ode23s('tmr',[0:10:200],[0.0017;0;0;0;0],[],0.75);
[z,c8]=ode23s('tmr',[0:10:200],[0.0005;0;0;0;0],[],0.79);
[z,c9]=ode23s('tmr',[0:10:200],[0.0019;0;0;0;0],[],0.82);
[z,c10]=ode23s('tmr',[0:10:200],[0.0024;0;0;0;0],[],0.83);
[z,c11]=ode23s('tmr',[0:10:200],[0.0007;0;0;0;0],[],0.86);
[z,c12]=ode23s('tmr',[0:10:200],[0.0009;0;0;0;0],[],0.89);
[z,c13]=ode23s('tmr',[0:10:200],[0.0026;0;0;0;0],[],0.91);
[z,c14]=ode23s('tmr',[0:10:200],[0.0011;0;0;0;0],[],0.93);
[z,c15]=ode23s('tmr',[0:10:200],[0.001;0;0;0;0],[],0.95);
[z,c16]=ode23s('tmr',[0:10:200],[0.0006;0;0;0;0],[],0.97);
[z,c17]=ode23s('tmr',[0:10:200],[0.0021;0;0;0;0],[],0.99);
[z,c18]=ode23s('tmr',[0:10:200],[0.0004;0;0;0;0],[],1.01);
[z,c19]=ode23s('tmr',[0:10:200],[0.0021;0;0;0;0],[],1.03);
[z,c20]=ode23s('tmr',[0:10:200],[0.0018;0;0;0;0],[],1.05);
% interpolation cases (21-23)
[z,c21]=ode45('tmr',[0:10:200],[0.001;0;0;0;0],[],0.6);
[z,c22]=ode45('tmr',[0:10:200],[0.0015;0;0;0;0],[],0.73);
[z,c23]=ode45('tmr',[0:10:200],[0.0005;0;0;0;0],[],1.0);
% extrapolation cases (24-24)
[z,c24]=ode45('tmr',[0:10:200],[0.0045;0;0;0;0],[],0.65);
[z,c25]=ode45('tmr',[0:10:200],[0.0003;0;0;0;0],[],1.20);
[z,c26]=ode45('tmr',[0:10:200],[0.0035;0;0;0;0],[],1.30);
% "c" Matrix for whole cases
c=[c1;c2;c3;c4;c5;c6;c7;c8;c9;c10;c11;c12;c13;c14;c15;c16;c17;c18;c19;c
20;c21;c22;c23;c24;c25;c26];
% Generate noise (3%) for all cases (1-26) "cs" using noise function
cs=noise(c,3);
x=0:10:200;
% Appling smoothing spline using spaps function for cellulose (tube)
% data points
for i=1:21:546
  y1=cs(i:i+20,1);
  ys1=spaps(x,y1,1e-7);
  ps1=fnval(ys1,x);
  p1(i:i+20)=ps1;
% Calculat first derivative
  d1=fnval(fnder(ys1),x);
  dc1(i:i+20)=d1;
end
% Appling smoothing spline using spaps for cellobiose (tube) data
```

```
% points
for i=1:21:546
   y2=cs(i:i+20,2);
   ys2=spaps(x,y2,1e-7);
   ps2=fnval(ys2,x);
   p2(i:i+20)=ps2;
 % Calculat first derivative
   d2=fnval(fnder(ys2),x);
   dc2(i:i+20)=d2;
end
% Appling smoothing spline using spaps for glucose (tube) data points
for i=1:21:546
   y3=cs(i:i+20,3);
   ys3=spaps(x,y3,1e-7);
   ps3=fnval(ys3,x);
    p3(i:i+20)=ps3;
  % Calculat first derivative
   d3=fnval(fnder(ys3),x);
   dc3(i:i+20)=d3;
end
% Appling smoothing spline using spaps for cellobiose (shell) data
% points
for i=1:21:546
   y4=cs(i:i+20,4);
   ys4=spaps(x,y4,1e-7);
   ps4=fnval(ys4,x);
   p4(i:i+20)=ps4;
 % Calculat first derivative
   d4=fnval(fnder(ys4),x);
   dc4(i:i+20)=d4;
end
% Appling smoothing spline using spaps for glucose (shell) data points
for i=1:21:546
   y5=cs(i:i+20,5);
   ys5=spaps(x,y5,1e-9);
   ps5=fnval(ys5,x);
    if ps5 < 0
       ps5=1e-6;
   end
   p5(i:i+20)=ps5;
% Calculat first derivative
   d5=fnval(fnder(ys5),x);
   dc5(i:i+20)=d5;
end
% smoothed process data "csm" (cases 1-26)
p1=p1';p2=p2';p3=p3';p4=p4';p5=p5';
csm=[p1 p2 p3 p4 p5];
% Normalize smoothed process data between (-1 and 1)using premnmx
function

[pn1,minp1,maxp1]=premnmx(p1);[pn2,minp2,maxp2]=premnmx(p2);[pn3,minp3,
maxp3]=premnmx(p3);
 [pn4,minp4,maxp4]=premnmx(p4);[pn5,minp5,maxp5]=premnmx(p5);
% Normalize flowrates "vf" between (-1 and 1)using premnmx function
vf=[0.6;0.62;0.66;0.67;0.70;0.71;0.75;0.79;0.82;0.83;0.86;0.89;0.91;...
   0.95;0.97;0.99;1.01;1.03;1.05;0.6;0.73;1.0;0.65;1.2;1.3];
[vfn,minvf,maxvf]=premnmx(vf);
```

```
% prepare feed flowrate "vf" as a fector for each case
vfn1=ones(21,1)*vfn(1);vfn2=ones(21,1)*vfn(2);vfn3=ones(21,1)*vfn(3);vf
n4=ones(21,1)*vfn(4);vfn5=ones(21,1)*vfn(5);vfn6=ones(21,1)*vfn(6);vfn7
=ones(21,1)*vfn(7);vfn8=ones(21,1)*vfn(8);vfn9=ones(21,1)*vfn(9);vfn10=
ones(21,1)*vfn(10);vfn11=ones(21,1)*vfn(11);vfn12=ones(21,1)*vfn(12);vf
n13=ones(21,1)*vfn(13);vfn14=ones(21,1)*vfn(14);vfn15=ones(21,1)*vfn(15
);vfn16=ones(21,1)*vfn(16);vfn17=ones(21,1)*vfn(17);vfn18=ones(21,1)*vf
n(18);vfn19=ones(21,1)*vfn(19);vfn20=ones(21,1)*vfn(20);vfn21=ones(21,1
)*vfn(21);vfn22=ones(21,1)*vfn(22);vfn23=ones(21,1)*vfn(23);vfn24=ones(
21,1)*vfn(24);vfn25=ones(21,1)*vfn(25);vfn26=ones(21,1)*vfn(26);
vfnt=[vfn1;vfn2;vfn3;vfn4;vfn5;vfn6;vfn7;vfn8;vfn9;vfn10;vfn11;vfn12;..
vfn13;vfn14;vfn15;vfn16;vfn17;vfn18;vfn19;vfn20;vfn21;vfn22;vfn23;vfn24
;vfn25;vfn26];
%Prepare inputs for HANN-1 model using normalized smoothed process data
%"Ptr_rb" is the normalized input of the  training cases (1-20) for ANN
%of rate of formation of cellobiose "rb"
Ptr_rb=[pn1(1:420) pn2(1:420) vfnt(1:420)]';
%"Pts_rb" is the normalized input of the  testing (interpolation and
% extrapolation)
%cases (21-26) for ANN of rate of formation of cellobiose "rb"
Pts_rb=[pn1(421:546) pn2(421:546) vfnt(421:546)]';
%"Ptr_rg" is the normalized input of the  training cases (1-20) for ANN
%of rate of formation of glucose "rg"
Ptr_rg=[pn2(1:420) pn3(1:420) vfnt(1:420)]';
%"Pts_rg" is the normalized input of the  testing (interpolation and
% extrapolation)
%cases (21-26) for ANN of rate of formation of glucose "rg"
Pts_rg=[pn2(421:546) pn3(421:546) vfnt(421:546)]';
      %Operating Parameters
      pp=1033.82;        % pressure on the shell side, g/cm.cm
      pf=1100;           % pressure at the entrance of the module
                         %(tube side), g/cm.cm
      pr=1070;           % pressure at the exit of the module,g/cm.cm
      l =200;            % tube length, cm
      r1=0.3;            % tube radius, cm
      lp=2.5e-5;         % hydraulic permeability,
                         %cc/[(cm.cm).min.(g/cm.cm)]
      % calculat "pt" transmembrane pressure drop for all cases
      pt=ones(21,26);
      for i=1:26
      pti=(pf-pp)+(pr-pf)*z/l;  % transmembrane pressure drop
      pt(:,i)=pti;
  end
   % v volumetric flow rates at a distance z from the entrance
   % of the reactor on the tube side
    v=ones(21,26);
   for i=1:26
   vi=-(2.0*pi*r1*lp)*((pf-pp)*z+(pr-pf)*z.*z/l/2.0)+vf(i);
   v(:,i)=vi;
    end
  % vbar volumetric flow rates at a distance z from the entrance
  dvdz=ones(21,26);
  for i=1:26
  dvdzi=-(2.0d0*pi*r1*lp)*((pf-pp)+(pr-pf)*z/l);
  dvdz(:,i)=dvdzi;
  end
  dvdz=dvdz(:);v=v(:);pt=pt(:);dc1=dc1';dc3=dc3';
```

```
  % Calculate reaction rates "rb" and "rg" for all cases (1-26)
  % rb are the rate of formation of cellobiose
  for i=1:546
  rb(i)=-dc1(i)/((pi*r1^2)/v(i))-cs(i,1)/(pi*r1^2)*dvdz(i);
  % rg are the rate of formation of glucose

rg(i)=dc3(i)/((pi*r1^2)/v(i))+cs(i,3)/(pi*r1^2)*dvdz(i)+((2*lp*pt(i))/r
1)*cs(i,3);
  end
% Normalize reaction rates "rb and rg"  between (-1 and 1)using premnmx
function
      [rbn,minrb,maxrb]=premnmx(rb);
      [rgn,minrg,maxrg]=premnmx(rg);
%Prepare targets for HANN1 model using normalized rate of reactions rb
and rg
%"Ttr_rb" is the normalized target of the  training cases (1-20) for
ANN
%of rate of formation of cellobiose "rb"
Ttr_rb=rbn(1:420);
%"Pts_rb" is the normalized target of the  testing (interpolation and
extrapolation)
% cases (21-26) for ANN of rate of formation of cellobiose "rb"
Tts_rb=rbn(421:546);
%"Ptr_rg" is the normalized target of the  training cases (1-20) for
ANN
% of rate of formation of glucoseg "rg"
Ttr_rg=rgn(1:420);
%"Pts_rg" is the normalized target of the  testing (interpolation and
extrapolation)
%cases (21-26) for ANN of rate of formation of glucose "rg"
Tts_rg=rgn(421:546);
```

B5- hann1.m

```
function cdot=hann1(z,c,flag,vf,rbnet,rgnet);
%This function(hann1) contains the structure of the first
% hybrid model which has the combination of the mass balance
% equations (5 ODEs) Ann which will predict the rate of reactions
% rb and rg
% Last update 5/14/03

%Operating Parameters
      pp=1033.82;        % pressure on the shell side, g/cm.cm
      pf=1100;           % pressure at the entrance of the module(tube
                         % side), g/cm.cm
      pr=1070;           % pressure at the exit of the module,g/cm.cm
      l =200;            % tube length, cm
      r1=0.3;            % tube radius, cm of the reactor, cc/min
      lp=2.5e-5;         % hydraulic permeability,
      % pt transmembrane pressure drop
      pt=(pf-pp)+(pr-pf)*z/l;
```

```
      % v volumetric flow rates at a distance z from the entrance of
      %the reactor on the tube side
      v=-(2.0*pi*r1*lp)*((pf-pp)*z+(pr-pf)*z^2/l/2.0)+vf;
      % vbar volumetric flow rates at a distance z from the entrance
       % of the reactor on the shell side
     vbar= vf-v+1.0e-15;
     dvdz=-(2.0d0*pi*r1*lp)*((pf-pp)+(pr-pf)*z/l);
      %  normalization parameters calculated from "hann_data.m"
      maxp1 = 0.0043; minp1 =  7.6182e-006;       % cellulose
      maxp2 =0.0049; minp2 = 0;                   % cellobiose  "tube"
      maxp3 = 0.0015; minp3 = 0;                  % glucose "tube"
      minvf=0.6; maxvf=1.3;                       % volumetric flowrate
      minrb=9.3791e-007 ;maxrb= 1.3956e-004;     % rate of formations
      minrg=0; maxrg= 1.0803e-005;               % rate of formations
  % Normalization of reactant concentrations: cn1,Cellobiose cn2 and
  % Glucose cn3 on the tube side.
  % Cellobiose cn4 and Glucose cn5 on the shell side.
   cn1= 2*(c(1)-minp1)/(maxp1-minp1) - 1;cn2 = 2*(c(2)-minp2)/(maxp2-
minp2)-1;
   cn3 = 2*(c(3)-minp3)/(maxp3-minp3) - 1;vfn = 2*(vf-minvf)/(maxvf-
minvf)-1;
   % Constrains for normalized concentrations in order to control ANN
behavior
   % rxnet is ANN to predict the normalized rate of formation of
cellobiose "rbn" and glucose "rgn"
   rbn=sim(rbnet,[cn1;cn2;vfn]);
   rgn=sim(rgnet,[cn2;cn3;vfn]);
   % de-normalized rbn and rbn
   rb= 0.5*(rbn+1)*(maxrb-minrb) + minrb;
   rg= 0.5*(rgn+1)*(maxrg-minrg) + minrg;
  % 5 ODEs to calculate the concentration profiles of cellulose
  % c(1),cellobiose c(2) and glucose c(3) on the tube side.
  % cellobiose c(4) and glucose c(5) on the shell side.
      cdot=[(pi*r1^2/v)*(-rb-c(1)*dvdz/pi/r1^2);(pi*r1^2/v)*(-rg+rb-...
      (c(2)*dvdz/pi/r1^2)-(2.0d0*lp*pt/r1)*c(2));(pi*r1^2/v)*...
      (rg-(c(3)*dvdz/pi/r1^2)-(2.0d0*lp*pt/r1)*c(3));(pi*r1^2/vbar)*...
      ((c(4)*dvdz/pi/r1^2)+(2.0*lp*pt/r1)*c(2));(pi*r1^2/vbar)...
      *((c(5)*dvdz/pi/r1^2)+(2.0*lp*pt/r1)*c(3))];
```

B6- hann2a_data.m

```
%This script file "hann2a_data" is used to prepare training,
%interpolation and extrapolation cases for HANN2 model
% Last update 8/9/03

%Operating Parameters
      pp=1033.82;         % pressure on the shell side, g/cm.cm
      pf=1100;            % pressure at the entrance of the module(tube
                          % side), g/cm.cm
```

```
      pr=1070;           % pressure at the exit of the module,g/cm.cm
      l =200;            % tube length, cm
      r1=0.3;            % tube radius, cm
      lp=2.5e-5;         % hydraulic permeability,
                         % cc/[(cm.cm).min.(g/cm.cm)]
      rm=1.39e-3;        % maximum reaction rate (cellobiose),
                         % g/(cc.min)
      rprimem=1.22e-3;   % maximum reaction rate (glucose)g/(cc.min)
% calculate "pt" transmembrane pressure drop for all cases
      pt=ones(21,26);
  for i=1:26
      pti=(pf-pp)+(pr-pf)*z/l;
      pt(:,i)=pti;
  end
 % Calculate  volumetric flow rates "v" at a distance z from the
 % entrance of the reactor
 % on the tube side
    v=ones(21,26);
   for i=1:26
   vi=-(2.0*pi*r1*lp)*((pf-pp)*z+(pr-pf)*z.*z/l/2.0)+vf(i);
   v(:,i)=vi;
    end
  % Calculation of Michaelis-Menten constants km and  and kprimem  for
  % all cases (1-26)
  % dc1 and dc3 are first derivatives of cellulose (tube) and glucose
  %(tube)
  % cs is  a matrix of all process data  for all cases (1-26)
  v=v(:);pt=pt(:);dc1=dc1';dc3=dc3';
  for i=1:546
  Km(i)=(rm*cs(i,1))/((2*lp*pt(i)*cs(i,1))/(pi*r1)-
dc1(i)*v(i)/(pi*r1^2))-cs(i,1);
  Kprimem(i)=(rprimem*cs(i,2))/(dc3(i)*v(i)/(pi*r1^2))-cs(i,2);
  end
% Normalize Km and  and Kprimem  for all cases (1-26)   between (-1 and
% 1)using premnmx function
      [Kmn,minKm,maxKm]=premnmx(Km);
      [Kprimemn,minKprimem,maxKprimem]=premnmx(Kprimem);
%Prepare targets for ANN-1 and ANN-2 in HANN2  using normalized Kmn and
Kprimemn.
%"Ttr_Kmn" is the normalized targets of the  training cases (1-20) for
ANN-1
Ttsr_Kmn=Kmn(1:420);
%"Tin_Kmn" is the normalized targets of the interpolation cases (21-23)
for ANN-1
Tin_Kmn=Kmn(421:483);
%"Tex_Kmn" is the normalized targets of the extrapolation cases (24-26)
for ANN-1
Tex_Kmn=Kmn(483:546);
%"Ttr_Kprimemn" is the normalized targets of the  training cases (1-20)
for ANN-2
Ttr_Kprimemn=Kprimemn(1:420);
%"Tin_Kprimemn" is the normalized targets of the interpolation cases
(21-23) for ANN-1
Tin_Kprimemn=Kprimemn(421:483);
%"Tex_Kprimemn" is the normalized targets of the extrapolation cases
(24-26) for ANN-1
Tex_Kprimemn=Kprimemn(483:546);
```

```
% Normalize process data between (-1 and 1)using premnmx function
 p1=cs(:,1);p2=cs(:,2);p3=cs(:,3);p4=cs(:,4);p5=cs(:,5);

[pn1,minp1,maxp1]=premnmx(p1);[pn2,minp2,maxp2]=premnmx(p2);[pn3,minp3,
maxp3]=premnmx(p3);
 [pn4,minp4,maxp4]=premnmx(p4);[pn5,minp5,maxp5]=premnmx(p5);
%Prepare inputs for ANN-1 and ANN-2 in HANN2 by using normalized
%process data using (pn1, pn2, pn3,and Vfnt)
% HANN1 models
%"Ptr_Km" is the normalized inputs of the  training cases (1-20) for
%ANN-1
Ptr_Kmn=[pn1(1:420) pn2(1:420) vfnt(1:420)]';
%"Pin_Km" is the normalized inputs of interpolation cases (21-23)for
%ANN-1
Pin_Kmn=[pn1(421:483) pn2(421:483) vfnt(421:483)]';
%"Pex_Km" is the normalized inputs of extrapolation cases (24-26)for
%ANN-1
Pex_Kmn=[pn1(484:546) pn2(484:546) vfnt(484:546)]';
%"Ptr_Kprimem" is the normalized inputs of the  training cases (1-20)
%for ANN-2
Ptr_Kprimemn=[pn2(1:420) pn3(1:420) vfnt(1:420)]';
%"Pin_Kprimem" is the normalized inputs of interpolation cases (21-
%23)for ANN-2
Pin_Kprimemn=[pn2(421:483) pn3(421:483) vfnt(421:483)]';
%"Pex_Kprimem" is the normalized inputs of extrapolation cases (24-
%26)for ANN-2
Pex_Kprimemn=[pn2(484:546) pn3(484:546) vfnt(484:546)]';
```

B7- hann2b_data.m

```
%This script file "hann2b_data" is used to prepare training,
% interpolation and extrapolation cases for HANN2b model
% Last update 7/15/03

%Operating Parameters
      pp=1033.82;        % pressure on the shell side, g/cm.cm
      pf=1100;           % pressure at the entrance of the module(tube
                         % side), g/cm.cm
      pr=1070;           % pressure at the exit of the module,g/cm.cm
      l =200;            % tube length, cm
      r1=0.3;            % tube radius, cm
      lp=2.5e-5;         % hydraulic permeability,
                         % cc/[(cm.cm).min.(g/cm.cm)]
      rm=1.39e-3;        % maximum reaction rate (cellobiose),
                         % g/(cc.min)
      rprimem=1.22e-3;   % maximum reaction rate (glucose)g/(cc.min)
% calculate "pt" transmembrane pressure drop for all cases
      pt=ones(21,26);
```

```
  for i=1:26
      pti=(pf-pp)+(pr-pf)*z/l;
      pt(:,i)=pti;
  end
 % Calculate  volumetric flow rates "v" at a distance z from the
 % entrance of the reactor
 % on the tube side
    v=ones(21,26);
   for i=1:26
   vi=-(2.0*pi*r1*lp)*((pf-pp)*z+(pr-pf)*z.*z/l/2.0)+vf(i);
   v(:,i)=vi;
    end
  % Calculation of Michaelis-Menten constants km and  and kprimem  for
  %all cases (1-26)
  % dc1 and dc3 are first derivatives of cellulose (tube) and glucose
  %(tube)
  % csm is  smoothed process data  for all cases
  v=v(:);pt=pt(:);dc1=dc1';dc3=dc3';
  for i=1:546
  Km(i)=(rm*cs(i,1))/((2*lp*pt(i)*cs(i,1))/(pi*r1)-
dc1(i)*v(i)/(pi*r1^2))-cs(i,1);
  Kprimem(i)=(rprimem*cs(i,2))/(dc3(i)*v(i)/(pi*r1^2))-cs(i,2);
  end
% Normalize Km and  and Kprimem  for all cases (1-26)   between (-1 and
% 1)using premnmx function
      [Kmn,minKm,maxKm]=premnmx(Km);
      [Kprimemn,minKprimem,maxKprimem]=premnmx(Kprimem);
%Prepare targets for ANN-1 and ANN-2 in HANN2  using normalized Km and
Kprimem.
%"Ttr_Kmn" is the normalized targets of the  training cases (1-20) for
%ANN-1
Ttr_Kmn=Kmn(1:420);
%"Tin_Kmn" is the normalized targets of the interpolation cases (21-23)
%for ANN-1
Tin_Kmn=Kmn(421:483);
%"Tex_Kmn" is the normalized targets of the extrapolation cases (24-26)
%for ANN-1
Tex_Kmn=Kmn(483:546);
%"Ttr_Kprimemn" is the normalized targets of the  training cases (1-20)
%for ANN-2
Ttr_Kprimemn=Kprimemn(1:420);
%"Tin_Kprimemn" is the normalized targets of the interpolation cases
(21-23) for ANN-1
Tin_Kprimemn=Kprimemn(421:483);
%"Tex_Kprimemn" is the normalized targets of the extrapolation cases
%(24-26) for ANN-1
Tex_Kprimemn=Kprimemn(483:546);

%Prepare inputs for ANN-1 and ANN-2 in HANN2 by using normalized
%smoothed process data using
% The same normalized smoothed process data (pn1, pn2, pn3,and
Vfnt)generated for BANN and
% HANN1 models
%"Ptr_Km" is the normalized inputs of the  training cases (1-20) for
%ANN-1
Ptr_Kmn=[pn1(1:420) pn2(1:420) vfnt(1:420)]';
```

```
%"Pin_Km" is the normalized inputs of interpolation cases (21-23)for
%ANN-1
Pin_Kmn=[pn1(421:483) pn2(421:483) vfnt(421:483)]';
%"Pex_Km" is the normalized inputs of extrapolation cases (24-26)for
%ANN-1
Pex_Kmn=[pn1(484:546) pn2(484:546) vfnt(484:546)]';
%"Ptr_Kprimem" is the normalized inputs of the  training cases (1-20)
%for ANN-2
Ptr_Kprimemn=[pn2(1:420) pn3(1:420) vfnt(1:420)]';
%"Pin_Kprimem" is the normalized inputs of interpolation cases (21-
%23)for ANN-2
Pin_Kprimemn=[pn2(421:483) pn3(421:483) vfnt(421:483)]';
%"Pex_Kprimem" is the normalized inputs of extrapolation cases (24-
%26)for ANN-2
Pex_Kprimemn=[pn2(484:546) pn3(484:546) vfnt(484:546)]';
```

B8- hann2.m

```
function cdot=hann2(z,c,flag,vf,Kmnet,Kprimemnet);
%This function(hann2) contains the structure of the second
% hybrid model which has the combination of the mass balance
% equations (5 ODEs), simplified rate expressions, and ANN
% In this model ANN (ANN-1 and ANN-2 ) are used to predict
% Michaelis-Menten constants km and and kprimem
% Last update 7/20/03

%Operating Parameters
      pp=1033.82;        % pressure on the shell side, g/cm.cm
      pf=1100;           % pressure at the entrance of the module(tube
                         % side), g/cm.cm
      pr=1070;           % pressure at the exit of the module,g/cm.cm
      l =200;            % tube length, cm
      r1=0.3;            % tube radius, cm of the reactor, cc/min
      lp=2.5e-5;         % hydraulic permeability,
      rm=1.39e-3;        % maximum reaction rate (cellobiose),
                         % g/(cc.min)
      rprimem=1.22e-3;   % maximum reaction rate (glucose)g/(cc.min)
      %  calculate pt transmembrane pressure drop
      pt=(pf-pp)+(pr-pf)*z/l;
      % calculate volumetric flow rates at a distance z from the
      % entrance of
      % the reactor on the tube side (v)
      v=-(2.0*pi*r1*lp)*((pf-pp)*z+(pr-pf)*z^2/l/2.0)+vf;
      % calculate  volumetric flow rates at a distance z from the
      % entrance
      % of the reactor on the shell side (vbar)
     vbar= vf-v+1.0e-15;
     dvdz=-(2.0d0*pi*r1*lp)*((pf-pp)+(pr-pf)*z/l);

  %  Normalization parameters.
      maxp1 = 0.0043; minp1 =  7.6182e-006;       % cellulose
      maxp2 =0.0049; minp2 = 0;                   % cellobiose  "tube"
```

```
      maxp3 = 0.0015; minp3 = 0;                  % glucose "tube"
      minvf=0.6; maxvf=1.3;                       % volumetric flowrate
     minKm =0.0422; maxKm =0.1526;                % Michaelis-Menten
     %constants(cellobiose)
     minKprimem =0.1870; maxKprimem =0.6858;      % Michaelis-Menten
     %constants (glucose)
  % Normalization of reactant concentrations: cellulose
(cn1),Cellobiose (cn2), and Glucose (cn3)
  % in the tube side.
   cn1= 2*(c(1)-minp1)/(maxp1-minp1) - 1;cn2 = 2*(c(2)-minp2)/(maxp2-
minp2)-1;
   cn3 = 2*(c(3)-minp3)/(maxp3-minp3) - 1;vfn = 2*(vf-minvf)/(maxvf-
minvf)-1;

% Kmnet is ANN to predict the normalized Michaelis-Menten constants Km.
% Kprimemnet is ANN to predict the normalized Michaelis-Menten
% constants Kprimemn.
   Kmn=sim(Kmnet,[cn1;cn2;vfn]);
   Kprimemn=sim(Kprimemnet,[cn2;cn3;vfn]);

   % de-normalized Kmn and Kprimemn
   Km= 0.5*(Kmn+1)*(maxKm-minKm) + minKm;
   Kprimem= 0.5*(Kprimemn+1)*(maxKprimem-minKprimem) + minKprimem;

  % Calculation of rb  using simplified  rate of formation of
cellobiose
    rb=rm*c(1)./(Km+c(1));
 %  Calculation of rg using simplified  rate of formation of glucose
    rg=rprimem*c(2)/(Kprimem+c(2));

  % 5 ODEs to calculate the concentration profiles of cellulose
  % c(1),cellobiose c(2) and glucose c(3) on the tube side.
  % cellobiose c(4) and glucose c(5) on the shell side.
      cdot=[(pi*r1^2/v)*(-rb-c(1)*dvdz/pi/r1^2);(pi*r1^2/v)*(-rg+rb-...
      (c(2)*dvdz/pi/r1^2)-(2.0d0*lp*pt/r1)*c(2));(pi*r1^2/v)*...
      (rg-(c(3)*dvdz/pi/r1^2)-(2.0d0*lp*pt/r1)*c(3));(pi*r1^2/vbar)*...
      ((c(4)*dvdz/pi/r1^2)+(2.0*lp*pt/r1)*c(2));(pi*r1^2/vbar)...
      *((c(5)*dvdz/pi/r1^2)+(2.0*lp*pt/r1)*c(3))];
```