Zhang J.

**Batch-to-batch optimal control of a batch polymerisation process based on stacked neural network models.**


**Copyright:**

NOTICE: This is the authors' version of a work that was accepted for publication in *Chemical Engineering Science*. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in *Chemical Engineering Science*, 63(5), March 2008

DOI: 10.1016/j.ces.2007.07.047

Always use the definitive version when citing.

**Further information on publisher website:** [http://www.elsevier.com/](http://www.elsevier.com/)

**Date deposited:** 10th June 2014

**Version of article:** Author accepted manuscript

This work is licensed under a [Creative Commons Attribution-NonCommercial 3.0 Unported License](http://creativecommons.org/licenses/by-nc/3.0/)

*ePrints – Newcastle University ePrints*

[http://eprint.ncl.ac.uk](http://eprint.ncl.ac.uk)
Batch-to-Batch Optimal Control of a Batch Polymerisation Process based on Stacked Neural Network Models

Jie Zhang
School of Chemical Engineering & Advanced Materials
Newcastle University, Newcastle upon Tyne NE1 7RU, U. K.
E-mail: jie.zhang@newcastle.ac.uk

Abstract
A neural network based batch to batch optimal control strategy is proposed in this paper. In order to overcome the difficulty in developing mechanistic models for batch processes, stacked neural network models are developed from process operational data. Stacked neural networks have enhanced model generalisation capability and can also provide model prediction confidence bounds. However, the optimal control policy calculated based on a neural network model may not be optimal when applied to the true process due to model plant mismatches and the presence of unknown disturbances. Due to the repetitive nature of batch processes, it is possible to improve the operation of the next batch using the information of the current and previous batch runs. A batch to batch optimal control strategy based on the linearization of stacked neural network model is proposed in this paper. Applications to a simulated batch polymerisation reactor demonstrate that the proposed method can improve process performance from batch to batch in the presence of model plant mismatches and unknown disturbances.

Keywords: Batch processes, neural networks, polymerisation, run to run control, optimisation, process control, iterative learning control.

1. Introduction
Batch processes are suitable for the responsive manufacturing of high value added products. A typical feature of batch processes is that the same process unit is used in the manufacturing of
different products and each product is of relatively small amount and high added value (Bonvin, 1998; Ruppen et al., 1995). Optimal control of batch processes is therefore of great importance. The core of optimal control is an accurate process model. Developing mechanistic models usually requires significant amount of time and effort. This may not be feasible for batch processes where frequent changes in product specifications occur and a type of product is usually manufactured for a limited time period in response to the dynamic market demand. Data based empirical modelling can be a very useful alternative in this case. Neural networks have been shown to be capable of approximating any continuous nonlinear functions (Cybenko, 1989; Girosi and Poggio, 1990; Park and Sandberg, 1991) and have been applied to nonlinear process modelling (Bhat and McAvoy, 1990; Bulsari, 1995; Narendra and Parthasarathy, 1990; Su et al., 1992).

In order to build accurate neural network models, ideally a large amount of data covering different process operating conditions should be used in network training. In batch processes, data for building neural network models are usually not abundant due to limited batch runs in the manufacturing of a particular product. Thus the neural network model developed is only an approximation of the modelled batch process and model plant mismatches are unavoidable. Due to the model plant mismatches, the optimal control policy calculated on the model may not be optimal when applied to the actual process.

Due to the repetitive nature of batch process operations, it would be possible to improve the operation of the next batch using the information of the current and previous batch runs. Various run-to-run control strategies for the product quality have been proposed in the literatures. The operating policy can be optimised by run-to-run optimising control for the final product quality in order to address the problems of model-plant mismatches and/or unmeasured disturbances in batch.
processes (Clarke-Pringle and MacGregor, 1998; Doyle et al., 2003). Lee and co-workers (Chae et al., 2000; Chin et al., 2000; Lee et al., 2000) propose the quadratic criterion-based ILC approach for tracking control for temperature of batch processes based on a linear time-varying tracking error transition model. This paper presents a neural network model based batch to batch optimal control strategy. The neural network model is linearised around the current batch. Based on this linearised model, the control policy for the next batch is modified to minimise the control errors at the end of the next batch. The procedure is repeated from batch to batch.

The paper is organised as follows. Section 2 presents neural network modelling of batch processes. A neural network model based batch to batch optimal control strategy is presented in Section 3. Section 4 presents an application to a simulated batch polymerisation reactor. The last section concludes this paper.

2. Neural Network Modelling of Batch Processes

2.1 Bootstrap Aggregated Neural Networks

A limitation of single neural network models is that they can lack generalisation when applied to unseen data, i.e. the trained neural network gives good performance on the training data but gives unsatisfactory performance on unseen data which is not used in the training process. Over the past years several techniques have been developed to improve neural network generalisation capability, such as regularisation (Bishop, 1991), early stopping (Bishop, 1995), Bayesian learning (MacKay, 1992), training with both dynamic and static process data (Zhang, 2001), and combining multiple networks (Sridhar et al., 1996; Wolpert, 1992; Zhang et al., 1997). In training with regularisation, the magnitude of network weight is introduced as
a penalty term in the training objective function and unnecessarily large network weights are avoided. In training with early stopping, neural network performance on the testing data is checked during the training process and the training process stops when the neural network prediction errors on the testing data start to increase. Among these techniques, combining multiple networks is a very promising approach to improving model predictions on unseen data. The emphasis of this approach is on generalisation accuracy on future predictions (i.e. predictions on unseen data). When building neural network models, it is quite possible that different networks perform well in different regions of the input space. By combining multiple neural networks, prediction accuracy on the entire input space could be improved. Stacked neural networks have been successfully used for the inferential estimation of polymer quality (Zhang et al., 1997), prediction of final product quality (Zhang et al., 1998), and estimation of reactive impurities and reactor fouling (Zhang et al., 1999) in a batch polymerisation process.

A diagram of bootstrap aggregated neural networks is shown in Figure 1, where several neural network models are developed to model the same relationship. Instead of selecting a “best” single neural network model, these individual neural networks are combined together to improve model accuracy and robustness. The overall output of the aggregated neural network is a weighted combination of the individual neural network outputs. This can be represented by the following equation.

\[
 f(X) = \sum_{i=1}^{n} w_i f_i(X) 
\]

(1)

where \( f(X) \) is the aggregated neural network predictor, \( f_i(X) \) is the \( i \)th neural network, \( w_i \) is the aggregating weight for combining the \( i \)th neural network, \( n \) is the number of neural networks,
and $X$ is a vector of neural network inputs. Proper determination of the stacking weights is essential for good modelling performance. A popular choice of stacking weights is simple averaging, i.e. the stacked neural network output is an average of the individual network outputs. Perrone and Cooper (1993) show that combining $n$ independent predictors by simple averaging can reduce the mean squared prediction errors by a factor of $n$. This result is interesting although the individual models are generally not independent. An implication of this result is that significant improvement in model prediction can be obtained if dissimilar models are combined. Since the individual neural networks are highly correlated, appropriate stacking weights could be obtained through principal component regression (PCR) (Zhang et al., 1997). Instead of using constant stacking weights, the stacking weights can also dynamically change with the model inputs (Ahmad and Zhang, 2005; 2006).

Another advantage of bootstrap aggregated neural network is that model prediction confidence bounds can be calculated from individual network predictions (Zhang et al., 1999). The standard error of the $i$th predicted value is estimated as

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

where $y(x_i; \cdot) = \sum_{b=1}^{n} y(x_i; W^b)/n$ and $n$ is the number of neural networks in a stacked neural network. Assuming that the individual network prediction errors are normally distributed, the 95% prediction confidence bounds can be calculated as $y(x_i; \cdot) \pm 1.96\sigma_e$. A narrower confidence bound, i.e. smaller $\sigma_e$, indicates that the associated model prediction is more reliable.
Training a stacked neural network containing \( n \) individual networks requires the training of the \( n \) individual networks and, thus, requires \( n \) times of computation effort than training a single neural network. However, with the current computer technology, computation time is no longer an issue in training feed forward neural networks. Thus, the enhanced generalisation capability of stacked neural networks far out weight the extra training time required.

2.2 Neural Network Modelling of Batch Processes

Unlike continuous processes, which are typically operated at or near some optimal steady state conditions, batch processes are operated in transient states (Ruppen et al., 1995). Thus an accurate batch process model should be a nonlinear dynamic model. Neural networks are suitable for modelling nonlinear dynamics and can be used for building empirical models for batch processes.

A neural network based dynamic model is typically of the following form:

\[
y(t) = f[y(t-1), y(t-2), ..., y(t-m_o), u(t-1), u(t-2), ..., u(t-m_i)]
\]  

(3)

where \( y \) is the process output, \( u \) is the process input, \( t \) represents the discrete time, \( m_o \) and \( m_i \) are the time lags in the model output and input respectively. To build this type of model, process input and output data are required. In batch process optimal control, the process output variables of interest are typically product quality variables, which are usually difficult to measure. Thus collecting a large amount of product quality measurements is generally difficult.

In batch processes, the ultimate interest lies in the end of batch product quality. To calculate the optimal control policy for a batch process, the model utilised in the optimisation
calculation should be able to provide accurate long range predictions (Tian et al., 2001). When Eq(3) is used for long range predictions, the model predictions are recursively fed back to the model as input to predict the process output at further prediction horizons. In doing so the prediction errors could be accumulated unless recurrent neural networks are used in modelling the batch processes.

To address the above two issues, this paper proposes to use the following model to model product quality variables at several fixed time points of a batch, including the batch ending point.

$$Y = f(X_0, U) \tag{4}$$

where $Y = [y(t_1) y(t_2) ... y(t_f)]^T$ is a matrix of product quality variables (or a vector of product quality variables) at times $t_1$ to $t_f$, with $t_f$ being the batch end time, $X_0$ is the batch process initial condition, and $U = [u_1 u_2 ... u_N]^T$ is a vector of control actions. The above nonlinear function $f()$ is represented using a stacked neural network with the network weights trained using process operational data.

### 3. Batch-to-Batch Optimal Control

Based on the model Eq(4), the optimal control policy $U$ can be obtained by solving the following optimisation problem:

$$\min_U J[Y(t_f)] \tag{5}$$

s.t. product quality and process constraints
However, model plant mismatches always exist, especially when the model is a data based empirical model developed from a limited amount of process operational data. In this case, the optimal control policy calculated from Eq(5) is only optimal on the model and may not be optimal when applied to the real process. Furthermore, there always exist disturbances such as raw material variations, reactive impurities, and reactor fouling. The presence of such disturbances can also deteriorate the control performance of the previously obtained optimal control policy.

To limit the deterioration of control performance due to model plant mismatches and unknown disturbances, a batch to batch optimal control strategy is proposed in this paper. This control strategy utilises the information of the current and previous batch runs to enhance the operation of the next batch.

The first order Taylor series expansion of Eq(4) around a nominal control profile can be expressed as

\[ \hat{y}(t_f) = f_0 + \frac{\partial f}{\partial u_1} \Delta u_1 + \frac{\partial f}{\partial u_2} \Delta u_2 + \cdots + \frac{\partial f}{\partial u_N} \Delta u_N \]  

(6)

where \( \Delta u_1 \) to \( \Delta u_N \) are deviations in the control profile from the nominal control profile.

For the \( k \)th batch, the actual product quality can be written as the model prediction plus an error term

\[ y_k(t_f) = \hat{y}_k(t_f) + e_k \]  

(7)

where \( y_k(t_f) \) and \( \hat{y}_k(t_f) \) are the actual and predicted product quality values at the end of a batch, and \( e_k \) is the model prediction error.
The prediction for the \((k+1)^{th}\) batch can be approximated using the first order Taylor series expansion based on the \(k^{th}\) batch

\[
\hat{y}_{k+1}(t_f) = \hat{y}_k(t_f) + \left. \frac{\partial f}{\partial u_1} \right|_{u_i} (u_1^{k+1} - u_1^k) + \left. \frac{\partial f}{\partial u_N} \right|_{u_i} (u_N^{k+1} - u_N^k)
\]

\[
= \hat{y}_k(t_f) + G^T \Delta U^{k+1}
\] (8)

where

\[
\Delta U^{k+1} = [\Delta u_1^{k+1} \Delta u_2^{k+1} \ldots \Delta u_N^{k+1}]^T
\]

\[
G^T = \left[ \left. \frac{\partial f}{\partial u_1} \right|_{u_i} \left. \frac{\partial f}{\partial u_2} \right|_{u_i} \ldots \left. \frac{\partial f}{\partial u_N} \right|_{u_i} \right]^T.
\]

Assuming that the model prediction errors for the \(k^{th}\) batch and the \((k+1)^{th}\) batch are the same, then optimal control of the \((k+1)^{th}\) batch can be represented as

\[
\min_{\Delta U^{k+1}} J = \left\| \hat{y}_k(t_f) + G^T \Delta U^{k+1} + e_k - y_d \right\|_Q^2 + \left\| \Delta U^{k+1} \right\|_R^2
\]

where \(y_d\) is a vector of desired product quality, \(Q\) is a weighting matrix for the product quality control errors and \(R\) is a weighting matrix for the control effort.

Set \(\frac{\partial J}{\partial \Delta U^{k+1}} = 0\), the optimal control updating can be calculated as

\[
\Delta U^{k+1} = (GQG^T + R)^{-1} GQ[y_d - \hat{y}_k(t_f) - e_k]
\]

\[
U^{k+1} = U^k + \Delta U^{k+1}
\] (10)

(11)

For a neural network model, the gradient of model output with respect to the control \(U\), \(G\), can be calculated analytically. If a single hidden layer feed forward neural network is used, hidden layer neurons use the sigmoidal activation function, and output layer neurons use the linear
activation function, then the element on the $i^{th}$ row and $j^{th}$ column of $G$, $G_{ij}$, can be calculated as

$$G_{ij} = \frac{\partial y_i}{\partial u_j} = \sum_{k=1}^{nh} \frac{\partial y_i}{\partial O_k} \frac{\partial O_k}{\partial u_j} = \sum_{k=1}^{nh} W_{2}^{k,i} O_k (1 - O_k) W_{1}^{j,k}$$  \hspace{1cm} (12)$$

where $nh$ is the number of hidden neurons, $O_k$ is the output of the $k^{th}$ hidden neuron, $W_{2}^{k,i}$ is the output layer weight from the $k^{th}$ hidden neuron to the $i^{th}$ output layer neuron, and $W_{1}^{j,k}$ is the output layer weight from the $j^{th}$ input to the $k^{th}$ hidden neuron.

If a stacked neural network containing $n$ single hidden layer neural networks is used, then the element on the $i^{th}$ row and $j^{th}$ column of $G$, $G_{ij}$, can be calculated as

$$G_{ij} = \sum_{l=1}^{n} w_l G_{ij}^l$$  \hspace{1cm} (13)$$

where $w_l$ is the weight for combining the $l$th network and $G_{ij}^l$ is contribution from the $l$th network calculated using Eq(12).

Note that the proposed technique can also be applied with a mechanistic model or a hybrid model. In such cases, the linearised gain matrix $G$ in Eq(8) to Eq(11) would generally be obtained numerically.

4. Application to a Simulated Batch Polymerisation Reactor

4.1 A batch polymerisation reactor

The batch polymerisation reactor studied in this paper is a simulated process based on a pilot scale polymerisation reactor developed in the Department of Chemical Engineering, Aristotle
University of Thessaloniki, Greece. The batch polymerisation reactor is shown in Figure 2. The free-radical solution polymerisation of methyl methacrylate (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. By circulating water at appropriate temperature through the reactor jacket, heating and cooling of the reaction mixture is achieved. 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 set-point of the two secondary controllers. 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 general description of the reactions during the free radical solution polymerisation of MMA initiated by benzoyl peroxide is given in (Achilias and Kiparissides, 1992; Penlidis et al. 1992). A detailed mathematical model covering reaction kinetics and heat and mass balances was developed and validated using the real reactor operation data (Achilias and Kiparissides, 1992). Based on this model, a rigorous simulation programme was developed and used to generate polymerisation data under different batch operating conditions. These data were used to build and validate neural network based inferential estimation models.

4.2 Modelling of the batch polymerisation reactor using neural network

Since the number average and weight average molecular weights are usually difficult to measure, it is typical that only a small number of samples of the molecular weights are
collected during reactor operation. Thus, each batch only provides a limited data set for building a neural network model. When building neural network models for a batch polymerisation reactor, these practical considerations should be taken into account if reliable models (fit for purpose) are to be built.

In this study, we consider the following modelling and control scheme. The nominal batch time for this reactor is about 180 minutes. Samples of the monomer conversion and the number average and weight average molecular weights are collected from 60 minutes at a 20 minutes interval. Thus during a batch up to 7 samples of molecular weights are collected. The polymer quality variables, the number average and weight average molecular weights, are measured sparsely because they are generally difficult to be measured on-line. The control variables considered here are the initial reactor temperature setpoint and the reactor temperature setpoints at 40, 60, 80, 100, 120, 140, and 160 minutes. These reactor temperature setpoints provide a piecewise constant control trajectory for the reactor.

A neural network model for predicting polymer quality variables at time $t_N$ is then of the following form:

$$Y(t_N) = f(I_0, U(t_N))$$

(14)

where

$$Y(t_N) = [Mn(t_N) \ Mw(t_N) \ Xc(t_N)]^T$$

$$U(t_N) = [T_{sp0} \ T_{sp1} \ T_{sp2} \ ... \ T_{spN}]^T$$

In the above equations, $T_{sp0}$ to $T_{spN}$ are the trajectory of reactor temperature setpoints, $I_0$ is the initial initiator mass, $Xc(t_N)$, $Mn(t_N)$, and $Mw(t_N)$ are the monomer conversion, the number average molecular weight, and weight average molecular weight at time $t_N$ respectively.
In order to “simulate” the building of neural network models in an industrial environment, 50 batches were simulated with controls generated from Monte-Carlo simulation. The sampled data were corrupted with white noise to represent the effects of measurement noises. From the generated data, bootstrap re-sampling with replacement (Efron, 1982) was used to generate 30 replications of the data. In generating a replication of this data set, a sample (a batch) is randomly taken from the 50 batches. This sample is copied and put back to the original data set. This is repeated until a replication of 50 samples has been generated. For each re-sampled data set (a replication of the original data), 7 neural network models in the form of Eq(3) for predicting polymer quality at the 7 discrete time points were developed. Each of the neural network contains 10 hidden neurons and the network weights were initialised as random numbers in the range (-0.1, 0.1). The networks were trained using the Levenberg-Marquardt optimisation algorithm with regularisation and cross-validation based “early-stopping”. Due to the different magnitudes of the model input and output data, the data for neural network training have to be scaled first. In this study, $M_n$ is scaled down by a factor of $10^5$, $M_w$ is scaled down by a factor of $10^6$, $I_0$ is scaled down by a factor of 3, and the reactor temperature setpoint is scaled through $(T_{sp} - 338)/20$. The 30 individual networks for predicting polymer quality at a particular time $t_N$ were then combined together through PCR. A further 20 batches were simulated to generate a set of unseen data to validate the developed neural network models.

Figure 3 shows the scaled sum of squared errors (SSE) of the individual networks on training and validation data sets. It can be observed that the performances of these networks on the training and validation data sets are not consistent. A network having small SSE on the
training data set may have quite large SSE on the validation set. This indicates the non-robust nature of a single neural network model. The minimum SSEs of individual networks on the training and validation data sets are 18.0 and 19.0 respectively. The SSEs from the aggregated network on the training and validation data sets are 9.8 and 13.8 respectively. Thus the model accuracy is significantly improved by combining multiple imperfect models.

4.3 Batch to batch iterative optimal control

In this study, the desired final product quality is selected as \( y_d = [2\times10^5 \ 5\times10^5 \ 1]^T \), the weighting matrix \( Q \) is selected as \( Q = \text{diag}(2\times10^{-5}, \ 8\times10^{-6}, \ 4) \), and the weighting matrix \( R \) is selected as \( R = \text{diag}(0.1, \ 0.1, \ 0.1, \ 0.1) \). The following constraint is also considered: reactor temperature is bounded between 320 K and 360 K.

Based on the neural network model, optimal control policy was first calculated off-line. Since the neural network model can predict polymer quality variables at 60, 80, 100, 120, 140, 160, and 180 minutes, each of these time instances is considered as a possible batch ending time during optimisation and the one that leads to the best performance is selected. In this study, a batch ending time of 100 minutes gives the best performance. When this optimal control policy is applied to the process (i.e. mechanistic model simulation), the final product quality variables are: \( Mn(t_f) = 1.3898\times10^5 \text{ g/mol} \), \( Mw(t_f) = 5.1655\times10^5 \text{ g/mol} \), and \( Xc(t_f) = 0.8924 \). It can be seen that there exist significant differences between the actual performance and desired performance. This is due to model plant mismatches of the neural network model.

For comparison purpose, the proposed batch to batch optimal control method were implemented on the stacked neural network and three “good” single neural networks (network
No. 1, 11, and 25). It can be seen from Figure 3 that these three single networks give the best performance among the 30 individual networks. With the implementation of the proposed batch to batch optimal control method, it can be seen from Figure 4 that the control performance is improved from batch to batch under the stacked neural network model. The improvement is significant during the first 4 batches. After the 6th batch, the control performance is still improving from batch to batch, but the magnitude of improvement is not very significant indicating the batch to batch optimal control strategy has nearly converged. The final product quality variables at the 10th batch are: $M_n(t_f) = 1.8132 \times 10^5$ g/mol, $M_w(t_f) = 5.0819 \times 10^5$ g/mol, and $X_c(t_f) = 0.8588$. These are very close to the desired final product quality. This demonstrates that the proposed batch to batch optimisation technique can effectively overcome the problem of model plant mismatches. Although batch to batch control based on the three single networks also in general improve the process operation from batch to batch, they give quite difference performance. While performance under network No. 1 is comparable to that under the stacked network, the sum of squared control errors under networks No. 11 and No. 25 at the 10th batch are much larger than that under the stacked network. This indicates the non-robust nature and unreliability of single neural networks. In order to show the robustness of stacked neural network models, the proposed batch to batch control strategy is also implemented on two other stacked neural network models: stacking 20 networks and stacking 25 networks. Studies in Zhang (1999) show that the performance of stacked neural networks typically stabilises after stacking about 10 to 20 networks. Figure 5 shows the batch to batch control performance based on the three stacked neural networks. It can be seen that the three stacked neural networks give similar control performance, demonstrating the reliability of stacked neural network models. This is in contrast to the performance of single neural networks shown in Figure 4.
To test the performance of the proposed control strategy under unknown disturbances, the amount of initial initiator was reduced from its nominal value of 2.5 g to 2.0 g from the 11th batch forward to simulate the presence of 0.5 g of reactive impurities. Reactive impurities commonly exist in industrial polymerisation processes (Penlidis et al., 1988; Kiparissides, 1996; Zhang et al., 1999). Reactive impurities in polymerisation processes are typically traces of inhibitors or oxygen. The studies of Penlidis et al. (1988) show that reactive impurities in an emulsion polymerisation process rapidly consume reactive free radicals, thus preventing particle generation and decreasing the growth of polymer particles already present in the emulsion. The effect of reactive impurities can usually be represented by a step decrease in the initial initiator concentration.

Due to the presence of this unknown amount of reactive impurities, the optimal control profile at the 10th batch is no longer optimal and the sum of squared control error increased sharply at the 11th batch as shown in Figure 6. The final product quality variables at the 11th batch with the control policy from stacking 30 networks are: \( Mn(t_f) = 2.2521 \times 10^5 \) g/mol, \( Mw(t_f) = 6.3507 \times 10^5 \) g/mol, and \( Xc(t_f) = 0.8590 \), which are much worse than those at the 10th batch. With the implementation of the batch to batch optimal control strategy, the control performance is significantly improved from batch to batch. At the 20th batch, the batch to batch optimal control almost converged with the final product quality variables as \( Mn(t_f) = 1.9358 \times 10^5 \) g/mol, \( Mw(t_f) = 5.7465 \times 10^5 \) g/mol, and \( Xc(t_f) = 0.8816 \). The results demonstrate that the batch to batch iterative optimal control strategy can effectively overcome the problems cause by the presence of an unknown amount of disturbance. Figure 6 also indicates...
that the control policies under the three stacked neural network models give similar performance. This demonstrates the reliability of stacked neural network models.

Figure 7 shows the final product quality variable variations from batch to batch, while Figure 8 shows the optimal control policies during iterative optimisation under the stacked neural network model. It can be seen that overall the product quality variables converge to their desired values despite the presence of model plant mismatches and unknown disturbances. Figure 7 indicates that the monomer conversion becomes worse in the first a few batches. However, the improvements in the number average molecular weight and weight average molecular weights far out weight this. By selecting the values for the Q matrix one can specify which quality variable is more important. Performance of the proposed control technique could be further improved if the stacked neural network model were updated when further batch operational data were available. The individual networks can be retrained and/or the stacking weight in Eq(1) can be re-estimated when more batches become available.

5. Conclusions

A neural network model based batch to batch iterative optimal control strategy is proposed in this paper. To avoid the difficulty in developing detailed mechanistic models, neural networks are used to model batch processes from process operational data. Due to model plant mismatches and the presence of unknown disturbances, an optimal control policy calculated based on a neural network model may not give optimal performance when applied to the real processes. The repetitive nature of batch processes allows the information of previous batch runs being used to update the control profile of the current batch. Linearization of a stacked neural network model is used in calculating the updated optimal control profile. Applications
to a simulated batch polymerisation process demonstrate that the proposed method can effectively overcome the problems of model plant mismatches and unknown disturbances through batch to batch updating of control profile.

Acknowledgements

The author thanks Prof. C. Kiparissides of Aristotle University of Thessaloniki, Greece, for providing the polymerisation simulation program.

Nomenclature

e : prediction errors
f : function
G : linearised gain matrix
I₀ : initial initiator mass
J : optimisation objective function
Mₙ : number average molecular weight
Mₘ : weight average molecular weight
mᵢ : time lag in model input
mₒ : time lag in model output
n : number of networks
Oₖ : output of the kᵗʰ hidden neuron
Q, R : weighting matrices
t : discrete time
tᵢ : batch time
U : a vector of control actions
u : process input
W : matrix of neural network weights
w : aggregating weights
X : neural network inputs
Xₑ : monomer conversion
$X_0$ : batch process initial condition
$Y$ : a matrix of product quality variables
$y$ : process output
$\hat{y}$ : model prediction
$y_d$ : desired final product quality
$\sigma_e$ : standard error
$\Delta$ : deviations

References


Figure 1. A bootstrap aggregated neural network

Figure 2. A batch polymerisation reactor (CW: cold water; HW: hot water; Tsp: Temperature setpoint; TT: temperature transducer; TC: temperature controller)
Figure 3. Model errors of individual networks

Figure 4. Sum of squared control errors
Figure 5. Sum of squared control errors based on three stacked neural network models

Figure 6. Sum of squared control errors under reactive impurities
Figure 7. Product quality variables

Figure 8. Control policies