ELSEVIER

Chemical Engineering Journal 69 (1998) 135–143

Chemical Engineering Journal

Prediction of polymer quality in batch polymerisation reactors using robust neural networks

J. Zhang [a,*], A.J. Morris [a], E.B. Martin [a], C. Kiparissides [b]

[a] *Centre for Process Analysis, Chemometrics and Control, Department of Chemical and Process Engineering, University of Newcastle, Newcastle upon Tyne NE1 7RU, UK*

[b] *Department of Chemical Engineering, Aristotle University of Thessaloniki, Thessaloniki 54006, Greece*

Received 27 June 1997; revised 4 December 1997; accepted 10 December 1997

Abstract

A technique for predicting polymer quality in batch polymerisation reactors using robust neural networks is proposed in this paper. Robust neural networks are used to learn the relationship between batch recipes and the trajectories of polymer quality variables in batch polymerisation reactors. The robust neural networks are obtained by stacking multiple nonperfect neural networks which are developed based on the bootstrap re-samples of the original training data. Neural network generalisation capability can be improved by combining several neural networks and neural network prediction confidence bounds can also be calculated based on the bootstrap technique. A main factor affecting prediction accuracy is reactive impurities which commonly exist in industrial polymerisation reactors. The amount of reactive impurities is estimated on-line during the initial stage of polymerisation using another neural network. From the estimated amount of reactive impurities, the effective batch initial condition can be worked out. Accurate predictions of polymer quality variables can then be obtained from the effective batch initial conditions. The technique can be used to design optimal batch recipes and to monitor polymerisation processes. The proposed techniques are applied to the simulation studies of a batch methylmethacrylate polymerisation reactor.

Keywords: Neural networks; Batch processes; Polymerisation reactor; Optimal control

1. Introduction

Batch reactors are suitable for manufacturing high value added specialty chemicals such as specialty polymers, pharmaceuticals, and biochemicals. Optimal control of batch polymerisation reactors have been studied by several researchers (e.g., Refs. [1–8]). The main objective of these optimal control strategies is to obtain a product with desired physical and mechanical properties within a minimum time. Optimal profiles of reactor temperature and/or initiator adding policies are calculated based on first principles models of polymerisation processes.

The development of detailed first principles models for complex polymerisation processes is usually very time consuming and effort demanding. It is quite common for a comprehensive mechanistic polymerisation model to involve dozens of differential and algebraic equations. Empirical models based on neural networks can be used to ease the effort in model development. Neural networks have been shown to be able to approximate any continuous nonlinear functions (e.g. Refs. [9–11]) and have been applied to process modelling and control (e.g., Refs. [12–15]).

A key issue in neural network based modelling is the network generalisation capability. The neural network model should perform reliably when applied to unseen data. Neural network generalisation capability is mainly determined by the network training method and training data. To build an accurate neural network model, ideally the training data should be abundant and cover a wide range of the system input space. In many practical situations, the amount of training data is often limited due to the cost in conducting experiments and the difficulties in measuring some physical variables such as polymer quality variables. In batch polymerisation production, polymer quality variables including the number average molecular weight and the weight average molecular weight are usually measured through laboratory analysis and, typically, only a very limited samples of these measurements are made during a batch. When the amount of training data is limited, a neural network model often tends to over-fit the training data and result in significant errors when applied to unseen data. To overcome the difficulty due

* Corresponding author. Tel.: +44-191-2227240; fax: +44-191-2225292; e-mail: jie.zhang@newcastle.ac.uk

1385-8947/98/$19.00
PII S1385-8947(98)00069-2

to limited process data, Tsen et al. [16] propose to augment experimental data by using an approximate mechanistic model of the process. They used the first order Taylor series expansion of the approximate mechanistic model to extrapolate around the experimental data. Augmented data are generated in this way. They applied this technique to polymer quality control in a batch polymerisation reactor. The limitations of this approach are the need for an appropriate mechanistic model and the potential problem of the assumed linear behaviour around the operating point.

Some neural network training techniques have been developed to improve network generalisation capability. One of the techniques is training with regularisation [10,17]. The aim of regularisation is to prevent unnecessarily large neural network weights which can result in large prediction errors on unseen data. The idea of regularisation has been widely used in statistical model building and a variety of techniques, such as ridge regression, principal component regression, and partial least squares regression, have been developed. Neural network generalisation capability can also be improved by using a parsimonious network structure. Network pruning techniques have been developed to remove unnecessary neurons [18]. A sequential orthogonal training technique for building parsimonious network using mixed types of hidden neurons was proposed by Zhang et al. [19]. An attractive approach to improve neural network model robustness is to develop a set of neural network models and combine them. The combined neural network model is known as a stacked neural network model [20–22]. In a stacked neural network, the final model prediction is a combination of the predictions from the individual neural networks.

In this paper, we propose a robust neural network based technique for predicting trajectories of polymer quality variables in batch polymerisation reactors from batch recipes. Stacked neural networks are used to learn the relationship between batch recipes and the trajectories of polymerisation quality variables. A practically very important aspect in polymerisation is the problem of reactive impurities. The economic operation of polymer reactors requires to recover unreacted monomers and solvent. The recovered monomer and solvent are recycled back to polymerisation reactors. This will inevitably introduce reactive impurities which are mainly in the form of oxygen and traces of inhibitors. Reactive impurities can rapidly consume free radicals and cease or slow down the polymerisation process. When there exist reactive impurities, the effective batch initial condition will be different from that defined by the batch recipe. Predictions of polymer quality variables based on the nominal batch initial condition will therefore possess significant errors. A neural network based technique for estimating reactive impurities has been developed by Zhang et al. [23]. The technique can accurately estimate the amount of reactive impurities during the initial stage of a batch. Once the amount of reactive impurities has been estimated, the effective batch initial conditions can be worked out. Accurate predictions of trajectories of polymer quality variables can then be obtained based on the effective batch initial conditions. A batch polymerisation reactor can be monitored and controlled based upon the predicted trajectories of polymer quality variables. If the predicted final polymer quality differs from the desired value then appropriate control actions, such as varying reactor temperature or varying batch ending time, should be taken to prevent any off-specification products being produced. The neural network model for predicting polymer quality can also be used to design optimal batch recipes.

The paper is organised as follows. Section 2 discusses neural network based process modelling. Problems in conventional neural network modelling are discussed and stacked neural networks are presented. Section 3 describes the prediction of polymer quality using neural networks. Disturbance estimation is presented in Section 4. Application of the proposed technique to a simulated batch polymerisation reactor is presented in Section 5. Section 6 discusses optimal batch recipe design based on neural network models. Section 7 concludes this paper.

2. Neural network based process modelling

2.1. Multilayer feed forward neural networks

Neural networks have been shown to be able to approximate any continuous nonlinear functions (e.g., Refs. [9–11]) and have been applied to nonlinear process modelling and control recently (e.g., Refs. [12–15]). The most commonly used neural network architecture is the multilayer feed forward neural network shown in Fig. 1. The basic feed forward network performs a nonlinear transformation of the input data in order to approximate the output data.

Inputs to a neural network are presented at the input layer. The data from the input neurons is then propagated through the network via the interconnections such that every neuron in a layer is connected to every neuron in the adjacent layers. It is the hidden layer structures which essentially define the topology of a feed forward network. Each interconnection has associated with it a scalar weight which acts to modify the strength of the signal passing through it. The neurons within the hidden layer perform two tasks: they sum the weighted inputs to the neuron and then pass the resulting summation through a nonlinear activation function. In addition to the weighted inputs to the neuron, a bias is included

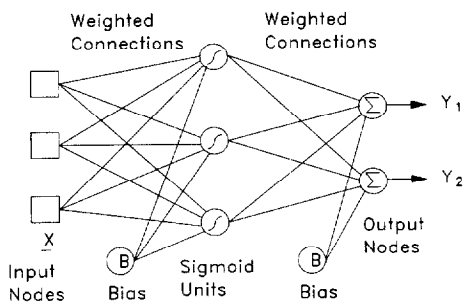

Fig. 1. A multilayer feed forward neural network.

in order to shift the space of the nonlinearity. The output of a hidden neuron can be represented as follows:

$$S=\sum_{i=1}^{nh}(b+w_iI_i) \tag{1}$$

$$O=\frac{1}{1+\exp(-S)} \tag{2}$$

where b is a bias, I_i is the ith input to the hidden neuron, w_i is the weight associated with I_i, and O is the hidden neuron output. Eq. (2) is known as the sigmoidal neuron activation function and its output is in the range (0, 1). Output layer neurons can also use the sigmoidal activation function. However, for process modelling applications, output layers neurons usually use the linear activation function since it can give a wide range of outputs.

Network weights are such trained so that the sum of squared network prediction errors is minimised. The training objective function can be defined as follows:

$$J=\frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t)-y(t))^2 \tag{3}$$

where N is the number of training data points, $\hat{y}$ is the network prediction, y in the target value, and t is an index of the training data. The most commonly used network training method is the back propagation training method [24], where network weights are adjusted as follows.

$$\Delta W(k+1)=\alpha\Delta W(k)-\eta\frac{\partial J}{\partial W(k)} \tag{4}$$

$$W(k+1)=W(k)+\Delta W(k+1) \tag{5}$$

In Eqs. (4) and (5), $W(k)$ and $\Delta W(k)$ are the weight and weight adaptation at the training step k, respectively, α is the momentum coefficient, and η is the learning rate. Training can be terminated when the error gradient is less than a prespecified value, e.g., 10^{-6}. Training can also be terminated by a cross validation based stopping criterion. When using a cross validation based stopping criterion, data for building a neural network model is divided into a training data set and a testing data set. During network training, the network prediction error on the testing data is continuously monitored. Training is terminated when the testing error stops decreasing.

Although in theory a neural network can approximate any continuous nonlinear functions, a perfect neural network model is usually very difficult, if not impossible, to build in practice, especially when the amount of training data is limited. This is due to several factors. First, network training is a nonlinear optimisation problem which is solved through numerical search methods. There is no guarantee that the global minimum will be reached and network training may converge to a local minimum. Secondly, data collected from an industrial process will inevitably contain measurement noise. Over-fitting of noise can seriously deteriorate the network generalisation capability and result in significant prediction errors when the network is applied to unseen data.

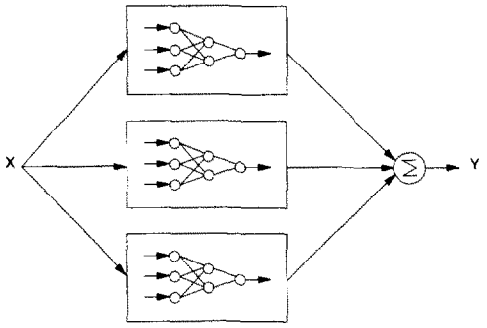

Fig. 2. A stacked neural network.

2.2. Stacked neural networks

In recognition of the difficulties in building a perfect neural network model, several researchers have recently shown that a robust neural network model with improved generalisation capability can be obtained by combining several nonperfect neural network models (e.g., Refs. [20–22,25,26]). The combination of multiple neural networks results in a stacked neural network.

A diagram for a stacked neural network is shown in Fig. 2, where several neural network models are developed to model the same relationship and are combined together. The individual neural networks are trained using different training data sets and/or from different initial weights. Instead of selecting a single neural network model, a stacked neural network model combines several neural networks to improve model accuracy and robustness. The overall output of the stacked neural network is a weighted combination of the individual neural network outputs. This can be represented by the following equation.

$$f(X)=\sum_{i=1}^{n}w_if_i(X) \tag{6}$$

where $f(X)$ is the stacked neural network predictor, $f_i(X)$ is the ith neural network predictor, w_i is the stacking weight for combining the ith neural network, n is the number of neural networks, and X is a vector of neural network inputs.

Stacking weights can be determined in a number of ways. A simple approach is to take equal weights for the individual networks and the stacking weights are all at $1/n$. Another approach to obtain the stacking weights is through multiple linear regression. However, this approach has problems due to the severe correlation among the individual predictors. Since each network is developed to model the same relationship, these networks are usually highly correlated. We found that obtaining stacking weights through multiple linear regression does not give good performance. This was also experienced by Breiman [27] and he suggests to put a constraint on the stacking weights such that they are nonnegative. Since the individual neural networks are highly correlated, appropriate stacking weights could be obtained through principal component regression (PCR) [22].

Let $\mathbf{y}$ be a vector of the expected model outputs and $\hat{\mathbf{y}}_i$ be a vector of the predictions from the ith neural network predictor. Predictions from a set of n predictors can be put in a matrix as follows.

$$\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1\ \hat{\mathbf{y}}_2 \cdots \hat{\mathbf{y}}_n] \tag{7}$$

where each column corresponds to an individual predictor. The vector of predictions from the stacked neural network model, $\hat{\mathbf{y}}_{\text{stack}}$, can be represented as

$$\begin{aligned} \hat{\mathbf{y}}_{\text{stack}} &= \hat{\mathbf{Y}}w \\ &= w_1\hat{\mathbf{y}}_1 + w_2\hat{\mathbf{y}}_2 + \ldots + w_n\hat{\mathbf{y}}_n \end{aligned} \tag{8}$$

The matrix $\hat{\mathbf{Y}}$ can be decomposed into the sum of a series of rank one matrices through principal component decomposition.

$$\hat{\mathbf{Y}} = t_1 p_1^T + t_2 p_2^T + \ldots + t_n p_n^T \tag{9}$$

In the above equation, t_i and p_i are the ith score vector and loading vector, respectively. The score vectors are orthogonal, likewise the loading vectors, in addition they are of unit length. The loading vector p_1 defines the direction of the greatest variability and the score vector t_1, also known as the first principal component, represents the projection of each column of $\hat{\mathbf{Y}}$ onto p_1. Thus, the first principal component is that linear combination of the columns in $\hat{\mathbf{Y}}$ explaining the greatest amount of variability ($t_1 = \hat{\mathbf{Y}}p_1$). The second principal component is that linear combination of the columns in $\hat{\mathbf{Y}}$ explaining the next greatest amount of variability ($t_2 = \hat{\mathbf{Y}}p_2$) subject to the condition that it is orthogonal to the first principal component. Principal components are arranged in decreasing order of variability explained. Since the columns in $\hat{\mathbf{Y}}$ are highly correlated, the first few principal components can explain the majority of variability in $\hat{\mathbf{Y}}$.

Through PCR, the stacked neural network model output is obtained as a linear combination of the first few principal components of $\hat{\mathbf{Y}}$. Suppose that the first k principal components are used in PCR and they are denoted by T_k and $T_k = \hat{\mathbf{Y}}P_k$, where $P_k = [p_1\, p_2 \ldots p_k]$, then the stacked neural network model can be represented as

$$\hat{\mathbf{y}}_{\text{stack}} = T_k\theta = \hat{\mathbf{Y}}P_k\theta \tag{10}$$

The least squares estimation of θ is:

$$\hat{\theta} = (T_k^T T_k)^{-1} T_k^T y = (P_k^T \hat{\mathbf{Y}}^T \hat{\mathbf{Y}} P_k)^{-1} P_k^T \hat{\mathbf{Y}}^T y \tag{11}$$

The stacking weight vector $\boldsymbol{w}$ calculated through PCR is then

$$\boldsymbol{w} = P_k\hat{\theta} = P_k(P_k^T \hat{\mathbf{Y}}^T \hat{\mathbf{Y}} P_k)^{-1} P_k^T \hat{\mathbf{Y}}^T y \tag{12}$$

We have found that the weights determined from PCR give very good performance. The number of principal components used can be found through cross validation. Different numbers of principal components are studied and the resulting model errors on testing data are compared. The number of principal components used is then determined based on the model errors on the testing data.

The method for building a stacked neural network model is summarised as follows. Firstly, data for building neural network models are re-sampled to form different training data sets. Bootstrap re-sampling [28] with replacement can be used. The idea of bootstrap is to suppose that a cumulative distribution function (CDF) $\hat{F}_n$ calculated from an observed sample $X_1, \ldots, X_n$ is sufficiently like the unknown CDF F so that one can use a calculation performed using $\hat{F}_n$ as an estimate of the calculation that we would like to perform using F. Distribution of the training data obtained through bootstrap re-sampling is similar to the original data distribution. Secondly, a neural network model is developed for each set of training data. Finally, the individual networks are combined together through PCR.

A problem in industrial applications of neural network models is the current lack of model prediction confidence bounds. The bootstrap re-sampling techniques can be used to estimate the standard errors of model predictions [28,29]. Based on the estimated standard errors, confidence bounds for neural network model predictions can be calculated. Neural network prediction confidence bounds give the process operator extra information about the predictions. The process operator can accept or reject a particular prediction from a neural network model by using the associated prediction confidence bounds.

Tibshirani [29] compared several error estimates for neural network models. These error estimate methods including the delta method [28], the sandwich method [30], and the bootstrapping method. It is shown that the bootstrapping method gives better estimate than other methods. The bootstrapping method for calculating neural network prediction confidence bounds is summarised as follows.

Step 1. Generate B samples, each one of size n drawn with replacement from the n training observations $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. Denote the bth sample by $\{(x_1^b, y_1^b), (x_2^b, y_2^b), \ldots, (x_n^b, y_n^b)\}$.

Step 2. For each bootstrap sample $b = 1, 2, \ldots, B$, train a neural network model. Denote the resulting neural network weights by W^b.

Step 3. Estimate the standard error of the ith predicted value by

$$\left\{\frac{1}{B-1}\sum_{b=1}^{B}[y(x_i;\ W^b) - y(x_i;\ \cdot)]^2\right\}^{1/2}$$

where $y(x_i;\ \cdot) = \sum_{b=1}^{B} y(x_i;\ W^b)/B$.

Step 4. Calculate the 95% confidence bounds by taking plus and minus 1.96 times the standard error of the mean of the predicted values.

3. Prediction of polymerisation trajectory

The dynamic model for a batch polymerisation reactor can be represented in the form of a nonlinear state space model as follows.

$$d\boldsymbol{x}/dt = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad (13)$$

$$\boldsymbol{y}(t) = g(\boldsymbol{x}(t)) \quad (14)$$

where $\boldsymbol{x}$ is a vector of state variables, $\boldsymbol{y}$ is a vector of polymer quality variables, and $\boldsymbol{u}$ is a vector of controls which are specified by the batch recipe. Eqs. (13) and (14) can be derived based on polymerisation kinetics, material balance, and energy balance. Due to the complexity in polymerisation kinetics, Eqs. (13) and (14) are generally very complicated and involves a large number of reaction constants which depend on some process variables such as the reactor temperature. Hence it is very effort demanding to develop such a complex first principles model. The calculation of polymer quality variables based on Eqs. (13) and (14) involves the numerical integration of a large number of differential equations. Furthermore, long term prediction of polymer quality variables will involve numerical integration over a long time period since many intermediate state variables are not measured. Errors due to the inaccuracy of certain model parameters can accumulate to a significant level when taking numerical integration over a long time interval. To overcome these problems, a stacked neural network is used to build a model which links the batch recipe, U, with a trajectory of polymer quality variables, $\boldsymbol{y}(t)$, $t = 1, 2, \ldots, n$.

Let $Y = [\boldsymbol{y}_1 \boldsymbol{y}_2 .. \boldsymbol{y}_n]$ be n points from the trajectory of a polymer quality variable, for example, the number average molecular weight. When there are no disturbances, Y is mainly determined by the batch recipe, U. Given the experimental data to a number of batches, it is possible to learn the relationships between U and Y using a neural network. The neural network model for predicting Y from U has the following form:

$$Y = p(U) \quad (15)$$

where $p()$ is a nonlinear function represented by a stacked neural network. Once a network has been trained to model the relationships between U and Y, it can be used to predict Y from U.

4. Disturbance estimation

The economic operation of a polymerisation process usually requires that unreacted species be recovered and recycled back into the process [31]. Associated with the recycle of solvent and unreacted monomers is also the recycle of reactive impurities which are introduced into the system in the fresh feed or as a by-product of chemical reactions. The levels of reactive impurities can be built up to the point where the reacting system is severely affected. Almost all types of polymerisation are sensitive to reactive impurities. In polymerisation processes, reactive impurities are usually traces of inhibitors or oxygen. The studies in Ref. [32] show that impurities in an emulsion system consume rapidly reactive free radicals, thus, preventing particle generation and reducing the growth of any polymer particles already present. Since reactive impurities can rapidly consume free radicals, their effect can generally be represented by a step decrease in initiator concentration or initiator efficiency [33]. The gross initial initiator weight can be expressed as the sum of the effective initiator weight and the amount of impurities as follows:

$$I_{0g} = I_0 + \Delta I_0 \quad (16)$$

where I_{0g} is the gross initial initiator weight, I_0 is the effective initial initiator weight, and ΔI_0 is the amount of impurities.

When there exist reactive impurities, the effective batch initial condition will be different from the nominal initial condition defined by the batch recipe. Predictions of polymer qualities based on the nominal initial condition could posses significant errors when there exist reactive impurities. Accurate predictions of polymer qualities can only be obtained when the amount of reactive impurities can be estimated. A neural network based technique for the estimation of reactive impurities has been developed by Zhang et al. [23]. The techniques can accurately estimate the amount of reactive impurities during the early stage of polymerisation.

In this method, a neural network based inverse model which maps a trajectory of monomer conversions to the corresponding initial initiator concentration is developed. The neural network model takes the following form.

$$I_0 = f(T, X(t_1), X(t_2), \cdots, X(t_n)) \quad (17)$$

where I_0 is the initial initiator weight, T is the reactor temperature, $X(t_1)$ to $X(t_n)$ are n discrete points in the monomer conversion trajectory during the early stage of a batch. Given a set of conversion measurements, the neural network model can be used to estimate the effective initial initiator weight. In this case, the amount of impurities is estimated as the difference between the gross initial initiator weight and the estimated effective initial initiator weight.

5. Application to a batch MMA polymerisation reactor

5.1. The batch polymerisation reactor

The batch polymerisation reactor studied in this paper is a pilot scale polymerisation reactor developed in the Department of Chemical Engineering, Aristotle University of Thessaloniki, Greece. The batch polymerisation reactor is shown in Fig. 3. The free-radical solution polymerisation of methylmethacrylate (MMA) is considered in this paper. The solvent used is water and the initiator used is benzoyl peroxide. The jacketed reactor is provided with a stirrer for thorough mixing of the reactants. Heating and cooling of the reaction mixture is achieved by circulating water at appropriate temperature through the reactor jacket. The reactor temperature is controlled by a cascade control system consisting of a primary PID and two secondary PID controllers. The reactor temperature is fed back to the primary controller whose output is taken as the setpoint of the two secondary controllers.

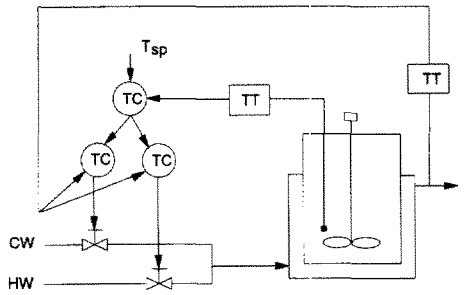

Fig. 3. A batch polymerisation reactor.

The manipulated variables for the two secondary controllers are hot and cold water flow rates. The hot and cold water streams are mixed before entering the reactor jacket and provide heating or cooling for the reactor. The jacket outlet temperature is fed back to the two secondary controllers. A simulation programme is developed and is used to test the techniques developed in this paper.

5.2. *Prediction of batch polymerisation trajectories*

In this reactor, batch recipes mainly include the reactor temperature setpoint and the initial initiator concentration. Neural network models are developed to predict trajectories of monomer conversions (X), number average molecular weights (Mn), and weight average molecular weights (Mw) from initial batch recipes. In this reactor, the nominal batch time is from 2 to 3 h. Polymer quality variables at 60, 80, 100, 120, 140, 160, and 180 min from the start of a batch are predicted from neural network models. A stacked neural network is developed for each of these time instant and it has two inputs and three outputs. The two network inputs are reactor temperature setpoint and the initial initiator concentration while the three network outputs are Mn, Mw, and X.

To generate training and testing data, 30 batches were simulated with batch recipes generated through Monte-Carlo simulation. For each batch, polymer quality variables at the seven discrete time instants were collected. To make the simulation close to reality, measurement noises are added to the polymer quality variables and they are in the ranges [−3000 g/mol, 3000 g/mol], [−6000 g/mol, 6000 g/mol], and [−1%, 1%] for Mn, Mw, and X, respectively. Bootstrap re-sampling with replacement [28] were used to generate 50 replica of the data set. For each re-sampled data set, 80% of the data points were randomly selected as training data while the remaining serve as testing data. A single hidden layer feed forward neural network was developed for each of the re-sampled data set. Each network contains 15 hidden neurons and the network weights were initialised as random numbers in the range (−0.1, 0.1). Networks were trained using the Levenberg–Marquardt optimisation [34] algorithm with regularisation. Training is terminated using a cross validation based 'early stopping' rule. During network training, the training algorithm continuously checks the network error on the testing data. Training is terminated at the point where the network error on the testing data is at its minimum. Early stopping is an implicit way to implement regularisation which can improve network robustness [35].

The 50 individual networks were combined together through PCR. A further 20 batches were generated as unseen validation data to test the reliability of the developed neural network models. Figs. 4 and 5 show the predicted and simulated polymer quality variables at 100 min and 160 min, respectively, on the 20 unseen validation batches. The 95% confidence bounds for the predictions are also shown. In Figs. 4 and 5, the simulated polymer quality variables are represented by 'o', the predicted values from the stacked neural networks are represented by '+', and the 95% confidence bounds are represented by the dash–dot lines. It can be seen that predictions are very accurate. The prediction confidence bounds give the process operator extra information on how confident the predictions are. Based on the confidence bounds, the process operator can accept or reject a particular prediction from a neural network model. Fig. 4 shows that the prediction confidence bounds are quite narrow for all the 20 batches indicating that these predictions are reliable. Fig. 5

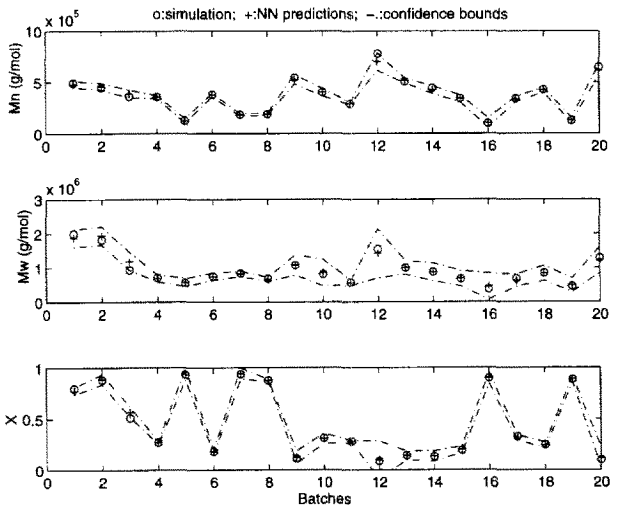

Fig. 4. Predictions of polymer quality at 100 min.

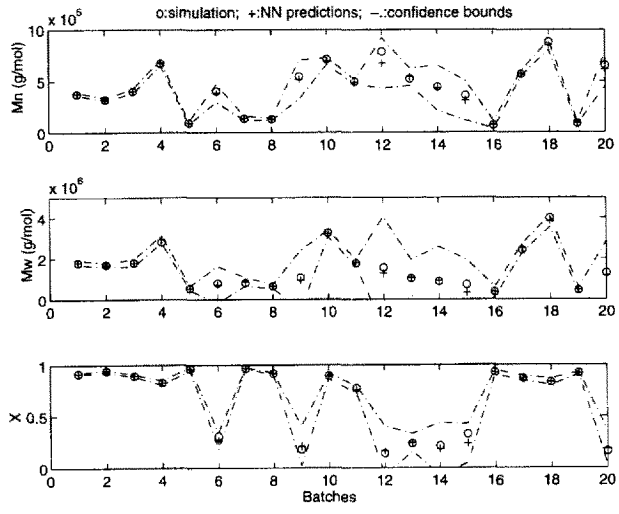

Fig. 5. Predictions of polymer quality at 160 min.

shows that the prediction confidence bounds for batches 12 to 15 are quite wide indicating the lack of confidence in these predictions. The process operator should therefore treat these predictions with care.

For the purpose of comparison, single neural network models were also developed to predict polymer quality variables. Seven single neural network models were developed to predict polymer quality variables at the above mentioned time instants. Each of these networks has 15 hidden neurons and network weights were initialised as random numbers in the range $(-0.1, 0.1)$. These networks were trained using the Levenberg–Marquardt optimisation algorithm with regularisation. Data from the first 30 batches were partitioned into a training data set and a testing data set. Data from the first 20 batches were used as training data while data from the remaining 10 batches were used as testing data. During network training, the training algorithm continuously checks the network error on the testing data. Training is terminated at the point where the network error on the testing data is at its minimum. Figs. 6–8 compare the scaled sum of squared errors (SSE) of the stacked neural network models and the single neural network models on the 20 unseen validation batches. It is clearly indicated that the stacked neural network models perform much better than the single neural network models.

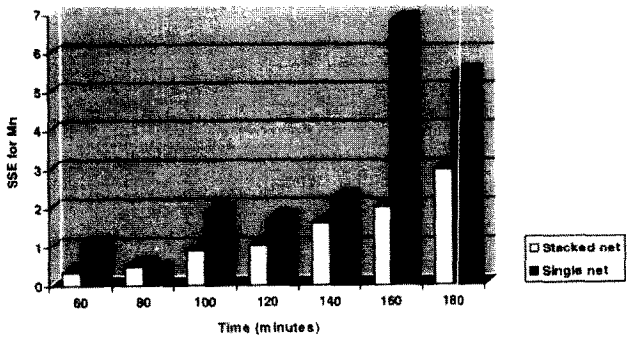

Fig. 6. Scaled SSE in predicting Mn on the validation data.

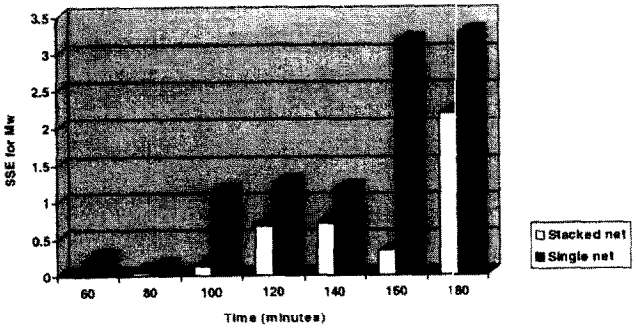

Fig. 7. Scaled SSE in predicting Mw on the validation data.

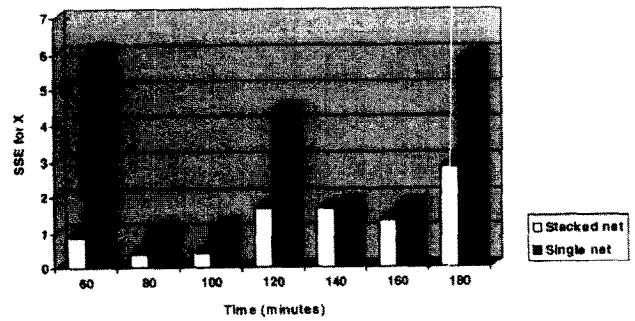

Fig. 8. Scaled SSE in predicting X on the validation data.

5.3. A case study

Consider a batch with the following nominal recipe: the reactor temperature setpoint is 351 K and the initial initiator weight is 2.5 g. Reactive impurities were added to this batch and their amount was arbitrarily taken as 0.83 g. Without knowing the existence of reactive impurities, predictions of polymer qualities based on the nominal batch recipe possess significant error as can be seen from Fig. 9. In Fig. 9, simulated process measurements are represented by 'o', neural network predictions based on the nominal batch recipe are represented by '*'.

To detect and estimate the amount of reactive impurities during the early stage of a batch, monomer conversion 'measurements' at 15, 20, 25, and 30 min from the batch start were collected. Using the technique presented in Section 4, the existence of reactive impurities was detected and their amount was estimated as 0.87 g, which is very close to the true amount of reactive impurities. Deducting this amount of reactive impurities from the nominal initial initiator weight, the effective initial initiator weight is then estimated as 1.63 g. Neural network predictions of polymer qualities based on this effective initial initiator weight are shown in Fig. 9 where they are represented as '+'. It can be seen from Fig. 9 that the predictions based on the effective initial initiator weight are very accurate. This demonstrates that the neural network based polymer quality prediction technique, when combined with the impurity estimation technique, can accurately predict trajectories of polymer quality variables even in the presence of an arbitrary amount of reactive impurities.

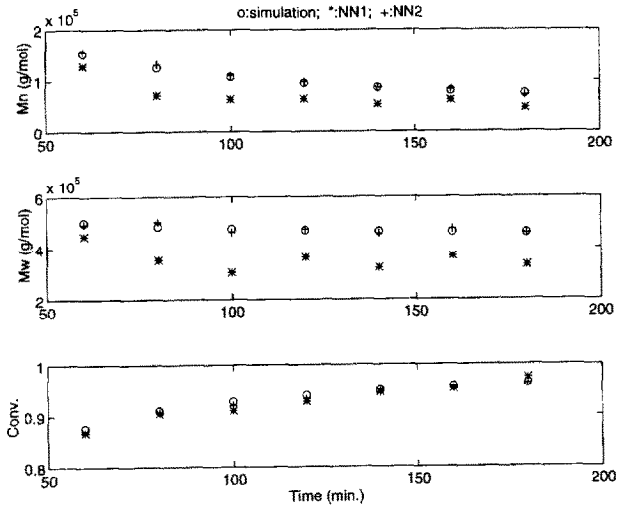

Fig. 9. Predicted polymerisation trajectories.

6. Optimum recipe design

The proposed technique can be used to design optimum batch recipes. The objective in operating batch polymerisation processes is to produce polymer products with desired quality and high monomer conversion at a minimum time. This can be expressed as the following objective function:

$$J = w_1(\mathrm{Mn}(t_f)/\mathrm{Mn}_d - 1)^2 + w_2(\mathrm{Mw}(t_f)/\mathrm{Mw}_d - 1)^2 + w_3(X(t_f) - 1)^2 + w_4 t_f \quad (18)$$

where t_f is the batch ending time, Mn_d and Mw_d are the desired number average molecular weight and weight average molecular weight, respectively, X is the monomer conversion, and w_1 to w_4 are positive weighting factors. Let U denotes the batch recipe, then the optimum batch recipe and batch ending time can be found by solving the following optimisation problem:

$$\min_{U,\,t_f} J \quad (19)$$

In this study, the weights w_1 to w_3 are all selected as one and the weight w_4 is selected as 0.001 h^{-1}. Since the developed neural network models predict polymer qualities at 60, 80, 100, 120, 140, 160, and 180 min from the batch start, these time instance are considered as the possible batch ending time. Optimisation is then performed for each of these possible batch ending time. The optimal batch recipe and batch ending time are selected as these which minimise the objective function.

Consider the following example where Mn_d and Mw_d are 2.8×10^5 g/mol and 8×10^5 g/mol, respectively, corresponding to a particular grade of polymer product. Optimisation results for the seven possible batch ending times are presented in Table 1. Among these, the batch ending time of 120 min and its corresponding recipe give the smallest objective function value. The optimal recipe was used in simulation and the resulting trajectories of polymer quality variables are presented in Fig. 10. It can be seen from Fig. 10 that the final product has the desired quality. Table 1 also indicates that for the production of this particular grade of product, faster reactions can be achieved through higher temperature and lower initiator concentration.

The proposed technique can also be used for the monitoring of batch polymerisation reactors. Given a batch recipe, trajectories of polymer quality variables can be predicted. A batch ending time can be determined based on these predictions. When there exist reactive impurities, the nominal batch ending time will be inappropriate. Based on the estimated amount of reactive impurities, trajectories of polymer quality variables can be predicted. Process operators can then anticipate the final product qualities and determine a suitable batch ending time.

Table 1
Optimum batch recipes

t_f (min)	J	T_s (K)	I_0 (g)
60	0.0394	350.29	1.09
80	0.0380	344.86	1.52
100	0.0382	340.85	2.02
120	**0.0378**	**337.77**	**2.5**
140	0.0489	337.07	2.5
160	0.0719	336.45	2.5
180	0.0804	336.03	2.5

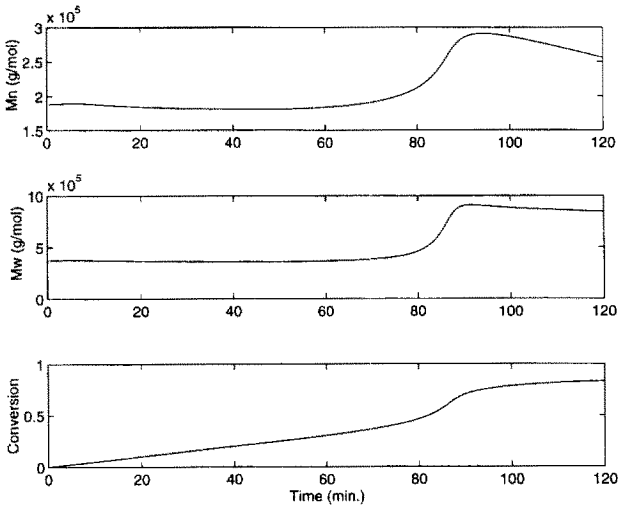

Fig. 10. Polymerisation trajectories under the optimum batch recipe.

7. Conclusions

Robust neural networks are used to predict trajectories of polymer quality variables in batch polymerisation from batch recipes. The robust neural network is obtained by developing several neural networks from bootstrap re-sampled training data and combining them. By this means, neural network generalisation capability can be significantly improved. The technique only requires a small amount of process operation data which can be accumulated during previous operations. By using neural networks to learn the relationship between batch recipes and polymerisation trajectories, the development of complex polymerisation kinetic models is avoided. The problem of numerical integration of a large number of complex differential equations over a long time period is also avoided.

The effect of disturbances, mainly in the form of reactive impurities, is considered in the proposed technique. Several conversion measurements are taken during the initial stage of a batch to estimate the amount of reactive impurities. A stacked neural network is used to estimate the amount of reactive impurities. From the estimated impurities, the effective amount of initial initiator can be calculated. Application of the proposed technique to a simulated batch MMA polymerisation reactor demonstrates that the technique can accurately predict trajectories of polymer quality variables even with the presence of reactive impurities. The technique has been used in designing optimum recipes for a batch poly-

merisation process. It can also be used for the prediction, control, and monitoring of batch polymerisation processes.

Acknowledgements

The work is supported by the European Community under the grant BRITE EURAM Project No. 7009.

References

[1] W.H. Ray, Can. J. Chem. Eng. 45 (1967) 356–360.
[2] S. Chen, W. Jeng, Chem. Eng. Sci. 33 (1978) 735–743.
[3] I.M. Thomas, C. Kiparissides, Can. J. Chem. Eng. 62 (1984) 284–291.
[4] S.R. Ponnuswamy, S.L. Shah, C. Kiparissides, Ind. Eng. Chem. Res. 26 (1987) 2229–2236.
[5] T. Takamatsu, S. Shioya, Y. Okada, Ind. Eng. Chem. Res. 27 (1988) 93–99.
[6] S.S. Jang, W.L. Yang, Chem. Eng. Sci. 44 (1989) 515–528.
[7] A.R. Secchi, E.L. Lima, J.C. Pinto, Polym. Eng. Sci. 30 (1990) 1209–1219.
[8] C. Kiparissides, Chem. Eng. Sci. 51 (1996) 1637–1659.
[9] G. Cybenko, Math. Control Signal Systems 2 (1989) 303–314.
[10] F. Girosi, T. Poggio, Biological Cybernetics 63 (1990) 169–179.
[11] J. Park, I.W. Sandberg, Neural Comput. 3 (1991) 246–257.
[12] N.V. Bhat, T.J. McAvoy, Comput. Chem. Eng. 14 (1990) 573–583.
[13] A.J. Morris, G.A. Montague, M.J. Willis, Trans. IChemE, Part A 72 (1994) 3–19.
[14] A.B. Bulsari (Ed.), Computer-Aided Chemical Engineering, Vol. 6: Neural Networks for Chemical Engineers, Elsevier, 1995.
[15] J.C. MacMurray, D.M. Himmelblau, Comput. Chem. Eng. 19 (1995) 1077–1088.
[16] A.Y.D. Tsen, S.S. Jang, D.S.H. Wong, B. Joseph, AIChE J. 42 (2) (1996) 455–465.
[17] C. Bishop, Neural Comput. 13 (1991) 579–588.
[18] J. Sietsma, R.J.F. Dow, Neural net pruning—why and how, Proceedings of IEEE International Conference on Neural Networks, 1988, pp. 325–333.
[19] J. Zhang, A.J. Morris, G.A. Montague, M.T. Tham, Dynamic system modelling using mixed node neural networks, Proceedings of IFAC Symposium ADCHEM'94, Kyoto, Japan, May 25–27, 1994, pp. 114–119.
[20] D.H. Wolpert, Neural Networks 5 (1992) 241–259.
[21] D.V. Sridhar, R.C. Seagrave, E.B. Bartlett, AIChE J. 42 (1996) 2529–2539.
[22] J. Zhang, E.B. Martin, A.J. Morris, C. Kiparissides, Comput. Chem. Eng. 21 (1997) s1025–s1030.
[23] J. Zhang, A.J. Morris, E.B. Martin, C. Kiparissides, Estimation of impurity and fouling in batch polymerisation reactors using stacked neural networks, Vol. 1, Proceedings of ACC97, Albuquerque, USA, June 4–6, 1997, pp. 247–251.
[24] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClell (Eds.), Parallel Distributed Processing, MIT Press, 1986.
[25] M.I. Jordan, R.A. Jacobs, Neural Comput. 6 (1994) 181–214.
[26] Y. Raviv, N. Intrator, Connection Sci. 8 (1996) 355–372.
[27] L. Breiman, Stacked regressions, Technical Report No. 367, Department of Statistics, University of California at Berkeley, USA, 1992.
[28] B. Efron, R. Tibshirani, An Introduction to Bootstrap, Chapman and Hall, London, 1993.
[29] R. Tibshirani, Neural Comput. 8 (1996) 152–163.
[30] T. Kent, Biometrika 69 (1982) 19–27.
[31] J.P. Congalidis, J.R. Richards, W.H. Ray, AIChE J. 35 (1989) 891–907.
[32] A. Penlidis, J.F. MacGregor, A.E. Hamielec, J. Appl. Polym. Sci. 35 (1988) 2023–2038.
[33] S.A. Mendoza-Bustos, A. Penlidis, W.R. Cluett, Ind. Eng. Chem. Res. 29 (1990) 82–89.
[34] D. Marquardt, SIAM J. Appl. Math. 11 (1963) 431–441.
[35] J. Sjoberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.Y. Glorennec, H. Hjalmarsson, A. Judisky, Automatica 31 (1995) 1691–1724.