Thermal Management in Laminated Die Systems Using Neural Networks

by

Jaho Seo

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical Engineering

Waterloo, Ontario, Canada, 2011

Author's declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The thermal control of a die is crucial for the development of high efficiency injection moulds. For successful thermal management, this research provides an effective control strategy to find sensor locations, identify thermal dynamic models, and design controllers. By applying a clustering method and sensitivity analysis, sensor locations are identified. The neural network and finite element analysis techniques enable the modeling to deal with various cycle-times for the moulding process and uncertain dynamics of a die. A combination of off-line training through finite element analysis and training using on-line learning algorithms and experimental data is used for the system identification. Based on the system identification which is experimentally validated using a real system, controllers are designed using fuzzy-logic and self-adaptive PID methods with backpropagation (BP) and radial basis function (RBF) neural networks to tune control parameters. Direct adaptive inverse control and additive feedforward control by adding direct adaptive inverse control to self-adaptive PID controllers are also provided. Through a comparative study, each controller's performance is verified in terms of response time and tracking accuracy under different moulding processes with multiple cycle-times. Additionally, the improved cooling effectiveness of the conformal cooling channel designed in this study is presented by comparing with a conventional straight channel.

Acknowledgements

I would like to express my gratitude to my supervisors, Dr. Amir Khajepour and Dr. Jan P. Huissoon for creative advice, sincere guidance and encouragement which have provided me with the best environment for my research.

I would like to acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC) for financial support and the Government of the Province of Ontario for the Ontario Graduate Scholarship (OGS).

I should be grateful to Dr. William Melek, Dr. Cecile Devaud, Dr. David Wang, and Dr. Rickey Dubay for serving as my examining committee members and an external examiner, respectively.

I would also like to thank Morvaid Karimi, Yaser Shanjani, Hossein Ahari and all members of Dr. Khajepour's Research Group for their great consideration.

I gratefully acknowledge Mr. Dean Dajko from WebPlas Inc. (http://www.webplas.ca), Jason Benninger and Martha Morales from University of Waterloo for their technical support.

I equally give thanks for help and friendship to members of the UW KGSA (University of Waterloo Korean Graduate Student Association) including my best friends, Sanghyun Lee, Hyoun-Tae Hwang, Myung-Gwang Kim and Dr. Sung-Wook Jeen.

None of achievements would have been possible without unconditional love, devotion, patience and prayer of my wife, Yunhee Lee. So I want to express my extreme gratitude to her as well as daughters, Marie and Ellie who are my treasures.

In addition, I deeply acknowledge my parents, parents-in-law, brothers-in-law, sisters and brother for their appreciation, financial and spiritual support and trust.

Finally, I am sincerely grateful to my Lord Jesus Christ, my dear Fathers, Sisters and community members (especially brother Mr. Ock-Hyun Baek, Dr. Wan-Ho Song, Dr. Young-Jin Park, Dr. Hyun-Tae (Joseph) Kim and Dr. Soo Jeon) of the Korean Community Catholic Churches in Kitchener and Montreal.

Table of Contents

List of Figures

List of Tables

Nomenclature

$B(i,k)$	Average distance from the i^{th} node to nodes in another cluster k
b_j	Width of Gaussian kernel function (h_j)
b_0	Neuron bias
b_{h0}^{w}	Bias of hidden layer
b_{i0}^{v}	Bias of output layer
C_j	Center of Gaussian kernel function (h_j)
Ct	Cycle-time
$c_{en\,j}$	Cluster centroid in K-means algorithm
DT	Desired average of four cycle average temperature (T_a)
E	Sum square of error
E_c	Network cumulative error (sum of E)
ET	Error between actual temperature and desired value
∇E_c	Gradient of E_c
e	Error between system output and neural network output
H	Output vector in hidden layer of RBF neural network
h_j	Gaussian kernel function
I	Identity matrix
J	Objective (cost) function
J_c	Jacobian matrix
K_D	Derivative gain of PID control
K_I	Integral gain of PID control
K_P	Proportional gain of PID control

$\hat{K}$	Gain matrix
$\tilde{K}$	Normalized gain matrix
M	Total number of samples during one cycle
n_u	Number of past inputs in NARX
n_y	Number of past outputs in NARX
$net_l^{(h)}$	Inputs in hidden layer of neural network with BP learning
$net_r^{(o)}$	Inputs in output layer of neural network with BP learning
o	Neuron output
o_i	Outputs in output layer (neural network output)
$o_j^{(i)}$	Outputs in input layer of neural network with BP learning
$o_l^{(h)}$	Outputs in hidden layer of neural network with BP learning
$o_r^{(o)}$	Outputs in output layer of neural network with BP learning
P	Orthonormal loading matrix
$P_{x,y}$	Pearson correlation coefficient
Q	Flow rate of coolant
$q^{(n)}$	Lipschitz quotient index
q_{ij}	Lipschitz quotient
r	Desired value of average of four T_a
$r_{\max}$	Maximum of temperature ratio between two nodes
r_{mean}	Mean of temperature ratio between two nodes
$r_{\min}$	Minimum of temperature ratio between two nodes
$s(i)$	Silhouette coefficient value
T	Temperature

T_a	Cycle average temperature
T_n	Temperature at the n^{th} sample time
T_s	Score matrix
Td	Temperature deviation
Tol	Tolerance rate
t_i	Target output
tn	Interval time
u	System input
$u_{Nor,m}$	Normalized value of the m^{th} input , u_m
v_{ih}	Weight of output layer
W	Weight vector between hidden and output layers of RBF neural network
$w(i)$	Average distance from the i^{th} node to the other nodes in its own cluster
w_{hj}	Weight of hidden layer
w_j	Neuron weight
$w_{lj}^{(h)}$	Weights in hidden layer of neural network with BP learning
$w_{rl}^{(o)}$	Weights in output layer of neural network with BP learning
$w^{(p)}$	Weight interacting between layer p and previous layer $p-1$
X	Input vector in hidden layer of RBF neural network
$X_{i,m}$	Sensitivity coefficient
x_i	Data point in K-means algorithm
x_j	Neuron input
y	System output
y_N	Neural network output

$y_{Nor,\,i}$	Normalized value of output at the i^{th} node
z_h	Outputs in hidden layer
α^*	Momentum factor in RBF neural network
β^*	Momentum factor in RBF neural network
γ	Momentum constant (factor)
δ_i	Error signal of i^{th} neuron of output layer
δ_h	Error signal of h^{th} neuron of hidden layer
η	Learning (speed) rate
η^*	Leaning speed rate in RBF neural network
η^{**}	Leaning speed rate for PID parameter update in RBF neural network
η^{op}	Optimized learning rate
μ	Learning parameter
σ	Sigmoid function
σ_h	Sigmoid function of hidden layer
σ_o	Sigmoid function of output layer

Chapter 1
Introduction

1.1 Laminated die systems with conformal cooling channels

Die casting is a manufacturing process in which molten metal under high pressure is forced into a die. Since complex shapes with a high degree of accuracy and repeatability can be fabricated with this process, die casting is widely used in industrial applications in which high dimensional accuracy is required [1].

Die fabrication technologies have been developed to provide tool flexibility and cost saving (in production and change of tools) [2]. Laminated die fabrication (Figure 1-1) is a rapid prototyping technique which enables the creation of high efficiency dies. In this technique, the die is fabricated by thin plates on top of each other rather than the traditional approach of machining the die from a solid block [3]. The direction and thickness of each layer in the die can be optimized to achieve the desired accuracy [4].

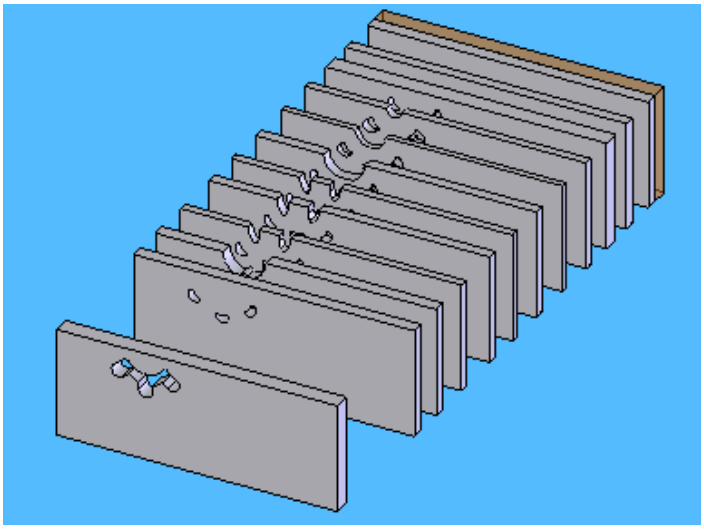


Figure 1-1 Laminating process

The laminating process also permits the inclusion of conformal cooling channels for more efficient cooling in injection moulds. This is different from conventional cooling channels (which consist of straight drilled holes into the mould) in that conformal cooling channels can be curved to follow the surface of the mould cavity [5]. The use of conformal cooling channels contributes to reduced cycle time and internal stresses (and consequently part distortion) by enhancing heat transfer and heat dissipation whereas the conventional cooling system may result in non-uniform cooling with undesirable effects on a part's physical properties [4] (this is demonstrated through simulation results in Chapter 4).

1.2 Thermal management in laminated die systems

There are many variables in the development of high efficiency casting or moulding dies such as die temperature, cooling rate, metal injection velocity, casting pressure and part geometry that need to be considered [6]. Among these, thermal control of a die is a key issue because of its effects on production quality and rate. For example, improper thermal control can lead to a number of defects in the part such as shrink porosity, poor fill and stuck castings [7]. Inaccurate die temperature (i.e., too high temperature) prolongs the cycle time by making solidification of the die casting longer [8]. Another benefit of a successful thermal control system is to minimize scrap and increase casting production rates through more uniform temperature in the die [9]. However, a die is a complex continuous system with cooling and heating channels which cause the overall control to be quite complicated.

Conventional approaches to thermal management of dies have been based on trial and error, the know-how of operators and rules of thumb. Thus, a more systematic approach to manage the thermal control of dies is required. According to the literature, most existing approaches to address die thermal control are limited to commercial industrial controllers such as conventional PID.

1.3 Research objectives

To overcome limitations of conventional thermal control approaches and to deal with uncertain dynamics of laminated die systems in the injection moulding process, the principal objective in this research is to identify a thermal dynamic model and develop a controller for temperature management of laminated die systems, using neural networks. The neural network method is an effective approach

in uncertain systems for which it is difficult to obtain a mathematical model. At the same time, this study will address the following specific objectives:

- Determination of optimum sensor locations and the required number of sensors that enables accurate state (temperature distribution) estimation using temperature-ratio based clustering and sensitivity analysis.
- Verification of conformal cooling channels as an effective alternative to conventional cooling channels in terms of uniform temperature distribution and reduction of cooling time.
- Modeling of die thermal dynamics considering uncertainties and cycle-times in plastic injection moulding.
- Conducting both off-line and on-line training to improve model estimation using finite element analysis and experimental data.
- Development of an efficient thermal controller based on the conducted system identification and fuzzy-neural techniques for better control of cavity-wall temperature and thus product quality.

The successful outcomes of this study will greatly contribute to improving the injection moulding process by providing the industry with practical guidance for thermal management.

1.4 Thesis organization

This thesis is organized as follows:

In the second chapter, a literature review describes conventional control strategies for thermal control in die systems, introduction of neural networks to die systems, neural network learning algorithms for system identification, neural control features and existing literature for optimum sensor location and number.

In the third chapter, the laminated die considered in this study and finite element analysis of the die are introduced. Methodologies such as clustering and sensitivity analysis for identifying optimal sensor locations are also presented. Then, sensor locations identified from different methods are evaluated from various aspects.

In Chapter 4, the cooling performance of conformal channels in the laminated die is compared to that of conventional channels with respect to uniformity of temperature distribution and cycle time.

In Chapter 5, the modeling of thermal dynamics is described. Specifically, off-line training using a neural model and finite element analysis, and real-time training using on-line learning and experimental data are explained. Then, the model validation results are provided.

In Chapter 6, fuzzy logic and neural control techniques are described to design the cavity-wall temperature controllers, and performance of designed controllers are discussed in terms of tracking accuracy and response time.

In Chapter 7, concluding remarks including summary, contributions and future work are provided.

Chapter 2
Literature Review

The literature reviewed in this chapter covers several topics that are pertinent to this work. A brief overview of conventional strategies for die thermal management is presented in Section 2.1. The use of neural networks for control is introduced in Section 2.2. In Section 2.3, neural network learning algorithms required for system identification are explained. In Section 2.4, the features and schemes of neural control are introduced. Finally, existing methodologies in optimal sensor location and number, and some studies for sensor construction in die systems are reviewed in Section 2.5.

2.1 Conventional thermal control strategies for die systems

Conventional methods for thermal management in die systems have certain problems including low efficiency due to the typical trial and error approach used and operator experience dependency. Consequently, there has been increasing demand for techniques that effectively manage the die temperature, and various studies have investigated die temperature control.

Dubay et al. [10] applied a PI controller to control coolant flow rate and manage the cavity temperature on a plastic injection moulding machine for achieving an effective cooling system. In the study of Bishenden and Bhola [7], a PID algorithm maintains the die cavity temperature within a preset range during the casting process, improving production through improved response time. Yang et al. [11], [12] also designed a fuzzy PID temperature controller to minimize the temperature differences between cooling channels. Through a control action with fuzzy rules depending on the temperature condition at each channel, the controller realizes a homogeneous temperature distribution as well as a reduction of the temperature of hot spots in the die. The aforementioned studies commonly regulate the die temperature by controlling the water flow rate in water lines using a PID controller. Although these approaches used a conventional (non-adaptive) PID for its simple implementation and easy tuning, this PID control has limitations that result in unsatisfactory performance in systems with uncertain dynamics.

To improve the performance of a PID controller, Gao et al. [13] utilized a Dahlin controller as a digital control algorithm; this looks like a PI controller with an additional dead-time term. In their study, PI, PID and Dahlin controller algorithms were evaluated in light of the mould temperature control in injection moulding. Using coolant temperature as a manipulated variable, the Dahlin

controller provides a better control performance than the other controllers (i.e., PI and PID) in the injection mould process with uncertain dynamics. However, the Dahlin algorithm could be effective through larger variations in the manipulated variable and thus may not be valid in some circumstances with more complex nonlinearities.

As another alternative to PID control, Model Predictive Control (MPC) has been utilized for the temperature control in die systems. Based on a simplified mathematical process model obtained from system identification, MPC predicts the effect of control actions on the output variables. In the MPC strategy, controller commands are determined by minimizing the error between the desired and predicted response trajectories in the cost function at every sampling time and calculated commands are sent to the corresponding regulatory controller setpoints in the process. In the study by Vetter et al. [14], a proposed MPC improved the temperature control in a wheel die casting process simulation by being able to reject process disturbance (i.e., variations in the incoming aluminum temperature and longer die open time). Dubay in [15] developed an ARX (Auto Regressive Exogenous) model based MPC to control the melt temperature in plastic injection moulding. Compared to PID methods, this MPC controller prevented the thermal degradation of the polymer without overshoot response and oscillations. In [16], [17], the MPC was used for the temperature control of an extrusion barrel in a plastic injection moulding process. Through a stochastic discrete-time mathematical model with recursive least-square estimation to identify unknown system parameters, the MPC algorithm enhances the set-point tracking performance and disturbance rejection capabilities.

MPC has an attractiveness to circumvent the limitation of a PID controller which shows reasonable performance within only a limited operating range. However, a simplified model of MPC based on a "best-fit" approximation is often not sufficiently accurate, since the performance of an MPC is affected largely by modeling errors. In addition to this, MPC has intensive computational requirements due to large matrix inversion calculations and numerous matrix multiplications, which are not suitable for real-time operations [17].

The methods outlined here have their own strengths and weaknesses, but they all demonstrate that the application of more viable techniques is required for obtaining high performance temperature control in complex and uncertain laminated die systems.

2.2 Introduction of neural networks

2.2.1 Neural network structure

A neural network (NN) is a mathematical model inspired by biological processes that information processing is carried out through the nervous system by a basic unit, the neuron. A NN is composed of a set of neurons arranged in parallel and distributed processing units. From a model of the neuron depicted in Fig. 2-1, the neuron takes n inputs ($x_j(k)$) at time instant k, sums the incoming signals weighted by weights w_j, adds a bias or threshold b_0, and applies the total sum to a nonlinear mapping using the nonlinear function $\sigma(\cdot)$.

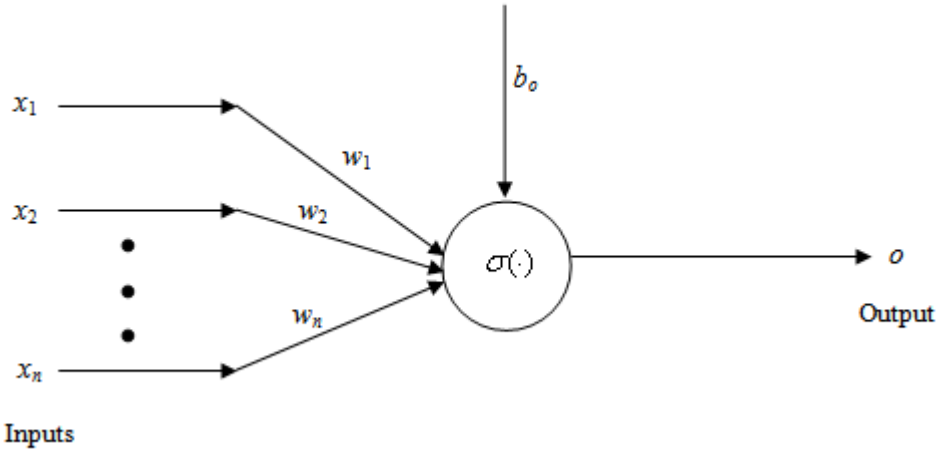


Figure 2-1 Model of a neuron

The output $o(k)$ of the neuron can be expressed as:

$$o(k) = \sigma\left(\sum_{j=1}^{n} w_j x_j(k) + b_0\right) \tag{2-1}$$

The bias b_0 is intended to occasionally inhibit the activity of some nodes [18].

The nonlinear function is called the activation function through which the nonlinear behaviour can be modeled. The sigmoid function is among various formats of activation functions commonly implemented in NNs. Because sigmoid functions take bounded output values for inputs between $-\infty$ and ∞ as illustrated in Fig. 2-2, the activation functions shift left or right as the bias b_0 changes. Since the derivative of the activation function is used for the learning algorithm, the activation function should be differentiable.

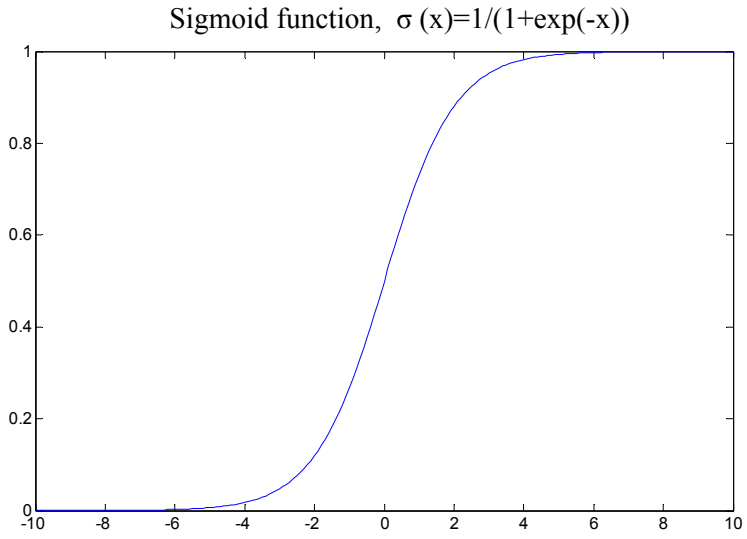


Figure 2-2 Profile of sigmoid function

The structure of the NN is usually made up of multiple layers in which neurons are ordered. This architecture is the most common implementation of the NN, and is known as the multilayer perception network (MLP), an example of which is depicted in Fig. 2-3. As seen from Fig. 2-3, the NN consists of an input layer, a hidden layer (or multiple hidden layers) and an output layer. Neurons in adjacent layers of the network are interconnected via weighted connections.

In the NN, an output is given by the following equation based on Eq. (2-1):

$$o_i(k) = \sigma_o\left(\sum_{h=1}^{l} v_{ih} z_h(k) + b_{i0}^v\right) = \sigma_o\left(\sum_{h=1}^{l} v_{ih}\sigma_h\left(\sum_{j=1}^{n} w_{hj} x_j(k) + b_{h0}^w\right) + b_{i0}^v\right) \tag{2-2}$$

where $i = 1,\ 2,\ ...,\ m$ (node number at the output layer). l and n are the node numbers in the hidden and input layer, respectively. o_i and z_h are the outputs in the output layer and hidden layer. w_{hj}, b_{h0}^w and σ_h are, respectively, the weight, bias, and activation function of the hidden layer, and v_{ih}, b_{i0}^v, and σ_o are the weight, bias, and activation function of the output layer. x_j is the input.

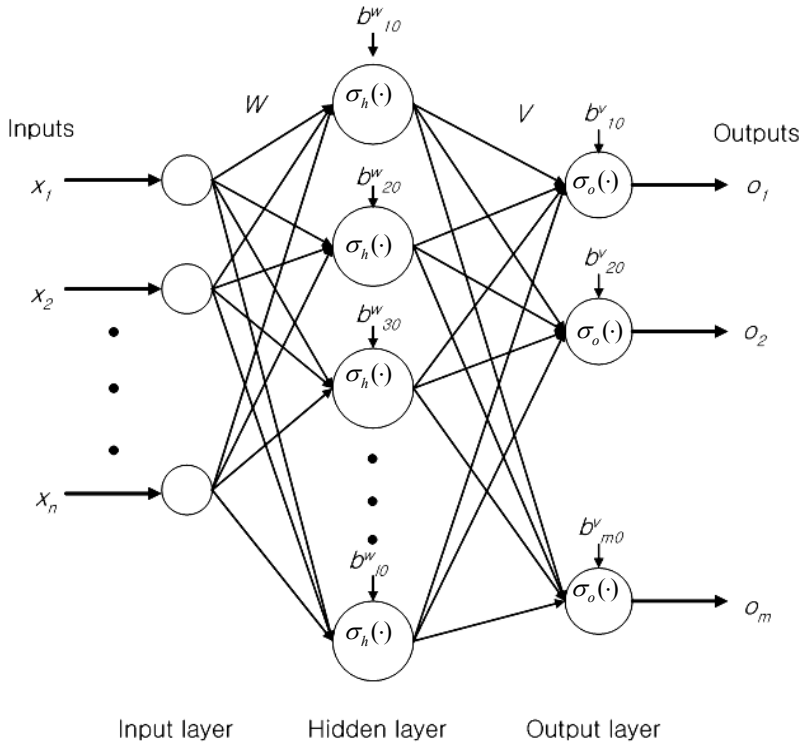


Figure 2-3 MLP network diagram

The internal structure of a NN with interconnections and (nonlinear) activation functions provides massive parallelism and computationally intensive learning through examples, which are useful for functional mapping of nonlinear systems with complex dynamics and uncertain behaviour.

2.2.2 Necessity for neural networks to model die thermal dynamics

Several factors need to be considered for the heat transfer between the mould and cooling channels. An analytical approach to solve the problem of heat conduction and convection is quite complex, even for simple geometries. In this study, the mould has cooling channels with a complex geometry (Fig. 2-4) which conforms to the cavity surface inside the mould, thus allowing fast and uniform heat removal from the mould. This complex geometry of the cooling channel affects the convective boundary conditions and thus leads to difficulty in obtaining analytical solutions to the heat transfer problem. This type of conformal cooling channel is possible using a laminated die fabrication method.

Radiation heat transfer is also difficult to compute, and while not as significant as conduction and convection, cannot be neglected for accurate modeling. Since radiation heat loss is proportional to the fourth power of the (material) temperature, it is treated as an unknown nonlinearity. Therefore, these factors can be the source of unmodeled dynamics.

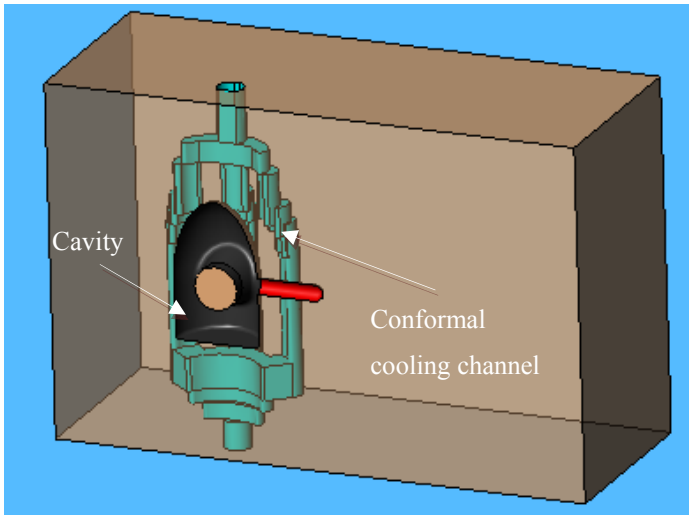


Figure 2-4 Laminated die with conformal cooling channels

A NN technique is adopted as a solution to modeling these physical complexities. Through its learning ability, a NN can learn the correlation between input combinations and the expected outputs, and generalize their relationship. After training, the NN can generate appropriate outputs in response to new inputs. These characteristics of the NN can solve the limitations of many systems in the real-

world with complex relationships and processes for which it is difficult to obtain a mathematical model or algorithm. As a black box approach, system identification using a NN can find an approximate model which can represent the dynamics of the system by training with experimentally measured input-output data.

2.2.3 Existing research using neural networks in die systems

There have been some studies in applying NNs to die casting systems. In the study by Rai et al. [19], a NN based casting process model was designed to predict productivity-quality factors (filling time, solidification time and porosity) for given process parameters of high pressure die casting (inlet melt temperature, mould initial temperature and inlet phase velocity). In the work done by Yarlagadda [20], a NN was applied to the pressure die casting process for the prediction of the optimum injection time. The NN in his study was used as a mechanism mapping between the injection time and interactive casting variables (e.g., injection pressure, die temperature). By finding a training algorithm suitable for this application among several candidates, his work made it easier to select the process parameters without prior knowledge of the die casting process. Krimpenis et al. [21] developed a hybrid model with a NN and a genetic algorithm (GA) in order to select optimal process parameters for pressure die casting. Specifically, the optimized process parameters are determined using the fitness function of the GA while the proposed NN models (Learning Vector Quantization and Feedforward NN) predict defects and solidification time. Through a NN which maps the process parameters to the quality index (degree of porosity), Faessler and Loher [22] presented a NN based classifier to replace a full X-ray examination of the quality of die cast pieces which is an expensive and time-consuming process. The NN with fuzzy logic in [23] was implemented to provide a process model for controlling the dimension of injection moulding parts, using the moulding process parameters such as injection time.

However, NN models from these studies are restricted to model the influences of process parameters on process outputs. In this study, the application of NNs will be extended to the thermal control as well as system identification of die systems with conformal cooling channels.

2.3 Neural network learning algorithms for system identification

2.3.1 Supervised learning mechanism in neural networks

Learning in NNs is the process by which the weights are adjusted to minimize the difference between the function computed and a desired function.

The supervised learning mechanism is most commonly used, and needs a priori known set of input and output data for training and an optimization process during which it seeks the minimization of the cumulative sum of the error between the desired output (target) and the estimated output by the NN. After a number of iterations, the best match between the target and estimated output is obtained where updated weights comply with the optimization criterion. The schematic of the supervised learning is shown in Fig. 2-5. In Section 2.3.3 and 2.3.4, the backpropagation and Levenberg-Marquardt algorithm are used to illustrate the weight updating process of a supervised learning algorithm.

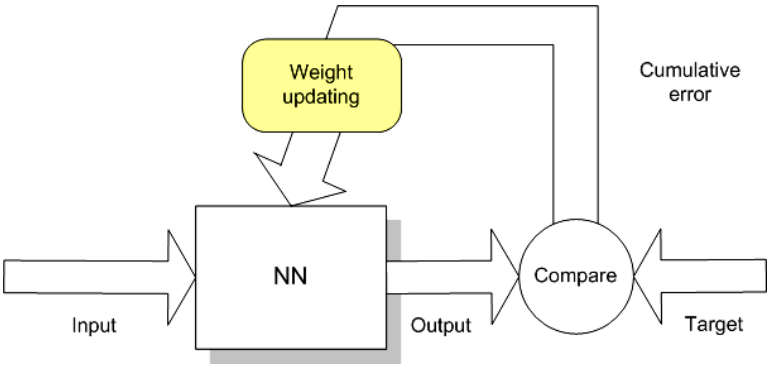


Figure 2-5 Schematic of supervised learning mechanism

2.3.2 Off-line and on-line learning

In 'off-line' learning, the weight updates are computed after summing over all of the training examples. In other words, the weight update for each input sample is computed and stored but the weights do not change until an epoch (one pass through the training set) finishes. At the end of the epoch, all the weight updates are added together and the network weights are updated. This differs from 'on-line' learning in which the weight updates are computed after each input sample and the

network weights are immediately updated. Therefore, the number of weight updates for these two learning methods is different, even for the same number of training examples.

The goal of off-line training is to teach the NN the mapping between the inputs and outputs of a system using a previously recorded data set. In contrast, on-line learning attempts to develop (and track) the mapping by continuously adjusting the network weights as new data becomes available.

2.3.3 Backpropagation learning algorithm

Backpropagation learning (or generalized delta rule) [24], [25] is one of the most popular learning algorithms for training an MLP. This algorithm is based on the gradient descent technique with backward error propagation, which minimizes the network cumulative error E_c:

$$E_c = \sum_{k=1}^{q} E(k) = \sum_{k=1}^{q} \left(\frac{1}{2} \sum_{i=1}^{m} e_i^2(k) \right) = \sum_{k=1}^{q} \left(\frac{1}{2} \sum_{i=1}^{m} \left[t_i(k) - o_i(k) \right]^2 \right) = \frac{1}{2} \sum_{k=1}^{q} \sum_{i=1}^{m} \left[t_i(k) - o_i(k) \right]^2 \quad (2\text{-}3)$$

where E_c is the sum of $E(k)$ for a total number of q training patterns. $E(k)$ represents the sum square of the error, $e_i(k)$, between the k^{th} target output, $t_i(k)$, and the k^{th} NN output, $o_i(k)$, for a total number of m neurons in the output layer.

Depending on the training type (either off-line or on-line training), a different error function (either E_c or $E(k)$) is utilized in the formulation for the learning algorithm. In the case of on-line training, the backpropagation learning rule has $E(k)$ as an error function as follows:

$$\Delta w^{(p)} = -\eta \frac{\partial E(k)}{\partial w^{(p)}} \quad (2\text{-}4)$$

where $\frac{\partial E(k)}{\partial w^{(p)}}$ is the gradient of the error $E(k)$ with respect to the vector $w^{(p)}$, which is the vector of weights interacting between layer p and the previous layer $p-1$. η represents the learning rate that affects convergence and stability. A larger η results in a larger change in weights per epoch and a faster convergence with possible oscillation. Ideally, η should be chosen as large as possible

without leading to oscillation. $\Delta w^{(p)}$ denotes the difference between the vector $w^{(p)}(k+1)$ and $w^{(p)}(k)$.

Based on (2-2) and (2-4), the weights connecting between the hidden and the output layer in the MLP are updated by the following equation:

$$\begin{aligned}\Delta v_{ih} &= -\eta \frac{\partial E(k)}{\partial v_{ih}} = -\eta \left[\frac{\partial E(k)}{\partial o_i}\right]\left[\frac{\partial o_i}{\partial net_i}\right]\left[\frac{\partial net_i}{\partial v_{ih}}\right] \\ &= \eta \left[t_i - o_i\right]\left[\sigma_o'(net_i)\right]\left[z_h\right] = \eta \delta_i z_h\end{aligned} \tag{2-5}$$

where $\delta_i = \left[t_i - o_i\right]\left[\sigma_o'(net_i)\right]$ is the error signal of the i^{th} neuron of the output layer. $net_i = v_{ih} z_h(k)$ is the sum of all signals from the hidden layer reaching the i^{th} neuron in the output layer.

Similarly, the weight update between the input and the hidden layer in the MLP is carried out by:

$$\begin{aligned}\Delta w_{hj} &= -\eta \frac{\partial E(k)}{\partial w_{hj}} = -\eta \left[\frac{\partial E(k)}{\partial z_h}\right]\left[\frac{\partial z_h}{\partial net_h}\right]\left[\frac{\partial net_h}{\partial w_{hj}}\right] \\ &= \eta \left(-\left[\frac{\partial E(k)}{\partial z_h}\right]\left[\sigma_h'(net_h)\right]\right)\left[x_j\right] = \eta \delta_h x_j\end{aligned} \tag{2-6}$$

where $\delta_h = -\left[\frac{\partial E(k)}{\partial z_h}\right]\left[\sigma_h'(net_h)\right] = \left[\sigma_h'(net_h)\right]\sum_{i=1}^{m} \delta_i v_{ih}$ is the error signal of the h^{th} neuron of the hidden layer. $net_h = w_{hj} x_j(k)$ is the sum of all signals reaching the h^{th} neuron in the hidden layer coming from the input layer. δ_i is defined in (2-5).

In the case of off-line training, E_c is used for the backpropagation learning rule expressed as:

$$\Delta w^{(p)} = -\eta \frac{\partial E_c}{\partial w^{(p)}} \tag{2-7}$$

By adding a momentum term to the backpropagation learning, the modified weight updating algorithm is obtained where the weight change (i.e., $\Delta w^{(p)}(t+1)$) is equal to the sum of the new current change by the backpropagation rule (i.e., $-\eta \frac{\partial E_c(t)}{\partial w^{(p)}}$) and a fraction of the previous weight change (i.e., $\gamma \Delta w^{(p)}(t)$) as follows:

$$\Delta w^{(p)}(t+1) = -\eta \frac{\partial E_c(t)}{\partial w^{(p)}} + \gamma \Delta w^{(p)}(t) \tag{2-8}$$

where γ is the momentum constant which determines the amount of influence from the previous change to the present one. $\Delta w^{(p)}(t+1) = w^{(p)}(t+1) - w^{(p)}(t)$.

If all weight changes in the right hand side of (2-8) are all in the same direction, the weight change ($\Delta w^{(p)}(t+1)$) will amplify because the previous weight change contributes cumulatively towards the new current update. In the opposite case, the effect of each change (the previous weight change and new current change) will tend to cancel each other. Therefore, the momentum term plays a role to speed up leaning (in the same direction) while it tends to stabilize convergence by reducing oscillations in the weight space (in the opposite direction).

2.3.4 Levenberg-Marquardt learning algorithm

Among various learning algorithms, the Levenberg-Marquardt (LM) algorithm is widely used due to its efficiency of realization accuracy [26]. By combining the gradient descent algorithm with the Gauss-Newton algorithm, it can give a compromise between the speed of the Newton algorithm and the stability of the steepest descent method, and thus constitutes a good compromise between these methods [27]. LM updates the weights of the NN as follows:

The performance function $E_c(w)$ from (2-3) can be expressed as:

$$E_c(w) = \frac{1}{2}\sum_{k=1}^{q}\sum_{i=1}^{m} e_i^{\,2}(k) = \frac{1}{2} e(w)^T e(w) \tag{2-9}$$

where $e(w) = \left[e_1(k{=}1) \ \ldots \ e_m(k=1) \ \ e_1(k=2) \ \ldots \ e_m(k=2) \ \ \ldots \ \ e_1(k=q) \ \ldots \ e_m(k=q)\right]^T$ is the cumulative error vector for all patterns and output layer's neurons; $E_c(w)$ and $e(w)$ are evaluated in the weight vector $w = \left[w_1 \ \ w_2 \ \ \cdots \ \ w_{nw}\right]^T$ which consists of all weights of the network; q , m and nw are the number of patterns, neurons in the output layer and weights of the network, respectively.

The gradient of $E_c(w)$ can be expressed as:

$$\nabla E_c(w) = J_c(w)^T e(w) \tag{2-10}$$

where $J_c(w)$ is the Jacobian matrix given by:

$$J_c(w) = \begin{bmatrix} \frac{\partial e_1(1)}{\partial w_1} & \frac{\partial e_1(1)}{\partial w_2} & \cdots & \frac{\partial e_1(1)}{\partial w_{nw}} \\ \frac{\partial e_2(1)}{\partial w_1} & \frac{\partial e_2(1)}{\partial w_2} & \cdots & \frac{\partial e_2(1)}{\partial w_{nw}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_m(1)}{\partial w_1} & \frac{\partial e_m(1)}{\partial w_2} & \cdots & \frac{\partial e_m(1)}{\partial w_{nw}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_1(q)}{\partial w_1} & \frac{\partial e_1(q)}{\partial w_2} & \cdots & \frac{\partial e_1(q)}{\partial w_{nw}} \\ \frac{\partial e_2(q)}{\partial w_1} & \frac{\partial e_2(q)}{\partial w_2} & \cdots & \frac{\partial e_2(q)}{\partial w_{nw}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_m(q)}{\partial w_1} & \frac{\partial e_m(q)}{\partial w_2} & \cdots & \frac{\partial e_m(q)}{\partial w_{nw}} \end{bmatrix} \tag{2-11}$$

In the LM algorithm based on (2-4), the weight increment Δw is computed by:

$$\Delta w = -\left[J_c(w)^T J_c(w) + \mu I \right]^{-1} \nabla E_c(w) \tag{2-12}$$

where I is the identity matrix, and μ is a learning parameter. If μ is increased so that $J_c(w)^T J_c(w)$ is negligible, the LM algorithm becomes gradient descent with a learning rate $1/\mu$. If μ is zero, the LM algorithm is the same as the Gauss-Newton algorithm.

The parameter μ is increased or decreased during the adaptation. If the error is reduced, then μ is decreased; μ is increased in other cases. Through this process, the LM algorithm tries to achieve the goal of having μ as small as possible while guaranteeing inversion of $\left[J_c(w)^T J_c(w) + \mu I \right]$.

2.4 Neural control

2.4.1 Introduction of neural control

Classical control strategies based mainly on linearization of systems requires mathematical modeling. However, such mathematical models might not exactly reflect the physical properties (or input-output physical relations) of many real-world systems having nonlinearities and unmodeled dynamics. Statistical models (e.g., auto-regressive (AR) model and auto-regressive moving-average (ARMA) model) are examples of mathematical models obtained by approximation of the actual physical properties [28]. With respect to control applications, such approximation through which the linear models are fitted to complex nonlinear systems might not be sufficiently accurate to capture the input and output of the systems.

An adaptive control strategy can be a good solution to solve this problem. It enables unknown parameters in the mathematical model to be estimated using a cost function during the control process. Thus it makes the mathematical model properties (or parameters) approach those of the actual system [29]. However, it is also based on linear system assumptions and therefore requires rebuilding the model and determining a new control law when changes in the actual system occur [28].

To avoid this problem, researchers have considered neural network control. This control has shown successful performance for complex and nonlinear control problems with which conventional approaches have struggled. Features of neural control are explained in the next section.

2.4.2 Features of neural control

Neural control is defined as the use of neural networks to generate control signals [28]. While neural control has the disadvantage that it is not obvious how the network makes a decision [30] (i.e., it is hard to retrieve the structural knowledge from the network which can be formulated for solving the respective control task), the following features of neural control make it an attractive alternative to conventional control approaches for solving nonlinear and complex control problems [28], [31]:

- Self-learning ability of neural control techniques eliminates the need for complex and difficult mathematical modeling.
- The use of a sigmoid (or other non-linear) activation function in the hidden layers provides nonlinear mapping ability for nonlinear control problems.
- The inherent parallelism in neural networks offers fast multiprocessing technique when implemented using parallel hardware.

Neural control schemes have been categorized in [18], [28] as follows:

- Series control scheme (or direct inverse control, as described in Section 6.2.4.1) [32]: This control leads ideally to an overall transfer function of unity by mapping the desired reference signals to the control inputs.
- Parallel control scheme (or additive feedforward control, as described in Section 6.2.4.2) [33]: This neural controller adjusts the control signal generated by a conventional controller such that the error between the desired output and plant output is minimized as much as possible.
- Self-tuning control scheme (as described in Section 6.2.3) [34]: A neural network is used to tune parameters of a conventional controller on-line.
- Emulator and controller scheme [35]: This control consists of two neural networks, which play a role of an emulator (or identifier) and a controller, respectively. In this structure, identification is carried out first, and then the identified model is compared with a reference (model) output and the error between them is calculated. Finally, this error is used to update the control law by modifying the controller weights.

2.5 Optimum sensor location and number

2.5.1 Existing literature in optimal sensor location and number

Since the quality of information extracted from the data can vary depending on the measurement point, the accuracy of the state estimate is affected by the sensor location and the number of sensors used. From this viewpoint, the sensors should be located in such a way that measurements are most informative about the estimated parameters and are most sensitive to any changes in the parameters. Because the overall performance and efficiency of the controller can be improved by means of optimal sensor location, the sensor location should be also considered as a significant factor in the control design [36].

The issue of optimal sensor location and number has attracted the interest of many researchers. In the past, the optimal sensor location problem was solved through positioning the given sensors, using the data from locations with a specific estimator, and repeating the procedure for different sensor locations [37]. The locations are considered as optimal when they produce the best parameter estimates. However, this trial and error method has a limitation in that it cannot provide any physical insight about why these locations are preferable to others.

To overcome this drawback, Udwadia [37] suggested a methodology for optimum sensor location for parameter identification. Using the Fisher information matrix (FIM) that maximizes a certain norm such as the trace, the author provided an algorithm to rank the sensor locations for both given fixed sensors and additional sensors. Since the maximization of the Fisher information matrix provides the minimum value of the covariance of the estimation error and since the Fisher information matrix is useful in terms of the computational ease and simplicity, this method can efficiently deal with the problem of optimal sensor location. Fadale et al. [38] applied the Fisher information matrix and the sensitivity analysis to determine the optimal sensor locations for the conductivity estimation of the thermal system. In their study, the Fisher information matrix was used as a measure of the sensitivity of the signals to noise. The location with maximum measure of the information matrix represents the optimum location where the total sensitivity of system is a maximum or the estimation error is minimal. Using the statistical information in variation of parameters and their interactions, the sensitivity analysis can find the optimal location where the sensitivity of the response with respect to the parameters is maximal. While the Fisher information matrix is based on the noise influence, the sensitivity analysis focuses on the effect of parameters.

As another type of information-based approach, a heuristic algorithm based on information entropy in the study of [39], [40] was proposed for constructing effective optimal sensor configurations. Since the information entropy is a measure of the uncertainty in the system parameter estimates, the optimal sensor configuration is chosen as that which minimizes the information entropy. Also, the information entropy is a decreasing function of the number of sensors because the value of the information entropy decreases as additional sensors are placed in the structure. By placing one sensor at a time in the structure at a position that results in the largest reduction in information entropy, the Sequential Sensor Placement algorithm (heuristic algorithm in the study of [39]) computes optimal sensor positions. This algorithm enables accurate estimates of optimal sensor configuration with minimal computational effort. However, the efficacy of entropy-based approaches is validated through only problems with a smaller set of possible candidate sensor configurations [41].

With good computational power, the aforementioned information-based approaches are attractive solutions for optimal sensor location in engineering problems where there are a large number of potential sensor locations [41]. However, these are efficient but suboptimal due to the iterative manner (i.e., iteratively reduce or expand the initial candidate sensor locations to the desired number of sensors). Therefore, the errors between the real and estimated responses at the sensor locations determined by these methods cannot be guaranteed to be a minimum.

In contrast to the information-based approaches, global optimization techniques such as genetic algorithms [42] and particle swarm optimization (PSO) algorithms [41] with the FIM as a fitness function have been applied. In PSO, a particle is regarded as a point in the search space which is a solution. All particles have fitness values and velocities, and are searched for the best position (i.e., global solution) with the best fitness value in the problem space, which is discovered by the whole population during the search process. The PSO algorithms are known as self-configuring because they utilize the acquired information by dynamically changing the PSO parameters over the course of the search process [41]. Using the PSO with a local search algorithm (Derivative-free nonlinear optimization algorithm), Rao and Anandakumar [41] tried to identify the optimal sensor positions and number for civil engineering structures for a desired level of accuracy. This hybrid PSO algorithm showed superior performance in terms of the generation of optimal senor locations and faster convergence when compared to the iterative information-based approaches.

2.5.2 Factors considered for optimal sensor construction in die systems

There are some studies which proposed a methodology for construction of sensors suitable for a die casting system. In Dour et al.'s study [43] to determine the die surface temperature and heat flux density in high pressure die casting (HPDC), it is reported that the sensor configuration for accurate determination of the die temperature and heat flux density depends on the position of the critical thermocouple (i.e., the closest thermocouple to the interface of die) and the duration of the heat input at the surface of the die.

Based on [43], accurate measurements of heat-transfer coefficient and the heat flux density at the casting-die interface for HPDC of magnesium alloys were studied [44]. Typically, measuring the temperature with a thermocouple in a die casting system cannot guarantee accurate and reproducible measurements due to the heat transfer through filling and solidification in the die system, and the sensor dynamics (i.e., slow response time of thermocouple compared to fast transient heat transfer in HPDC) [44].

Chapter 3
Optimal sensor location

3.1 Problem overview

The temperature distribution in a laminated die varies both spatially and temporally during the casting process in response to heat introduced by the molten material being cast and the heat removed via the coolant. This temperature distribution can be predicted using a discrete model, given the measured temperatures at several discrete locations within the die. However, the existing techniques described in Section 2.5.1 have not dealt with a problem of redundant sensor locations replaceable by a representative sensor location using a relationship of measured data between these locations. This issue is important for sensor configuration because it is related to the required number of sensors.

Even though the approaches described in Section 2.5.2 investigate an optimal method for the temperature measurement in the die casting system, their focus is on consideration of heat transfer through filling and solidification and sensor dynamics rather than finding optimal sensor locations and number of required sensors. To deal with the latter issues in a laminated die, clustering methods and sensitivity analysis are applied in this study.

In the following sections, the laminated die system used in this study is introduced and the methodologies for clustering and sensitivity analysis are explained. By comparing the simulation results between K-means and the temperature ratio method using several criteria including consistency in generating sensor locations, degree of sensitivity, mutual interaction using principal component analysis, controlling the number of sensors and accuracy in estimation of temperature response, the best method for optimal sensor location will be determined.

3.2 System description

To analyze the thermal dynamics of a laminated die with conformal cooling channels, the plastic injection mould shown in Figure 2-4 (pg 10) is considered in this study. The injected hot polymer enters the cavity and is cooled to a certain temperature through heat transfer to coolant flowing through the conformal cooling channels. The flow rate and temperature of the coolant are the controllable process parameters with which the temperature distribution of the die is controlled [45]. A finite element analysis (FEA) for identifying the thermal dynamics of the die is conducted based on a solid model of the die.

3.3 Finite element analysis of a laminated die system

3.3.1 Finite element analysis methodology

The injection moulding process with the aforementioned model in Section 3.2 can be briefly described as follows:

Polymer is injected into the cavity at a temperature of 165.5 ^oC before the injection moulding cycle starts. During the cycle, heat is removed from the mould and the melt through heat conduction with the coolant in the conformal cooling channels and natural convection with air. After one cycle, the mould is opened and the polymer part is ejected from the cavity.

Two FE analyses were conducted to find the optimal sensor locations. The first FEA was designed to provide the temperature distribution at nodes in the mould (which is used for clustering nodes). The second FEA is required to determine the most sensitive sensor locations for each cluster. For the FEA, ANSYS CFX 11 software package was used with 3D CAD files.

Material properties of the steel mould [46] and Santoprene polymer [47] given in Table 3-1 and Table 3-2 were applied. The coolant properties were assumed to be those of water. Flow within the cooling channels was assumed to be turbulent and the k-ε (k-epsilon) model was employed as turbulence model, which is a standard turbulence model due to numerical robustness and good prediction [48]. The mould was assumed to initially be at room temperature (25 ^oC). For natural convection with air across the surface of the mould, the value of 6 $W/m^2\ ^oC$ was chosen as the convective heat transfer coefficient. A transient state analysis in both FEA was carried out.

`

Table 3-1 Material properties of ASTM-A36 (mould)

Property	Value
Density (kg/m^3)	7830
Tensile strength, yield (MPa)	250
Modulus of elasticity (GPa)	200
Poissons ratio	0.260
Thermal conductivity ($W/m\ ^oC$)	54
Specific heat ($J/kg\ ^oC$)	490

Table 3-2 Material properties of Santoprene 8211-45 (polymer)

Property	Value
Density (kg/m^3)	790
Tensile strength (MPa) - Across flow (Break, 23°C)	3.60
Tensile elongation (%) - Across flow (Break, 23°C)	410
Thermal conductivity ($W/m\ ^oC$)	0.1
Specific heat ($J/kg\ ^oC$)	2380
Brittleness temperature (^oC)	-62.0

In order to reduce error due to the discretization of the domain and to obtain accurate solutions [49], a series of simulations were performed by increasing the number of elements until a convergence in the temperature distribution was achieved. Through the mesh convergence study, mesh models were generated with a total of 1129514 elements and 217719 nodes for the mould with conformal channels using the tetrahedral mesh (default mesh setting for CFX). Figure 3-1 exhibits the mesh model of the conformal cooling channel mould.

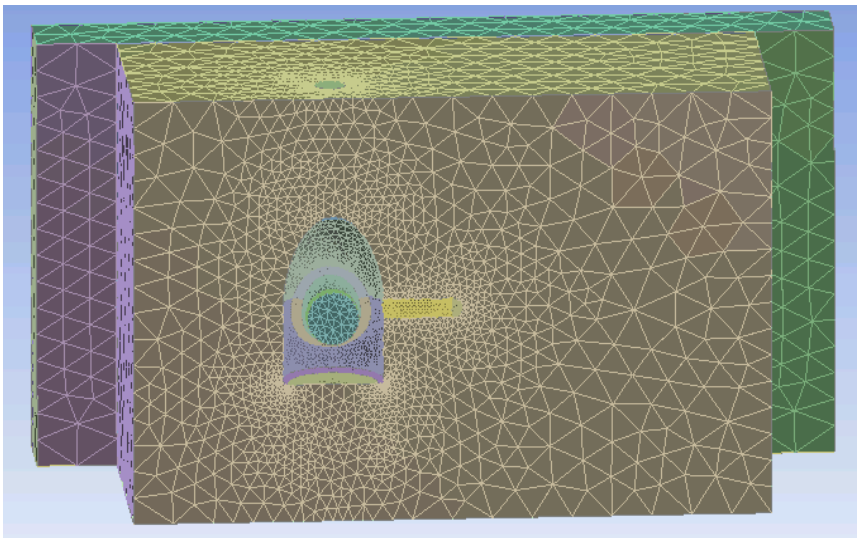


Figure 3-1 Mesh model of mould with conformal cooling channels

Additional boundary conditions of the first FEA are as follows:

The polymer temperature at the entrance to the die was fixed at 165.5 ^{o}C. The focus on the first FEA is on gathering the temperature profile at nodes in the mould which varies according to controllable inputs, i.e. cooling water flow rate and temperature. The other boundary conditions (including polymer temperature) were held constant during the simulation to observe the influence of the controllable inputs on the temperature profile at the model nodes. APRBS signals (amplitude modulated pseudo random binary signal) were used as simulation inputs to generate data sets which are required for system identification. These signals are suitable for identifying dynamic models (or training the NN) with uncertainties by exciting the process at a wide range of amplitudes and frequencies [50]. Therefore, the inputs were randomly generated with different time intervals and different levels of flow rate and temperature, which cover all possible combinations of operating conditions (see Fig. 3-2). A simulation time of 360 seconds was arbitrarily chosen to generate the operating input conditions.

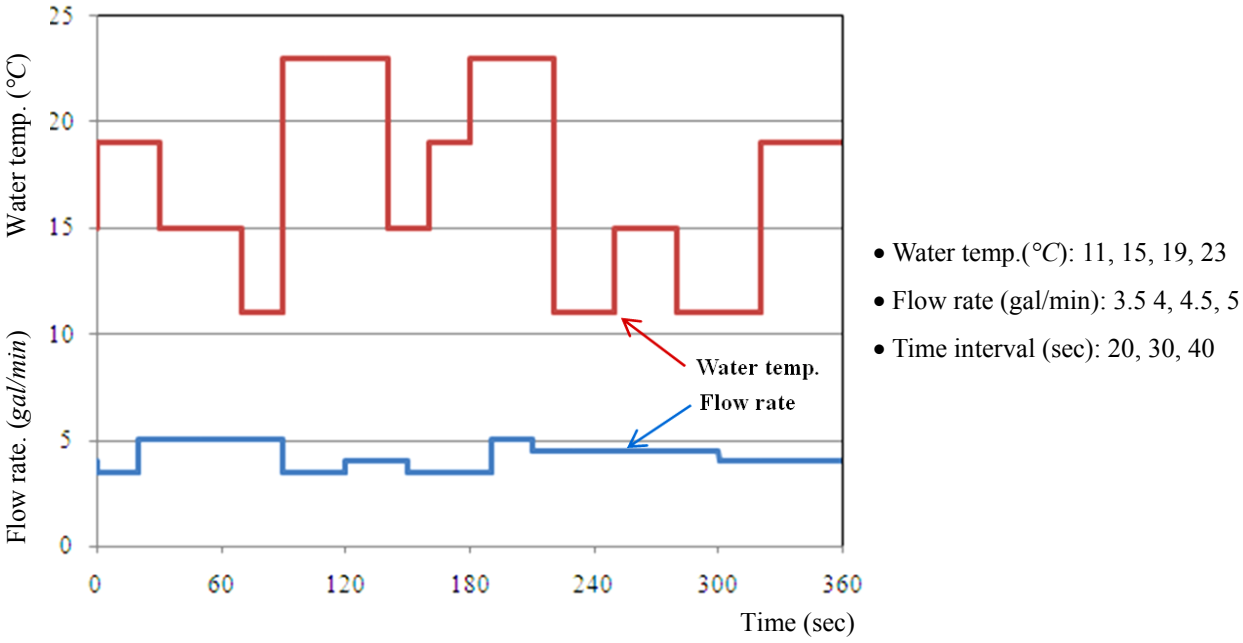


Figure 3-2 Input conditions for the first FEA

The nodes for potential (target) sensor locations where the temperature profile is monitored are shown in Fig. 3-3. The target nodes for potential sensor locations are located on a 3D grid (as a domain for total target nodes in Fig. 3-3-(b)) consisting of 990 unit cubic elements (unit element size: 10×10×10 mm) the vertices of which are considered as target nodes.

Some geometric constraints for target nodes were considered. As seen in Fig. 3-3-(b), the target nodes are offset by 10 mm from every face of the die, and offset by 15 mm from the back plate. In addition, nodes in the dotted cubic domain were excluded as potential target nodes since the installation of sensors at these nodes would not be possible due to proximity to the cooling channels. Finally, the remaining 1152 nodes were used as target nodes.

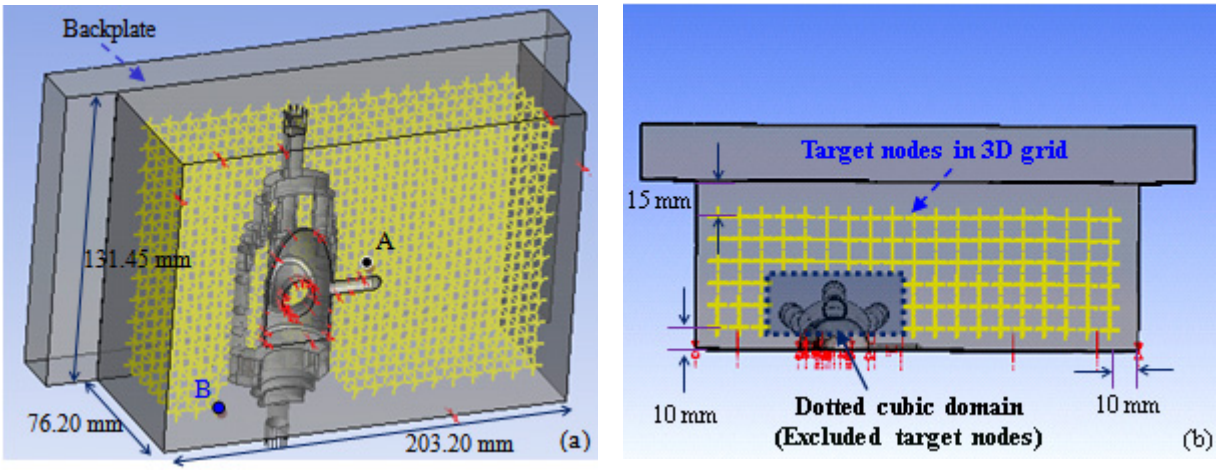


Figure 3-3 Monitored nodes of mould (side view (a) and top view (b))

The second FEA was carried out for a sensitivity analysis. Sensitivity is a measure of the effect of a change in one factor on another factor. It can be expressed by the partial derivative of the dependent variable with respect to the input parameter [51]:

$$X_{i,m} = \frac{\partial y_i}{\partial a_m} \tag{3-1}$$

where $X_{i,m}$ is the sensitivity coefficient of the dependent variable y with respect to the m^{th} parameter of the input variable a at the i^{th} observation point.

Equation (3-1) can be normalized by the input parameter value as follows:

$$X_{i,m} = \frac{\partial y_i}{\partial a_m / a_m} \tag{3-2}$$

This normalization makes the sensitivity coefficient with respect to any given input parameter have the same unit (dependant variable's unit). Thus, the comparison of sensitivity coefficients among different input parameters is available. In our case, the normalized form in Eq. (3-2) was adopted as two different types of input parameters were used.

By making a small perturbation in the input parameter value, the following approximation of Eq. (3-2) was used for the sensitivity analysis:

$$X_{i,m} = \frac{\partial y_i}{\partial a_m / a_m} \approx \frac{y_i(a_m + \Delta a_m) - y_i(a_m - \Delta a_m)}{2\Delta a_m / a_m} \tag{3-3}$$

where a_m is the input parameter value for the base case, Δa_m is a small change in the parameter, $y(a_m - \Delta a_m)$ and $y(a_m + \Delta a_m)$ are the values of the dependent variable corresponding to $a_m - \Delta a_m$ and $a_m + \Delta a_m$, respectively. Since the above central finite difference method is more accurate than the forward/backward finite difference method in many cases [52], it was applied in this study.

In order to calculate the sensitivity coefficient value of Eq. (3-3) with respect to two input parameters, 4gpm and 15^oC, flow rate and temperature of the coolant respectively, were considered as a base case for each input. For the simulation, the perturbation Δa_m in one input parameter was applied while the other input parameter was kept constant. As a rule of thumb, a perturbation size of 1~ 5% is recommended [51]. In the study, the perturbation size of 5% for each parameter's change was chosen to calculate $X_{i,m}$.

Some boundary conditions different from ones used for the first FEA were applied for the second FEA in order to reflect the actual injection moulding process. For example, the temperature of the polymer initially given as $165.5^{o}C$ was not kept fixed during the simulation. The polymer was cooled down through the heat transfer to the cooling channels. A simulation time of 87 seconds was chosen, corresponding to one cycle time of the actual injection moulding process.

3.3.2 FEA simulation results

The results of the first FE simulation with the input conditions shown in Fig. 3-2 are presented in Fig. 3-4. The temperature profiles at nodes A and B in Fig. 3-3-(a) inside the mould are depicted as an example. As illustrated in Fig. 3-4, the temperature at node A around the cavity with the hot polymer is higher than node B near the outlet of the cooling channels during the simulation time.

Based on the temperature profiles at the 1152 nodes from this FE simulation, the correlation coefficient of the temperature between each pair of nodes was calculated. Then it was used as a criterion for clustering target sensor locations by K-means, which will be explained in the next section.

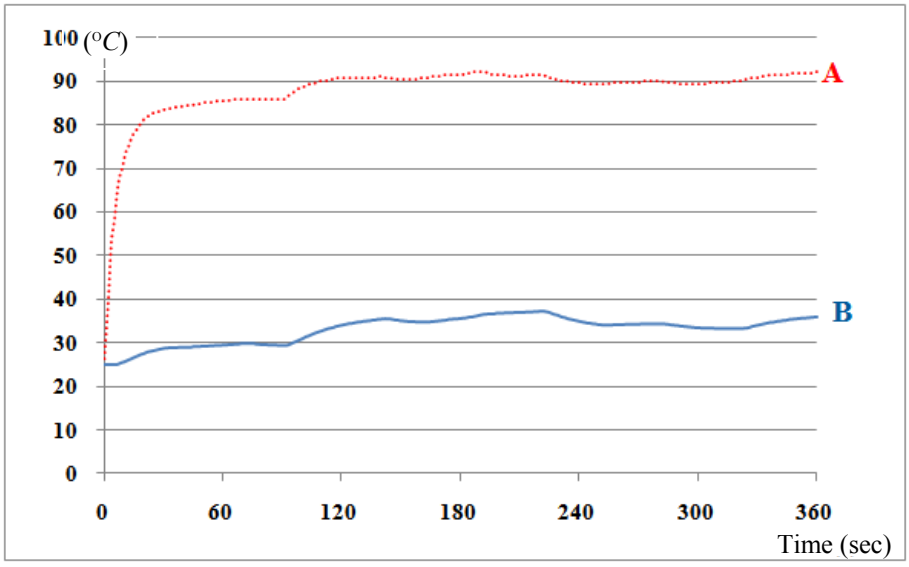


Figure 3-4 Temperature distributions through the first FEA at node A and B

To describe simulation results of the second FEA for sensitivity analysis, the response at node B was used. Figure 3-5 and Figure 3-6 show the temperature profile and sensitivity coefficient value at node B to a 5% change in flow rate and coolant temperature, respectively. Figure 3-5-(a) illustrates the temperature response from an ambient 25 ^{o}C as coolant of 15 ^{o}C conducts heat away. The coolant flow is initially 4.0gpm, and the two curves show the effect on the response of a 5% increase (4.2gpm) and 5% decrease (3.8gpm). It is noted that an increase in the flow rate of coolant causes a more rapid drop in temperature, as would be expected.

This is confirmed by a sensitivity coefficient value of the temperature at node B to the flow rate. Using the temperature response in Fig. 3-5-(a) with Eq. (3-3), the value of the sensitivity coefficient, $X_{i,m}$ plotted in Fig. 3-5-(b) was calculated as follows:

$$\begin{aligned} X_{i,m} &\approx \frac{y_i(a_m + \Delta a_m) - y_i(a_m - \Delta a_m)}{2\Delta a_m / a_m} \\ &= \frac{T_B(4\text{gpm}+0.2\text{gpm}) - T_B(4\text{gpm} - 0.2\text{gpm})}{2 \cdot 0.2\text{gpm} / 4\text{gpm}} \end{aligned} \tag{3-4}$$

where a_m is 4gpm, Δa_m is 0.2gpm, m is the flow rate parameter, i is node B, y is the temperature T. The negative values of $X_{i,m}$ at all nodes including node B imply a decrease in temperature given an increase in the flow rate.

The temperature response and sensitivity coefficient at node B to ± 5% coolant temperature (for a flow rate of 4gpm) are shown in Fig. 3-6. It is evident that a 5% change in coolant temperature has more significant effect than a 5% change in flow rate, and that as expected, a lower coolant temperature results in a lower mould temperature.

3.4 Methodology for sensor location identification

3.4.1 Clustering

Performing clustering on target sensor locations prior to determining the optimal sensor locations has some advantages. First, it creates a regular pattern through which grouping target locations according to similarity becomes available. Thus, the sensor locations belonging to different groups can provide

the unique information representing each group and also the location information (e.g., dense region of the sensor locations) can be obtained. Secondly, it helps reduce the required number of sensors by selecting one representative location per cluster as a sensor location instead of all target locations. In this case, the representative location could be utilized to estimate the response at another location. This advantage is very important for the sake of minimizing the cost of instrumentation, data processing and handling through smaller number of sensors [37]. In this section, K-means and a proposed temperature ratio based method for clustering target sensor locations are introduced.

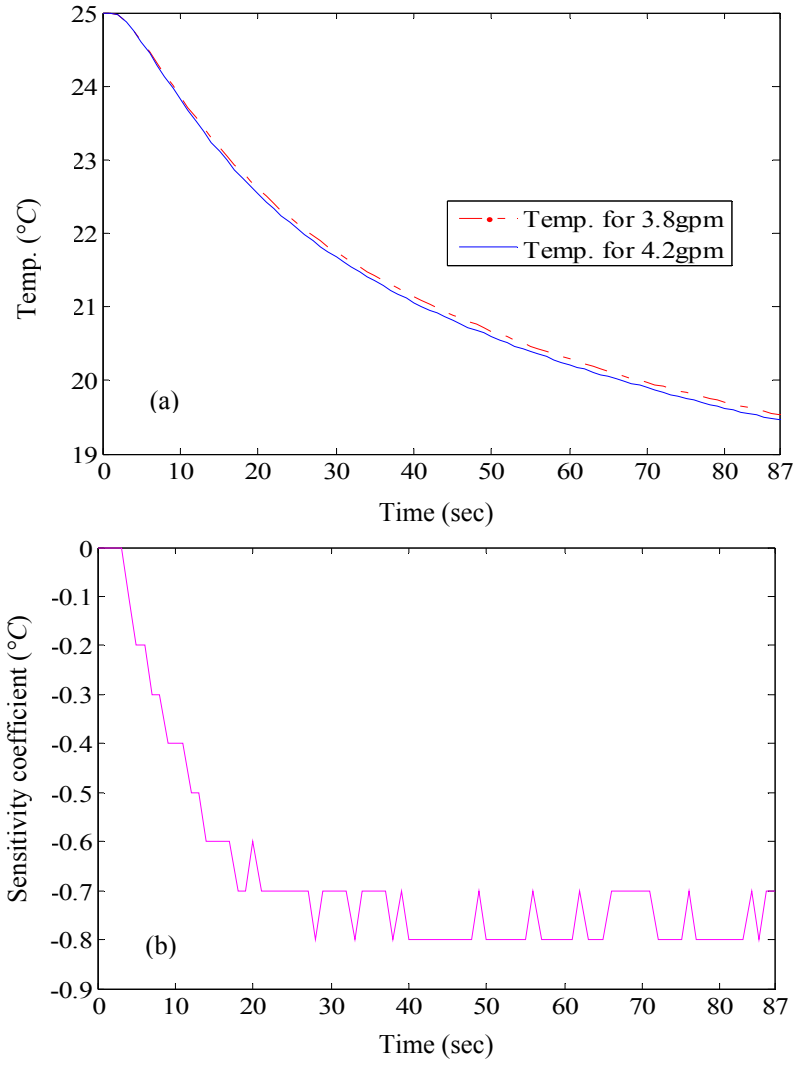


Figure 3-5 Temperature distributions with +5% and −5 % change in flow rate (a) and sensitivity coefficient value of temperature to flow rate (b) at node B

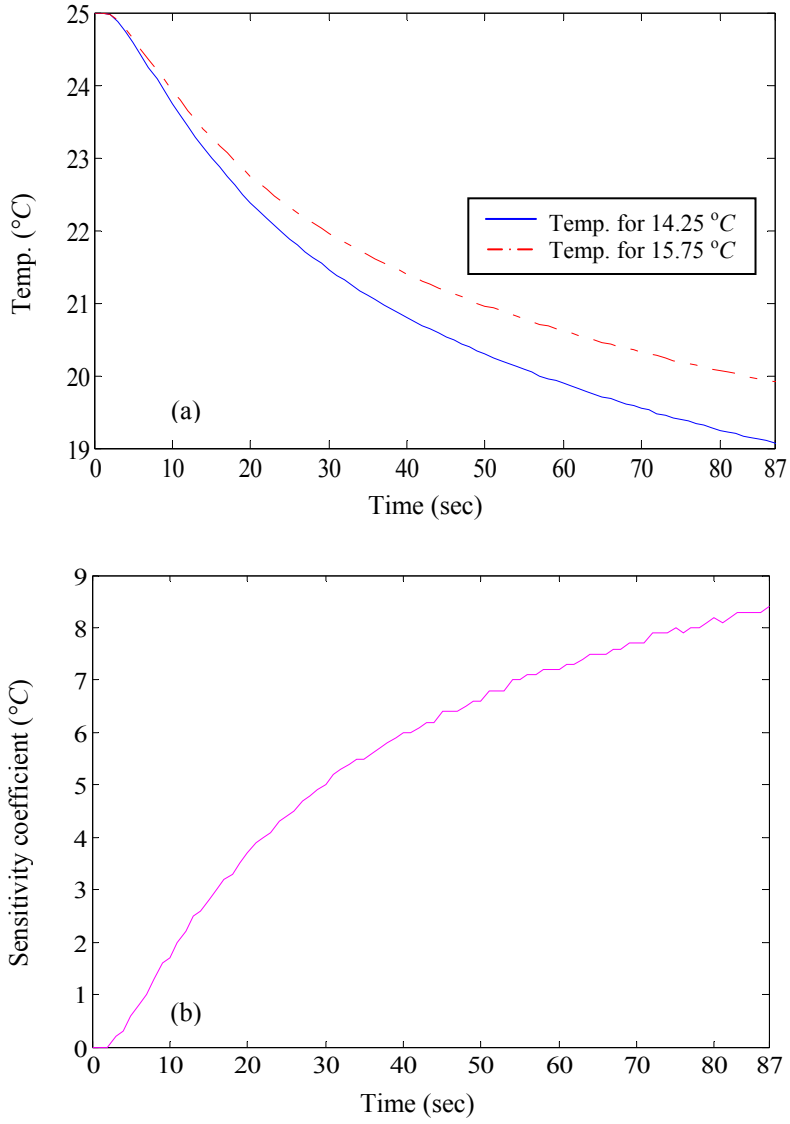


Figure 3-6 Temperature distributions with +5% and −5 % change in water temperature (a) and sensitivity coefficient value of temperature to water temperature (b) at node B

- K-means

The K-means algorithm is the most widely used method in partitioning due to its fast speed and ease of understanding [53]; it classifies a given data set into a predefined number of clusters, K. The first step in K-means clustering is to define K centroids, one for each cluster. The next step is to assign

points to the nearest centroid. Then, the new centroid of each cluster is recalculated by averaging points belonging to each cluster. The K-means clustering is repeatedly performed until the centroids no longer move. Namely, it is repeated until the centroids minimize the total variance within clusters, which is an objective function of the K-means algorithm defined as:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_{en\,j} \right\|^2 \tag{3-5}$$

where $\left\| x_i^{(j)} - c_{en\,j} \right\|^2$ is a distance measure between a data point $x_i^{(j)}$ and the cluster centroid $c_{en\,j}$, n is the number of data points, and k is the number of clusters.

The centroids can be computed by various distance measures. Among these, the Euclidean distance is generally used in dimensional space. By the definition of Euclidean distance, each centroid is the mean coordinate of all points in each cluster. In this study, target sensor locations are clustered according to the similarity of the thermal response of all nodes in the cluster (sensitivity to changes in input parameters) instead of spatial distance.

Therefore, the correlation between nodes in terms of temperature response was considered as the distance measure for K-means clustering. For this, the Pearson correlation coefficient, $P_{x,y}$ in Eq. (3-6) was used.

$$P_{x,y} = \frac{E((x - \mu_x)(y - \mu_y))}{\sigma_x \sigma_y} \tag{3-6}$$

where E is the expectation value operator, μ_x and μ_y are expected values for variables x and y, σ_x and σ_y are standard deviations for x and y.

The $P_{x,y}$ values between all target sensor locations could be calculated using the temperature distributions at 1152 nodes through the first FE simulation as shown in Fig. 3-4.

Since the K-means algorithm aims at minimizing an objective function, $1-P_{x,y}$ between nodes should be used rather than $P_{x,y}$. Specifically, if there is a high correlation between two nodes with respect to the temperature response to input changes, the $P_{x,y}$ value is high and thus $1-P_{x,y}$ becomes small. Therefore, by minimizing $1-P_{x,y}$ as an objective function, the K-means algorithm tries to group nodes that show the strong correlation in thermal response sensitivity to input changes.

To avoid an inherent problem of the K-means algorithm namely, that clustering results depend on initial cluster centroids and which frequently results in suboptimal clusters (i.e., local minima), 25 replications, each with a new set of initial centroid positions were carried out to find a global minimum for clustering. As an indicator of how similar a node is to nodes in its own cluster compared to nodes in other clusters, the Silhouette coefficient (SC) value [54] was adopted. Its definition is:

$$s(i) = \frac{b(i) - w(i)}{\max\{b(i), w(i)\}} \tag{3-7}$$

with

$$b(i) = \min_{k}\{B(i,k)\} \tag{3-8}$$

where $w(i)$ is the average distance from the i^{th} node to the other nodes in its own cluster, $B(i,k)$ is the average distance from the i^{th} node to nodes in another cluster k.

This value varies from +1 to −1. A value of $s(i) = 1$ indicates points in this cluster are very far from neighboring clusters. For example, if the number of clusters is equal to the number of objects (nodes), $w(i) = 0$ for all i, which makes $s(i) = 1$. The value of 0 indicates points that are not distinctly in one cluster or another; −1 indicates points that are probably assigned to the wrong cluster.

To assess the clustering quality, the average Silhouette coefficient (ASC) value in Eq. (3-7) for all points in a given cluster was calculated. The relationship between the ASC value and clustering quality is shown in Table 3-3 [54].

Table 3-3 Cluster quality measured by average Silhouette coefficient values

Average Silhouette coefficient value (ASC)	Cluster quality
$0.7 < \text{ASC} \leq 1.0$	Good
$0.5 < \text{ASC} \leq 0.7$	Medium
$0.25 < \text{ASC} \leq 0.5$	Low
$\text{ASC} \leq 0.25$	Absence of cluster structure

Four cases, k=2, k=4, k=6 and k=8 were considered as a predefined number of clusters because there is no general theoretical solution to find the optimal number of clusters for any given data set. Figure 3-7 shows a SC value of each node in the cases of k=2 and k=4. The calculated ASC for each cluster in k=2 is $\text{ASC} = \{\overline{s}(1\text{st})\ \ \overline{s}(2\text{nd})\} = \{0.5\ \ 0.77\}$ and that for k=4 is $\text{ASC} = \{\overline{s}(1\text{st})\ \ \overline{s}(2\text{nd})\ \ \overline{s}(3\text{rd})\ \ \overline{s}(4\text{th})\ \} = \{0.40\ \ 0.81\ 0.61\ \ 0.34\}$. Compared to the clusters for k=2, the 1st and 4th clusters for k=4 have relatively low values of the ASC. This is possible because it could be difficult to have a strong clustering quality for all 1152 nodes with a fixed cluster number. However, since the mean of the four ASC value is 0.541, this is greater than the lower value (0.5) for medium clustering quality, it can thus be said that the overall clustering quality for k=4 is moderate.

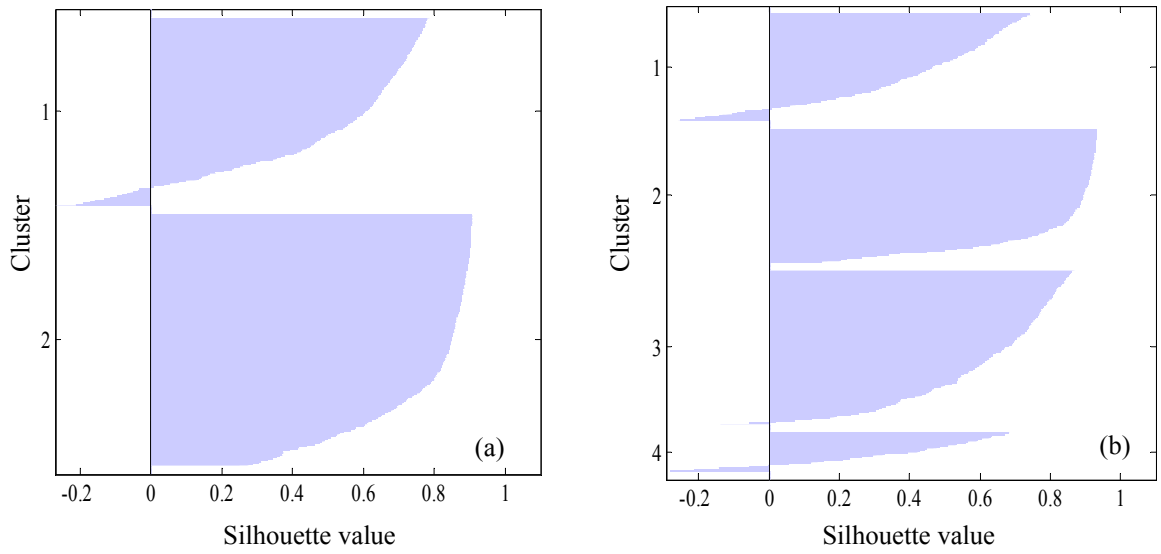


Figure 3-7 Silhouette plots in cases of k=2 (a) and k=4 (b)

Figure 3-8 shows a SC value of each node in the cases of k=6 and k=8. The ASC values per each cluster in k=6 and k=8 are $\{\overline{s}(1\text{st})\ \overline{s}(2\text{nd})\ \overline{s}(3\text{rd})\ \overline{s}(4\text{th})\ \overline{s}(5\text{th})\ \overline{s}(6\text{th})\} = \{0.47\ 0.43\ 0.58\ 0.42\ 0.36\ 0.81\}$ and $\{\overline{s}(1\text{st})\ \overline{s}(2\text{nd})\ \overline{s}(3\text{rd})\ \overline{s}(4\text{th})\ \overline{s}(5\text{th})\ \overline{s}(6\text{th})\ \overline{s}(7\text{th})\ \overline{s}(8\text{th})\} = \{0.42\ 0.44\ 0.65\ 0.47\ 0.47\ 0.46\ 0.87\ 0.37\}$, respectively. Despite some low ASC values (5th cluster for k=6 and 8th cluster for k=8), the overall clustering quality for all clusters in k=6 and k=8 is acceptable, since the mean for the ASC values is 0.51 in both cases.

- Clustering using a ratio of temperature

The Pearson correlation coefficient of the K-means could be a good criterion for clustering because it indicates the strength of the linear relationship between two nodes and therefore a linear relationship could be used as a basis of similarity for clustering. However, a high correlation coefficient value cannot always guarantee a linear relationship. This is because the Pearson correlation coefficient value does not completely characterize a linear relationship (e.g., an outlier is enough to produce a high correlation coefficient despite an actually poor relationship).

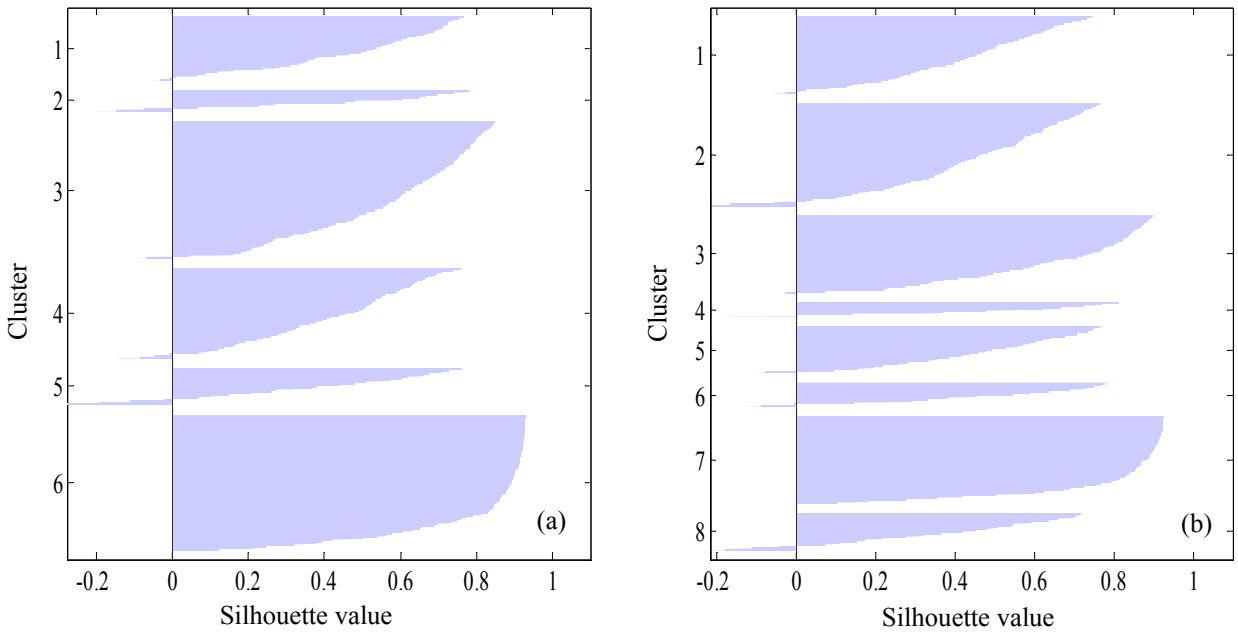


Figure 3-8 Silhouette plots in cases of k=6 (a) and k=8 (b)

In addition to this, a necessity for another criterion for clustering is raised due to the inherent disadvantage of K-means that it is difficult to determine the optimal number of clusters in advance, and thus the number of clusters is determined subjectively.

To overcome these limitations of K-means, a constant ratio of the temperature between nodes was considered as another clustering criterion. Using a constant ratio, nodes are clustered into groups with respect to a strong linear relationship with which an estimation of temperature response of less sensitive nodes using the most sensitive node in the same group is available. In order to use the ratio of the temperature, it is important that the ratio between nodes should stay relatively constant during the simulation time. Therefore, the following tolerance rate has a function of an indicator to judge how constant the ratio is.

$$Tol = \frac{r_{\max} - r_{\min}}{r_{mean}} \times 100 \ (\%) \tag{3-9}$$

where Tol is the tolerance rate. $r_{\max}$, $r_{\min}$, r_{mean} are respectively the maximum, minimum, mean values of the temperature ratio between two nodes during the simulation.

When the tolerance rate between two nodes is less than a defined cut-off value, the relationship between these two nodes is regarded as constant and thus these are clustered in the same group. If only $r_{\max}$ and $r_{\min}$ are used for the definition of the tolerance rate, the ratio of temperature responses at insensitive nodes (e.g., nodes far from the cavity and cooling channels) is more likely to be considered constant, compared to that at sensitive nodes which are our interest from the aspect of optimal sensor locations. To avoid this problem, the difference between $r_{\max}$ and $r_{\min}$ is normalized by r_{mean} in the definition.

As a cut-off value of tolerance rate for a constant ratio, 2.6%, 1.7%, 1.5% and 1.4% were selected by trial and error to generate 2, 4, 6 and 8 clusters, respectively.

3.4.2 Sensitivity analysis

After clustering nodes, the following step was taken to find the most sensitive node to input changes as a representative sensor location in each cluster.

The sensitivity coefficient, $X_{i,m}$ in Eq. (3-3) obtained through the second FE simulation was used to indicate the thermal response sensitivity at each node. In the profile of sensitivity coefficient during one cycle time (see Fig. 3-5-(b) and Fig. 3-6-(b)), the maximum absolute values of $X_{i,m}$ (minimum in Fig. 3-5-(b) and maximum in Fig.3-6-(b)) were utilized as an index to find a representative node in each cluster. Therefore, a selected representative node is the most sensitive to both the flow rate and the water temperature in each cluster.

3.5 Identified sensor locations

3.5.1 Comparison between K-means and temperature ratio methods

- Consistency in generating sensor locations

After applying the sensitivity analysis, optimal sensor locations with K-means for the cases of *k*=2, 4 and *k*=6, 8 are presented in Fig. 3-9 and 3-10 respectively. Numbering shown in the figures is for assigning target nodes from node 1 (i.e., Mo1) to node 1152 (i.e., Mo1152).

Through a comparison of optimal sensor locations between the four cases, it is observed that when the required number of sensors increases, the K-means with sensitivity analysis generates additional sensor locations inconsistently, i.e., all sensor locations required for a small number of sensors are not always included as ones for a larger number of sensors (e.g., only node Mo483 is common in all cases). This is because nodes belonging to the same cluster in one case are not always grouped into the same cluster in other cases; therefore a representative node per each cluster cannot be maintained as the most sensitive node in each cluster in the case of different number of clusters.
Figures 3-11 and 3-12 show optimal sensor locations obtained by using the temperature ratio method with sensitivity analysis.

Sensor locations in the case of *k*=8 (Fig. 3-12-(b)) include all sensor locations in the cases of *k*=2, 4, 6 (Fig. 3-11-(a), Fig. 3-11-(b), Fig. 3-12-(a), respectively) as well as newly generated locations, Mo292, Mo677 for *k*=8. This is due to a feature of the ratio method that nodes in different clusters are

seldom mingled with each other in order to form a new cluster given an increase in the number of clusters.

As seen in Fig. 3-9 – 3-12, optimal sensor locations generated by two methods are quite different. For example, only one node (Mo483 for each case) is commonly generated in both methods. In the following, the sensor locations obtained from the two methods are compared from different perspectives.

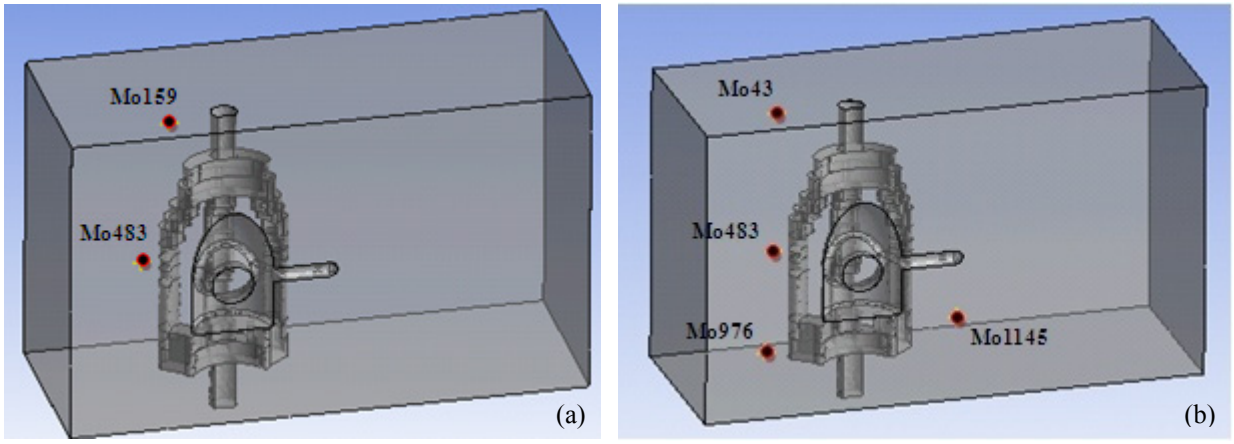


Figure 3-9 Optimal sensor locations with K-means in cases of *k*=2 (a) and *k*=4 (b)

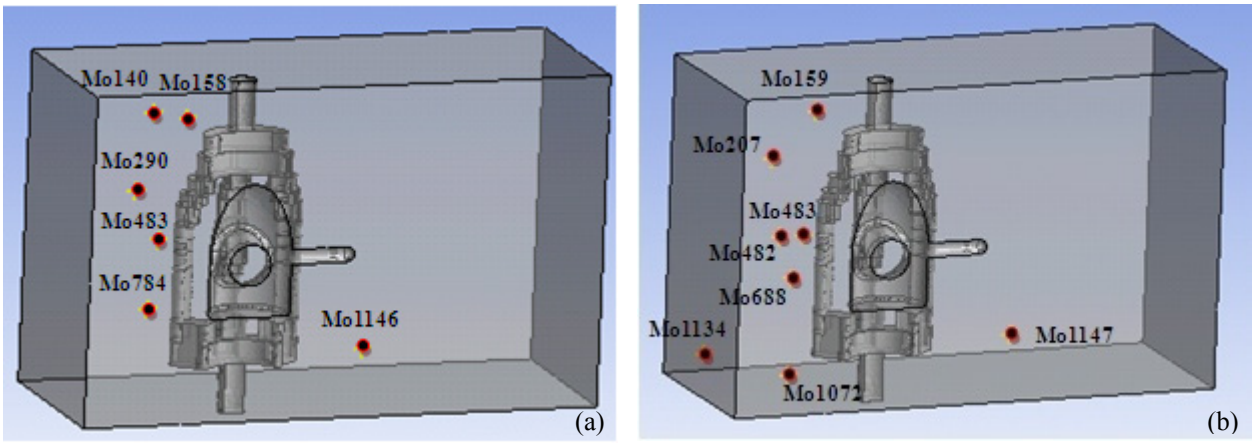


Figure 3-10 Optimal sensor locations with K-means in cases of *k*=6 (a) and *k*=8 (b)

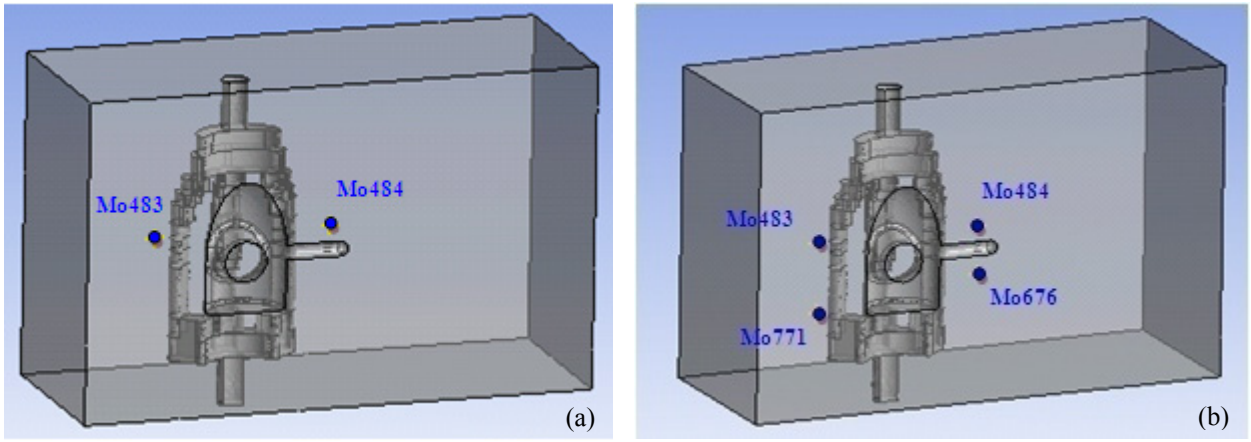


Figure 3-11 Optimal sensor locations with temperature ratio in cases of *k*=2 (a) and *k*=4 (b)

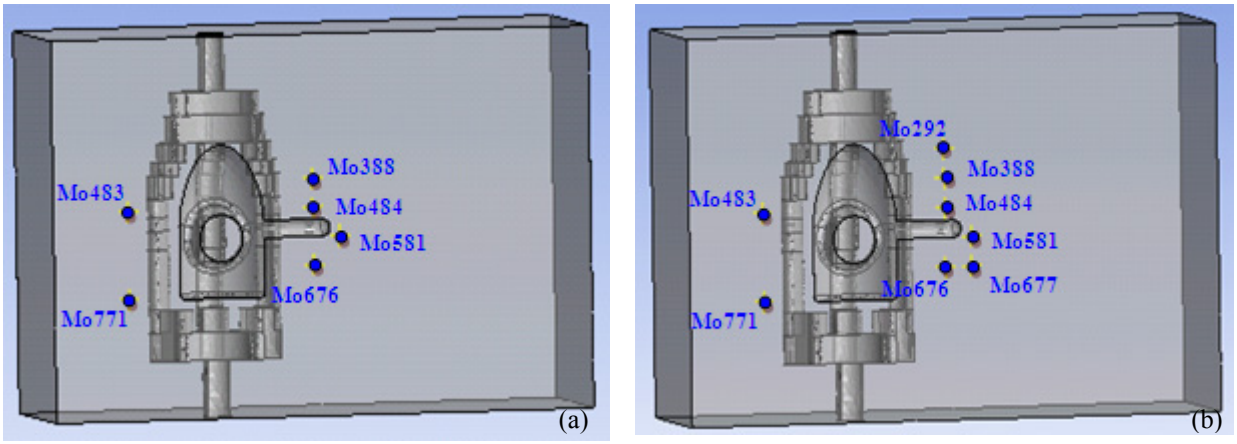


Figure 3-12 Optimal sensor locations with temperature ratio in cases of *k*=6 (a) and *k*=8 (b)

- Sensitivity of sensor locations to input changes

Table 3-4 – 3-7 describe the maximum absolute values of sensitivity coefficient to inputs (flow rate and temperature of coolant) at selected sensor locations using the two methods. In all cases of *k*=2, 4, 6 and 8, sensor nodes obtained from the temperature ratio method have higher sensitivity coefficient values to both inputs. This difference in the sensitivity can be easily identified by comparing the average values of the maximum $\left|X_{i,m}\right|$.

Therefore, the temperature ratio method is superior to the K-means in light of generating more sensitive sensor locations to input changes which have more informative data to monitor the temperature change during the injection moulding process.

Table 3-4 Sensitivity coefficients at optimal sensor locations in case of *k*=2

K-means (*k*=2)			Temp. Ratio (*k*=2)		
Node	$\lvert X_{i,m} \rvert$ to flow rate	$\lvert X_{i,m} \rvert$ to temp.	Node	$\lvert X_{i,m} \rvert$ to flow rate	$\lvert X_{i,m} \rvert$ to temp.
Mo159	0.6004	4.6	–		
Mo483	1.2	9.4	**Mo483**	1.2	9.4
–			Mo484	1	7.6
Mean	**0.9002**	**7**	**Mean**	**1.1**	**8.5**

Table 3-5 Sensitivity coefficients at optimal sensor locations in case of *k*=4

K-means (*k*=4)			Temp. Ratio (*k*=4)		
Node	$\lvert X_{i,m} \rvert$ to flow rate	$\lvert X_{i,m} \rvert$ to temp.	Node	$\lvert X_{i,m} \rvert$ to flow rate	$\lvert X_{i,m} \rvert$ to temp.
Mo43	0.6004	5.4	–		
Mo483	1.2	9.4	**Mo483**	1.2	9.4
Mo976	0.8002	8	Mo484	1	7.6
Mo1145	0.2002	1.6	Mo676	1	7.8
–			Mo771	1.2	9.2
Mean	**0.7002**	**6.1**	**Mean**	**1.1**	**8.5**

Table 3-6 Sensitivity coefficients at optimal sensor locations in case of *k*=6

K-means (*k*=6)			Temp. Ratio (*k*=6)		
Node	$\lvert X_{i,m} \rvert$ to flow rate	$\lvert X_{i,m} \rvert$ to temp.	Node	$\lvert X_{i,m} \rvert$ to flow rate	$\lvert X_{i,m} \rvert$ to temp.
Mo140	0.8002	6.4	–		
Mo158	0.6004	4.4	–		
Mo290	1	7	Mo388	1	7.4
Mo483	1.2	9.4	**Mo483**	1.2	9.4
Mo784	1	9	Mo484	1	7.6
Mo1146	0.2002	1.2	Mo581	0.8002	5.6
–			Mo676	1	7.8
–			Mo771	1.2	9.2
Mean	**0.8001**	**6.2333**	**Mean**	**1.0334**	**7.8333**

Table 3-7 Sensitivity coefficients at optimal sensor locations in case of *k*=8

K-means (*k*=8)			Temp. Ratio (*k*=8)		
Node	$\lvert X_{i,m} \rvert$ to flow rate	$\lvert X_{i,m} \rvert$ to temp.	Node	$\lvert X_{i,m} \rvert$ to flow rate	$\lvert X_{i,m} \rvert$ to temp.
Mo159	0.6004	4.6	–		
Mo207	0.8002	6.2	Mo292	0.9998	7
Mo482	1	7.6	Mo388	1	7.4
Mo483	1.2	9.4	**Mo483**	1.2	9.4
Mo688	1.2	9	Mo484	1	7.6
Mo1072	0.8002	7.4	Mo581	0.8002	5.6
Mo1134	0.4004	2.4	Mo676	1	7.8
Mo1147	0.2002	0.8002	Mo677	0.8002	5.8
–			Mo771	1.2	9.2
Mean	**0.775**	**5.925**	**Mean**	**1**	**7.475**

- Lowest mutual interaction

In this study, grouping target nodes is the first step to find out optimal sensor locations since it generates distinguishable groups according to similarity. Therefore, each representative node in each different group should provide unique information of the temperature profile. For the uniqueness, it is important to determine sensor locations to be characterized by having the lowest mutual interaction. Accordingly, the lowest mutual interaction could be another criterion to judge the quality and effectiveness of clustering between the two methods.

The study in [55] exploits the sensor locations to have lowest mutual interaction and highest sensitivity to inputs by applying singular value decomposition (SVD) using the sensitivity gain matrix (partial derivative of each output variable with respect to each input variable). Through this approach, the identified second most sensitive location interacts least with the most sensitive location. The third most sensitive location has low interaction with the first and second most sensitive locations.

By following the method proposed in the study by Moore [55], the sensitivity gain matrix has 1152 (no. of nodes) ×2 (two inputs) dimension. Then, SVD produces two nonzero singular values by two inputs and thus the two most sensitive locations which exhibit the least possible interactions with each other. However, since only two locations can be generated due to having two nonzero singular values, this approach is not appropriate for dealing with four, six and eight sensor locations. Therefore, principal component analysis (PCA) was applied instead of SVD.

PCA is a closely related statistical algorithm to SVD because PCA uses the SVD technique to determine the PC score and loading matrix (see Eq. (3-11)). PCA [56] is a way of identifying patterns in data, and highlighting their similarities and differences through an orthogonal linear transformation. Using this method, the data can be transformed to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (first principal component: a direction of best fit going through the middle of data), the second greatest variance on the second coordinate (second principal component: a direction of the second best fit) and so on. Since these principal components are orthogonal vectors, sensor locations belonging to different principal components can be interpreted to have low interaction between those.

By referring to the PCA procedure in [57], the gain matrix $\hat{K}$ is defined by using temperature profile at 1152 nodes during 360 seconds obtained from the first FEA:

$$\hat{\boldsymbol{K}}(360\times 1152)=\begin{bmatrix} \boldsymbol{T}_1\big|_{\boldsymbol{t}=1\text{sec}} & \cdots & \boldsymbol{T}_{\boldsymbol{i}}\big|_{\boldsymbol{t}=1\text{sec}} & \cdots & \boldsymbol{T}_{1152}\big|_{\boldsymbol{t}=1\text{sec}} \\ \vdots & & \vdots & & \vdots \\ \boldsymbol{T}_1\big|_{\boldsymbol{t}=\boldsymbol{j}\text{sec}} & \cdots & \boldsymbol{T}_{\boldsymbol{i}}\big|_{\boldsymbol{t}=\boldsymbol{j}\text{sec}} & \cdots & \boldsymbol{T}_{1152}\big|_{\boldsymbol{t}=\boldsymbol{j}\text{sec}} \\ \vdots & & \vdots & & \vdots \\ \boldsymbol{T}_1\big|_{\boldsymbol{t}=360\text{sec}} & \cdots & \boldsymbol{T}_{\boldsymbol{i}}\big|_{\boldsymbol{t}=360\text{sec}} & \cdots & \boldsymbol{T}_{1152}\big|_{\boldsymbol{t}=360\text{sec}} \end{bmatrix} \tag{3-10}$$

where $\boldsymbol{T}_{\boldsymbol{i}}\big|_{\boldsymbol{t}=\boldsymbol{j}\text{sec}}$ denotes the temperature of the i^{th} node at the j second, and the time interval used is 1 second.

By scaling data in $\hat{K}$ such that each column (for each node) is normalized to zero mean and unit variance, the gain matrix $\hat{K}$ becomes the normalized gain matrix $\tilde{K}$. Then the normalized gain matrix $\tilde{K}$ is decomposed as:

$$\tilde{K}=T_S P^T \tag{3-11}$$

where $T_S(360\times s)$ is the score matrix, $P(1152\times s)$ is the orthonormal loading matrix, and s is the number of principal components.

Since the numbers of sensors (clusters) considered are two (k=2), four (k=4), six (k=6) and eight (k=8), the most significant two, four, six and eight orthogonal principal components are selected from P, respectively. However, it is required to first verify the significance of these principal components. The significance of each principal component can be represented by the eigenvalues of the covariance matrix of $\tilde{K}$, which are presented in Table 3-8.

Table 3-8 Eigenvalue for each principal component

PC	Eigenvalue
1st	1066.11
2nd	71.02
3rd	9.51
4th	6.67
5th	1.09
6th	0.58
7th	0.11
8th	0.036
9th	0.024
10th	0.017
11th	0.004
12th	0.003
13th	0.001
14th	0.0007
⋮	⋮
1152th	0

Even though the eigenvalue for the 8th principal component (PC) is somewhat small, its value cannot be ignored, compared to quite small values of the remaining PCs after the 8th PC. Therefore, the top two PCs (1st - 2nd PC), the top four PCs (1st – 4th PC), the top six PCs (1st – 6th PC) and the top eight PCs (1st - 8th PC) out of all PCs in P can be considered as significant for the cases of k=2, k=4, k=6 and k=8, respectively.

Then, the interaction between sensor locations identified by the two clustering methods in the case of k=2 is verified using the top two significant PCs (1st - 2nd PC). Because the i^{th} element's loading in each PC presents a measure of the i^{th} node's contribution to each PC, selected sensor locations in Fig. 3-9-(a) (K-means) and Fig. 3-11-(a) (Temp. ratio) are classified into the PC among 2 PCs in which

each sensor location has an largest absolute value of loading, i.e., shows a significant contribution. Classification results are reported in Table 3-9.

Sensor locations by the K-means are more evenly distributed into two PCs (1^{st}, 2^{nd} PCs) whereas sensor locations by the temperature ratio method are in one PC (2^{nd} PC). Therefore, the K-means is slightly better than the temperature ratio method from the viewpoint of low interaction between sensor locations.

Classification of sensor locations shown from the K-means and the Temperature ratio method in the cases of k=4, 6, 8 are presented in Table 3-10 – 3-12. From the tables, it is observed that sensor locations by the K-means have a significant contribution on more PCs (three PCs in k=4, four PCs k=6, five PCs in k=8) than those by the temperature ratio method (two PCs in k=4, three PCs k=6, three PCs in k=8;). So, K-means is better than the temperature ratio method more or less in term of generating sensor locations to have lowest mutual interaction.

Table 3-9 Classification of selected sensor locations using PCA in case of k=2

K-means (k=2)		Temp. Ratio (k=2)	
Node	Corresponding PC	Node	Corresponding PC
Mo159	1^{st} PC	–	
Mo483	2^{nd} PC	**Mo483**	2^{nd} PC
–		Mo484	2^{nd} PC

Table 3-10 Classification of selected sensor locations using PCA in case of *k*=4

K-means (*k*=4)		Temp. Ratio (*k*=4)	
Node	Corresponding PC	Node	Corresponding PC
Mo43	3rd PC	–	
Mo483	2nd PC	**Mo483**	2nd PC
Mo976	2nd PC	Mo484	3rd PC
Mo1145	1st PC	Mo676	3rd PC
–		Mo771	2nd PC

Table 3-11 Classification of selected sensor locations using PCA in case of *k*=6

K-means (*k*=6)		Temp. Ratio (*k*=6)	
Node	Corresponding PC	Node	Corresponding PC
Mo140	3rd PC	–	
Mo158	5th PC	–	
Mo290	2nd PC	Mo388	5th PC
Mo483	2nd PC	**Mo483**	2nd PC
Mo784	2nd PC	Mo484	3rd PC
Mo1146	1st PC	Mo581	3rd PC
–		Mo676	5th PC
–		Mo771	2nd PC

Table 3-12 Classification of selected sensor locations using PCA in case of *k*=8

K-means (*k*=8)		Temp. Ratio (*k*=8)	
Node	Corresponding PC	Node	Corresponding PC
Mo159	5th PC	–	
Mo207	4th PC	Mo292	2nd PC
Mo482	7th PC	Mo388	5th PC
Mo483	2nd PC	**Mo483**	2nd PC
Mo688	2nd PC	Mo484	3rd PC
Mo1072	3rd PC	Mo581	3rd PC
Mo1134	7th PC	Mo676	5th PC
Mo1147	7th PC	Mo677	3rd PC
–		Mo771	2nd PC

- Determination of optimal sensor number

K-means has an inherent problem that the number of clusters needs to be specified. Therefore, it cannot solve the problem of determining the optimal number of sensors. However, the temperature ratio method automatically sets up the number of sensors according to the degree of tolerance rate in clustering. Namely, if a cut-off tolerance rate is decreased, the accuracy in clustering is increased. Then, the number of nodes grouped in the same cluster by a constant ratio with a strict accuracy decreases. Subsequently, the number of clusters or number of sensors increases. For instance from Fig. 3-11 and 3-12, reducing a cut-off tolerance from 2.6% to either 1.7% or 1.5% or 1.4% produces either two (Mo676, Mo771) or four (Mo388, Mo581, Mo676, Mo771) or six (Mo 292, Mo388, Mo581, Mo676, Mo677, Mo771) additional sensor locations.

Thus, the optimal number of sensors can be controlled by a cut-off tolerance rate to be desired using the temperature ratio method.

3.5.2 Estimation of temperature response using sensor locations

- Selection of estimation nodes and test nodes for temperature estimation

Another criterion for optimal sensor locations is how accurately selected sensors can estimate the temperature response of nodes where the installation of a sensor would be challenging due to geometric constraints (e.g., proximity of sensor to a cooling channel). The following two sets of nodes (Set1 (6 nodes), Set 2 (6nodes)) in Fig. 3-13 were evaluated for such proxy measurements.

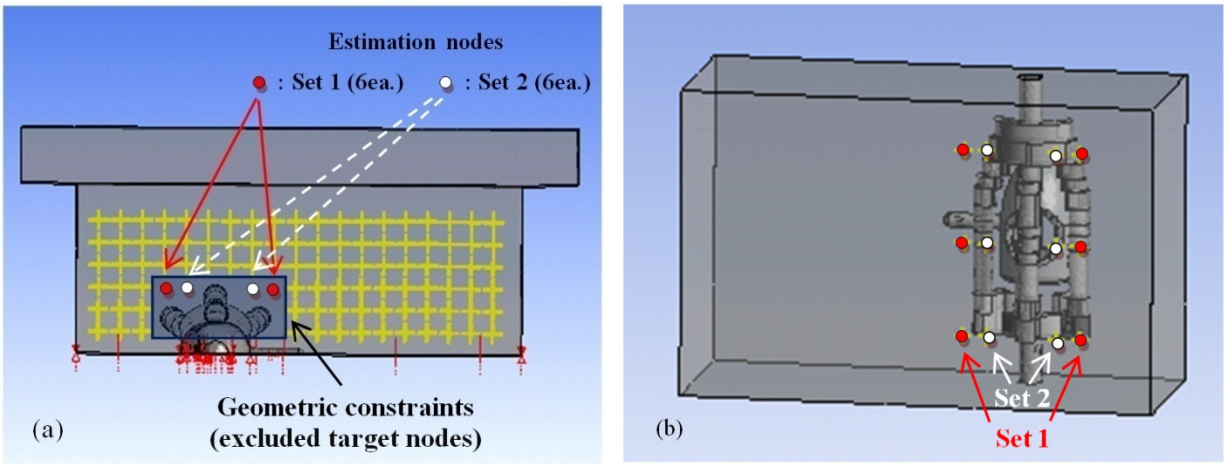


Figure 3-13 Nodes requiring estimation of temperature response using sensor locations ((a): top view and (b): rear view)

For the estimation of these two sets of nodes, selected test locations as well as locations determined by K-means and the temperature ratio method were used to compare the accuracy of estimation. Figure 3-14-(a) shows two sets (labelled Test1, Test2) of selected test locations very close to the estimation nodes (Set1, Set2). For each set, 2, 4, 6 and 8 locations among total 8 locations were considered for the comparison of estimation accuracy. Table 3-13 presents the sensor locations in each set (Fig. 3-14-(b)) which were used for the estimation of temperature response at Set1 and Set2 for the cases of k=2, 4, 6 and 8.

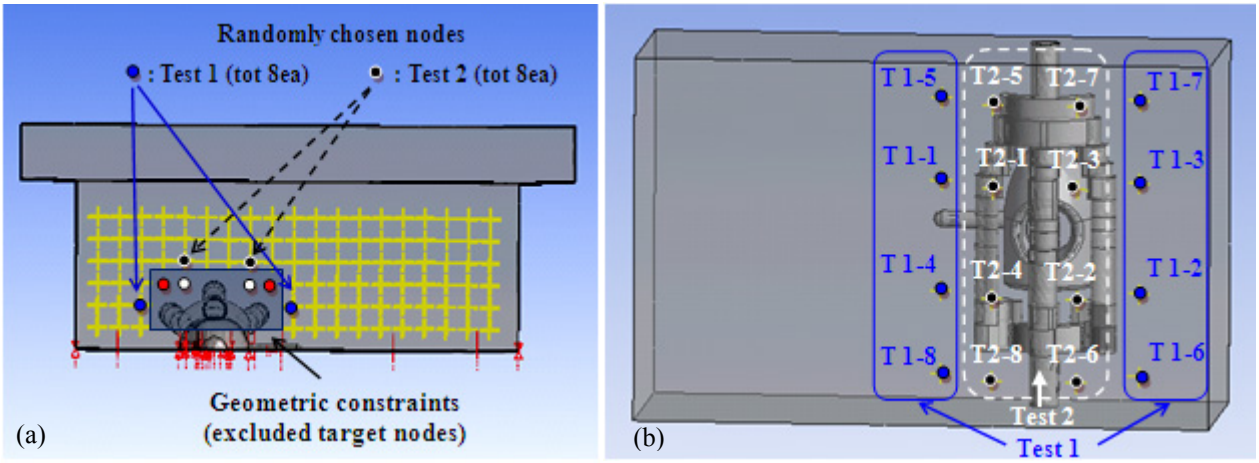


Figure 3-14 Selected test sensor locations (Test1 and Test2) used for temperature estimation ((a): top view and (b): rear view)

Table 3-13 Selected test sensor locations used for temperature estimation in *k*=2, 4, 6 and 8

No. of clusters (sensors)	Selected test sensor locations	
	Test1	Test2
k=2	T1-1, T1-2	T2-1, T2-2
k=4	T1-1, T1-2, T1-3, T1-4	T2-1, T2-2, T2-3, T2-4
k=6	T1-1, T1-2, T1-3, T1-4 T1-5, T1-6	T2-1, T2-2, T2-3, T2-4 T2-5, T2-6
k=8	T1-1, T1-2, T1-3, T1-4 T1-5, T1-6, T1-7, T1-8	T2-1, T2-2, T2-3, T2-4 T2-5, T2-6, T2-7, T2-8

- NARX model

For the estimation of temperature, a neural network (NN) technique with a NARX (Nonlinear Auto Regressive with eXogenous) model was selected because it is a powerful class of nonlinear dynamic models. In the NARX model, embedded memory is required to include temporal changes which have

an impact on the dynamic response of the system. The inputs to the NARX model are composed of past values of the system inputs and outputs as follows:

$$\begin{aligned} y(t) &= y_N(t) + e(t) \\ &= f(y(t-1), y(t-2), \ldots, y(t-n_y), u(t-1), u(t-2), \ldots, u(t-n_u)) + e(t) \end{aligned} \tag{3-12}$$

where $y(t)$, y_N, $u(t)$ and $e(t)$ are the system output, neural network output, system input and error between $y(t)$ and y_N at time t, respectively. n_y and n_u are the number of past outputs and past inputs, respectively and are referred to as the model orders.

When applying the NARX model to the problem in hand, the outputs in the NARX model are the temperatures at 6 nodes (Set1 or Set2 in Fig. 3-13) for estimation, and the inputs to the NARX model are the flow rate and temperature of the coolant, and the temperatures at the sensor locations (by the K-means, temperature ratio method, Test1 nodes, Test2 nodes). Next, it is important to determine the correct number of past outputs and past inputs (i.e., model orders) in the NARX model of Eq. (3-12). The method to determine the correct number of past outputs and past inputs (i.e., model orders) is introduced in the next section.

-Determination of model orders

The correct model order is necessary for a neural model to reliably represent a system's dynamics. However, model orders are not provided in most real-world systems and thus they need to be estimated before the system identification [58], [59]. Model order determination for the model in Eq. (3-12) was carried out based on the Lipschitz criterion [59], which has been proposed to identify the best orders of input-output models for unknown nonlinear dynamic systems. Because this approach is based only on measured input-output data for the system, it is useful for this study where the NN is used for finding the dynamics of the systems by estimating input-output data.

To estimate the model orders, the following Lipschitz quotient [59] is required to be calculated. For input-output data pattern $[x_k, y_k]$ where k =1, 2, …, N (total number of patterns), the Lipschitz quotient q_{ij} for a one-dimensional input is defined as:

$$q_{ij} = \frac{\left|y_i - y_j\right|}{\left|x_i - x_j\right|},\ (i \neq j) \tag{3-13}$$

where $\left|x_i - x_j\right|$ is the distance between two vectors in the input space, and $\left|y_i - y_j\right|$ is the distance between the output vectors corresponding to the respective input vectors.

The Lipschitz quotient for the multi-dimensional inputs could be applied to a MIMO system by decomposing the MIMO system into MISO sub-systems. For instance, the NARX model in (3-12) estimating the temperature at Set1 (6 nodes in Fig. 3-13-(b)) using the temperature profile at 4 sensor locations by the K-means (Fig. 3-9-(b)) is a MIMO system having 6 output variables (temperatures at 6 nodes in Set1) and 6 input variables (flow rate and temperature of the coolant, and temperatures at 4 nodes by the K-means). This MIMO system can be decomposed into 6 MISO sub-systems expressed as:

$$\begin{aligned} y_r(t) = f_r(&y_1(t-1),\ldots,\ y_1(t-n_{y_1}^r),\ y_2(t-1),\ldots,\ y_2(t-n_{y_2}^r),\ y_3(t-1),\ldots,\ y_3(t-n_{y_3}^r),\\ &y_4(t-1),\ldots,\ y_4(t-n_{y_4}^r), y_5(t-1),\ldots,\ y_5(t-n_{y_5}^r),\ y_6(t-1),\ldots,\ y_6(t-n_{y_6}^r),\\ &u_1(t-1),\ldots,\ u_1(t-n_{u_1}^r),\ u_2(t-1),\ldots,\ u_2(t-n_{u_2}^r),\ u_3(t-1),\ldots,\ u_3(t-n_{u_3}^r),\\ &u_4(t-1),\ldots,\ u_4(t-n_{u_4}^r),\ u_5(t-1),\ldots,\ u_5(t-n_{u_5}^r),\ u_6(t-1),\ldots,\ u_6(t-n_{u_6}^r)) + e_r(t) \end{aligned} \tag{3-14}$$

where $n_{u_1}^r$, $n_{u_2}^r$, $n_{u_3}^r$, $n_{u_4}^r$, $n_{u_5}^r$, $n_{u_6}^r$ are the model orders (or time delays) for u_1 (coolant temperature), u_2 (coolant flow rate), u_3, u_4, u_5, u_6 (temperatures at 4 nodes by the K-means) in the r^{th} sub-system (i.e., y_r), respectively. $n_{y_1}^r$, $n_{y_2}^r$, n_{y3}^r, n_{y4}^r, n_{y5}^r, n_{y6}^r are the model orders for the y_r (temperature at r^{th} node in Set1 where r=1, 2, 3, 4, 5, 6).

Then, the Lipschitz quotient ${}^r q_{ij}^{(n)}$ for the r^{th} sub-system is represented as:

$$
{}^{r}q_{ij}^{(n)} = {}^{r}q_{ij}^{(n_{y1}^{r}+n_{y2}^{r}+n_{y3}^{r}+n_{y4}^{r}+n_{y5}^{r}+n_{y6}^{r}+n_{u1}^{r}+n_{u2}^{r}+n_{u3}^{r}+n_{u4}^{r}+n_{u5}^{r}+n_{u6}^{r})} = \frac{\left|y_{ri}-y_{rj}\right|}{\sqrt{(x_{1i}-x_{1j})^2+\ldots+(x_{ni}-x_{nj})^2}}
$$

$$
= \frac{\left|y_{ri}-y_{rj}\right|}{\sqrt{\begin{array}{l}
(y_{1i}(t-1)-y_{1j}(t-1))^2+\ldots+(y_{1i}(t-n_{y_1}^{r})-y_{1j}(t-n_{y_1}^{r}))^2+(y_{2i}(t-1)-y_{2j}(t-1))^2+\ldots+(y_{2i}(t-n_{y_2}^{r})-y_{2j}(t-n_{y_2}^{r}))^2 \\
+(y_{3i}(t-1)_i-y_{3j}(t-1)_j)^2+\ldots+(y_{3i}(t-n_{y_3}^{r})-y_{3j}(t-n_{y_3}^{r}))^2+(y_{4i}(t-1)_i-y_{4j}(t-1)_j)^2+\ldots \\
+(y_{4i}(t-n_{y_4}^{r})-y_{4j}(t-n_{y_4}^{r}))^2+(y_{5i}(t-1)_i-y_{5j}(t-1)_j)^2+\ldots+(y_{5i}(t-n_{y_5}^{r})-y_{5j}(t-n_{y_5}^{r}))^2+ \\
(y_{6i}(t-1)_i-y_{6j}(t-1)_j)^2+\ldots+(y_{6i}(t-n_{y_6}^{r})-y_{6j}(t-n_{y_6}^{r}))^2+(u_{1i}(t-1)_i-u_{1j}(t-1))^2+\ldots+(u_{1i}(t-n_{u_1}^{r})-u_{1j}(t-n_{u_1}^{r}))^2 \\
+(u_{2i}(t-1)-u_{2j}(t-1))^2+\ldots+(u_{2i}(t-n_{u_2}^{r})-u_{2j}(t-n_{u_2}^{r}))^2+(u_{3i}(t-1)-u_{3j}(t-1))^2+\ldots+(u_{3i}(t-n_{u_3}^{r})-u_{3j}(t-n_{u_3}^{r}))^2 \\
+(u_{4i}(t-1)-u_{4j}(t-1))^2+\ldots+(u_{4i}(t-n_{u_4}^{r})-u_{4j}(t-n_{u_4}^{r}))^2+(u_{5i}(t-1)-u_{5j}(t-1))^2+\ldots+(u_{5i}(t-n_{u_5}^{r})-u_{5j}(t-n_{u_5}^{r}))^2 \\
+(u_{6i}(t-1)-u_{6j}(t-1))^2+\ldots+(u_{6i}(t-n_{u_6}^{r})-u_{6j}(t-n_{u_6}^{r}))^2
\end{array}}}
\tag{3-15}
$$

where n is the correct number of inputs. ${}^{r}q_{ij}^{(n)}$ is Lipschitz quotient for the r^{th} sub-system with r =1, 2, 3, 4, 5, 6.

From the study in [59], when one of the necessary input variables (assume x_n) is missing, the Lipschitz quotient $q_{ij}^{(n-1)}$ becomes larger than $q_{ij}^{(n)}$. On the other hand, when a redundant input variable (assume x_{n+1}) is added, the Lipschitz quotient $q_{ij}^{(n+1)}$ will be slightly smaller or larger than $q_{ij}^{(n)}$.

The following index is used to identify the optimal model order:

$$
q^{(n)} = (\prod_{k=1}^{p}\sqrt{n}q^{(n)}(k))^{1/p} \tag{3-16}
$$

where $q^{(n)}(k)$ is the k^{th} largest Lipschitz quotient among all $q_{ij}^{(n)}$ $(i \neq j \quad i,j=1,2,...,N)$ with the n input variables, p is equal to $(0.01 \sim 0.02)N$, and N is the total number of patterns.

The attractiveness of this method is that because it uses the relative magnitudes of the sequence $q^{(n)}$, the influence of low level noise does not change the qualitative characteristic of the relative magnitudes of sequence $q^{(n)}$ [59].

Because the Lipschitz criterion can be only extended to deal with MISO systems, it is necessary to find an optimal order for each MISO system using the above index, and then to find optimal orders for our MIMO system, which can be determined by a minimal combination between each order [60]. The data set from 1 to 312 second (time step: 1 second) from the first FEA was used to find each optimal order for 6 MISO sub-systems.

Figure 3-15 shows the result of the model order determination for 1st MISO sub-system (i.e., $y_1(t)$ in (3-14)) based on the Lipschitz quotient ($^1q_{ij}^{(n_{y1}^1+n_{u1}^1+n_{u2}^1+n_{u3}^1+n_{u4}^1+n_{u5}^1+n_{u6}^1+n_{y2}^1+n_{y3}^1+n_{y4}^1+n_{y5}^1+n_{y6}^1)}$ in (3-15)). For an input variable set for the Lipschitz quotient of the 1st MISO system, the past MISO system's output (i.e., $y_1(t-1)$) is first considered. Then, the past inputs (i.e., $u_1(t-1)$, $u_2(t-1)$, $u_3(t-1)$, $u_4(t-1)$, $u_5(t-1)$, $u_6(t-1)$) and the past remaining variables (i.e., $y_2(t-1)$, $y_3(t-1)$, $y_4(t-1)$ $y_5(t-1)$, $y_6(t-1)$) are used in turn as the following inputs.

For this sequence, we start to calculate the Lipschitz quotient index $^1q_{ij}^{(1+0+0+0+0+0+0+0+0+0+0+0)}$ with the first variable $y_1(t-1)$ for which the index value is equal to infinity. By adding the second input, $u_1(t-1)$, the Lipschitz quotient is $^1q_{ij}^{(1+1+0+0+0+0+0+0+0+0+0+0)} = \text{infinity}$. By setting the third, fourth and fifth inputs ($u_2(t-1)$, $u_3(t-1)$ and $u_4(t-1)$, respectively), the corresponding quotient is $^1q_{ij}^{(1+1+1+1+1+0+0+0+0+0+0+0)}$=2.664. By continuing the procedure, $^1q_{ij}^{(1+1+1+1+1+1+1+1+1+1+1+1)}$ for the 12th input ($y_6(t-1)$) shows the smallest value of 1.412. After this value, the following Lipschitz quotients are not significantly different from $^1q_{ij}^{(1+1+1+1+1+1+1+1+1+1+1+1)}$ (e.g., $^1q_{ij}^{(2+1+1+1+1+1+1+1+1+1+1+1)} = 1.438$ for the 13th input, $y_1(t-2)$). Therefore, the model order for the sub-system, $y_1(t)$ is determined as $n_{y_1}^1, n_{u_1}^1, n_{u_2}^1, n_{u_3}^1, n_{u_4}^1, n_{u_5}^1, n_{u_6}^1, n_{y_2}^1, n_{y_3}^1, n_{y_4}^1, n_{y_5}^1, n_{y_6}^1 = (1,1,1,1,1,1,1,1,1,1,1,1)$.

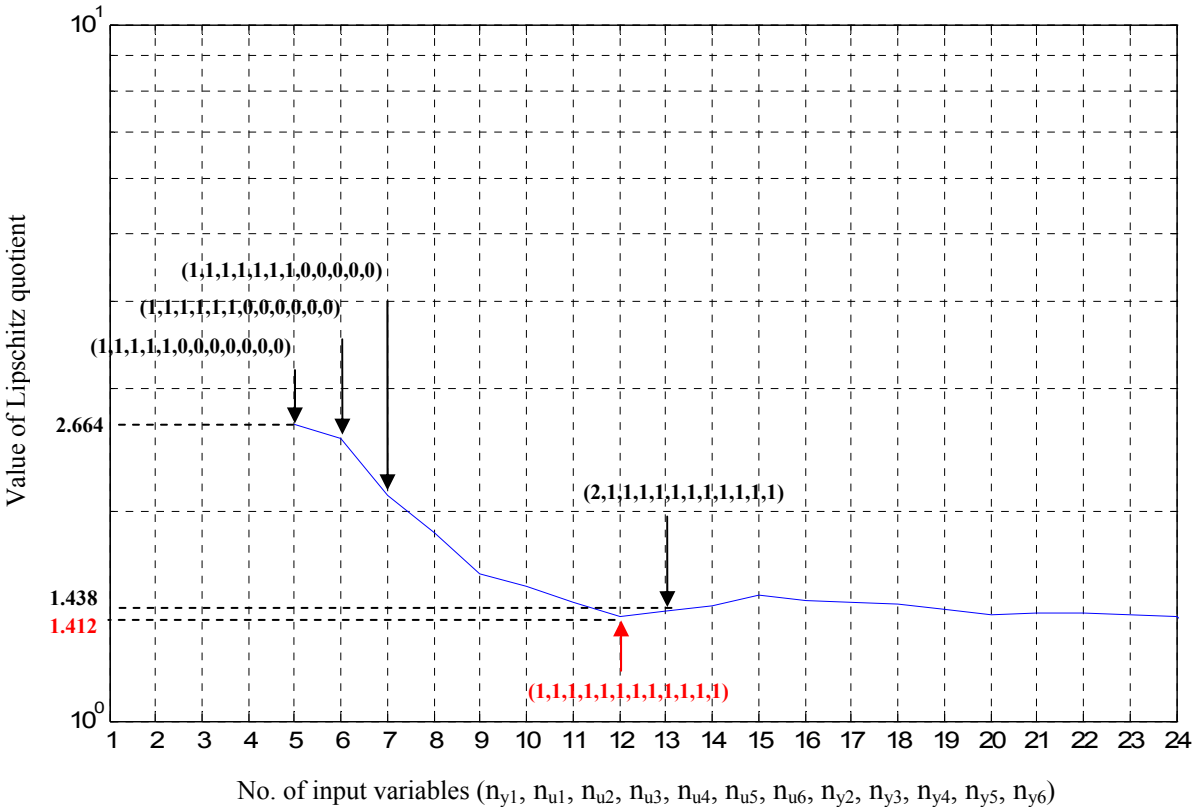


Figure 3-15 Result of model order determination using Lipschitz quotient for y_1(t)

The optimal orders $(n_{y_1}, n_{y_2}, n_{y_3}, n_{y_4}, n_{y_5}, n_{y_6}, n_{u_1}, n_{u_2}, n_{u_3}, n_{u_4}, n_{u_5}, n_{u_6})$ of each MISO system, $y_r(t)$ with $r = 1, 2, 3, 4, 5, 6$ are all (1,1,1,1,1,1,1,1,1,1,1,1). Therefore, the combined minimum combination for the optimal orders of the MIMO system is (1,1,1,1,1,1,1,1,1,1,1,1).

Therefore, it can be stated that the following NARX neural network with the optimal model orders represents the process dynamics:

$$y_N(t) = f(y_1(t-1),\ y_2(t-1),\ y_3(t-1),\ y_4(t-1),\ y_5(t-1),\ y_6(t-1),\ u_1(t-1),\ u_2(t-1),\ u_3(t-1),\ u_4(t-1),\ u_5(t-1),\ u_6(t-1)) \quad (3\text{-}17)$$

The model order of this NARX model obtained by the Lipschitz criterion is 1 second for all past outputs and past inputs.

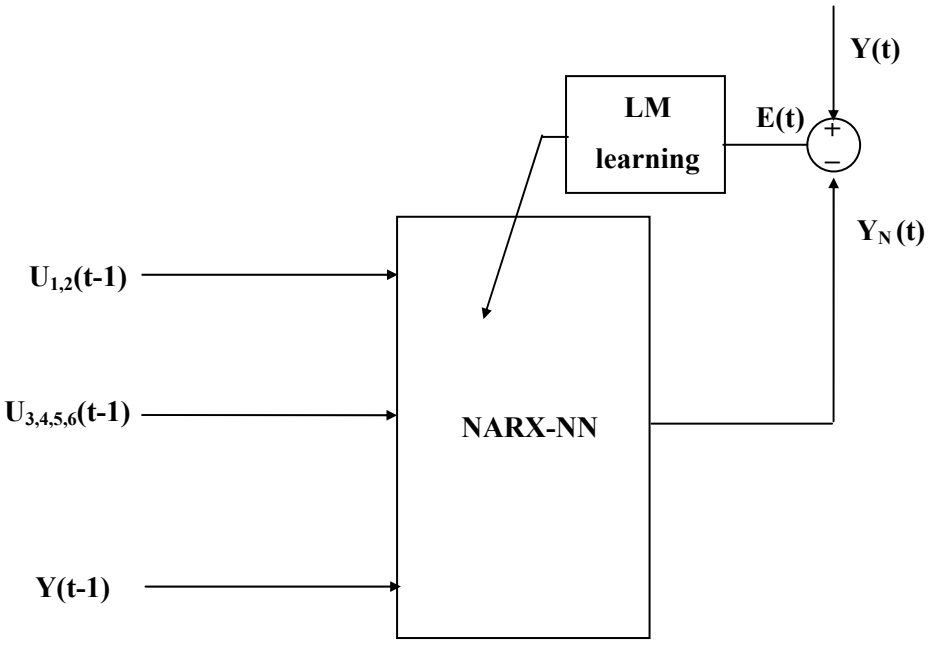
-Comparison of estimation accuracy at sensor locations

The schematic for the NARX model with the model orders determined by the Lipschitz criterion in (3-17) is presented in Fig. 3-16. Then, training the NARX neural network with model orders was conducted using the training data (1– 312 second) from the first FEA. The Levenberg-Marquardt (LM) algorithm in Section 2.3.4 and sigmoid function were used as the training algorithm and activation function, respectively. The number of hidden layer nodes was chosen by trial and error, based on the smallest mean square error (MSE) for the best estimation performance. After completing the training with 200 epochs, the NN performance (i.e., accuracy) was tested with the testing data set (313 – 360 second) from the first FEA.

Finally, the MSE of output values of the NARX model between the K-means and the temperature ratio method was compared to validate the accuracy in estimation of the temperature at nodes of Set1 and Set2. The estimation accuracy with two methods was also compared with ones using test nodes (Test1 and Test2 in Fig. 3-14).

Tables 3-14 and 3-15 present the estimation accuracy (i.e., MSE) using different sensor locations (by the K-means, temperature ratio method, Test1 nodes, Test2 nodes) for estimation nodes Set1 and Set2, respectively.

From Tables 3-14 and 3-15, it is seen that the K-means and temperature ratio method are satisfactory for the estimation accuracy for both Set1 and Set2 in the cases of 2, 4, 6 and 8 locations. However, a NN with sensor locations obtained by the temperature ratio method exhibits slightly better performance than NNs with sensor locations by K-means and Test1 nodes, Test2 nodes for all Set1 and Set2. Therefore, in terms of the accuracy in the estimation of temperature response for nodes (i.e., Set1 and Set2) near the cooling channels where locating sensors is challenging, the temperature ratio method is superior to the K-means.



- Temperature at 6 estimation nodes of Set1 which are $S_{1\text{-}1}$, $S_{1\text{-}2}$, $S_{1\text{-}3}$, $S_{1\text{-}4}$, $S_{1\text{-}5}$, $S_{1\text{-}6}$:

  $Y(t)= [y_1(t); y_2(t); y_3(t); y_4(t); y_5(t); y_6(t)]$

  $= [Ts_{1\text{-}1}(t); T_{S1\text{-}2}(t); T_{S1\text{-}3}(t); T_{S1\text{-}4}(t); T_{S1\text{-}5}(t); T_{S1\text{-}6}(t)]$

- Neural network output: $Y_N(t)$
- System inputs:

  $U_{1,2}(t)=[u_1(t)$(water temp.); $u_2(t)$(water flow rate)$]$
- Temperature at 4 sensor locations (Mo43, Mo483, Mo976, Mo1145) by K-means:

  $U_{3,4,5,6}(t)=[u_3(t); u_4(t); u_5(t); u_6(t)] =[T_{Mo43}(t); T_{Mo483}(t); T_{Mo976}(t); T_{Mo1145}(t)]$

Figure 3-16 Schematic of NARX model

Table 3-14 MSE comparison of NARX with different sensor locations for estimation nodes Set1

No. of sensors	MSE of testing data			
	Sensors locations by K-means	Sensors locations by Ratio method	Test location (Test1)	Test location (Test2)
2	0.0300	0.0137	0.0288	0.0260
4	0.0284	0.0133	0.0271	0.0237
6	0.0294	0.0093	0.0243	0.0242
8	0.0648	0.0102	0.0410	0.0541

Table 3-15 MSE comparison of NARX with different sensor locations for estimation nodes Set2

No. of sensors	MSE of testing data			
	Sensors locations by K-means	Sensors locations by Ratio method	Test location (Test1)	Test location (Test2)
2	0.1014	0.0726	0.0765	0.1226
4	0.1879	0.0335	0.0559	0.0573
6	0.1040	0.0309	0.0532	0.0638
8	0.0882	0.0465	0.0800	0.0880

3.6 Summary

To deal with the problem of optimal sensor location in a laminated die system, clustering target sensor locations with respect to the thermal response was presented by means of the K-means and temperature ratio method. This is because clustering enables independent information from each identified sensor to be used and to reduce the required number of sensors by selecting a representative location per cluster.

The correlation coefficient of the temperature response and the tolerance rate of the temperature ratio were used as the criteria to define the similarity between nodes for the K-means and the temperature ratio method, respectively. Then, sensitivity analysis of the temperature distribution at each target node was performed to identify a sensor location in each cluster, which is the most sensitive to system input variations during the injection moulding process.

Based on optimal sensor locations obtained from the two clustering methods with sensitivity analysis in the cases of k=2, 4, 6 and 8, these methods were compared from the following perspectives.

In terms of consistency in generating sensor locations, the temperature ratio method is better because when a required number of sensors increases, sensor locations identified for a small number of sensors are included as locations for a larger number of sensors.

In light of sensitivity of sensor locations to input changes, sensor locations obtained by the temperature ratio method have the greater values of the sensitivity coefficient to flow rate and temperature of coolant (system inputs). So, sensor locations by the temperature ratio method provide more informative information in terms of the sensitivity.

By applying PCA, the mutual interaction between selected sensor locations was verified. Since sensor locations by K-means have more varied orthogonal principal components than ones by the temperature ratio method, the K-means is efficient for generating sensor locations with the lowest mutual interaction.

Another advantage of the temperature ratio method is to determine the number of clusters (i.e., sensors) automatically according to the tolerance rate while the K-means needs the predefined number of sensors.

From the viewpoint of accuracy in estimation of temperature response at nodes near the cooling channels where sensor installation is difficult, the temperature ratio method outperforms the K-means.

From an overall perspective, it is concluded that the temperature ratio method with sensitivity analysis is more effective than the K-means with sensitivity analysis to find optimal sensor locations and number.

Thus, the temperature distributions at the optimal sensor locations (in Fig. 3-17) by the temperature ratio method will be used for identifying a thermal model using the FEA. For an experimental validation of the identified thermal model, thermocouples will be installed at these locations.

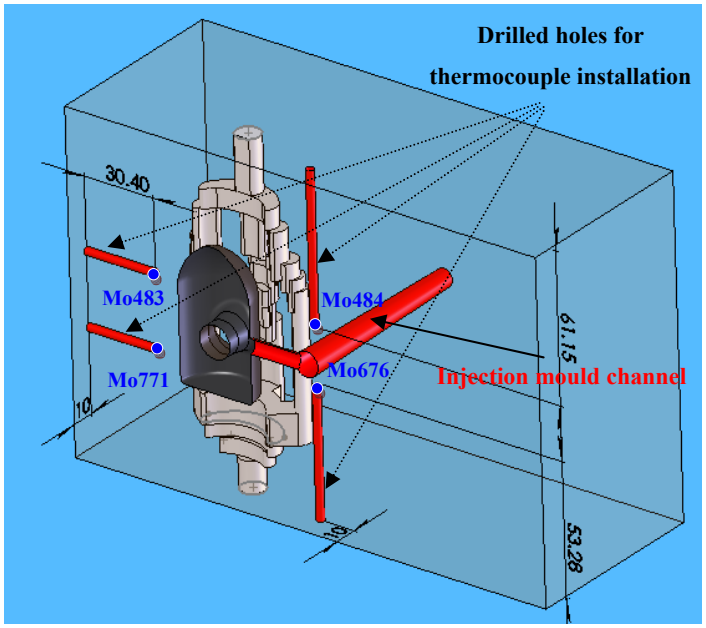


Figure 3-17 Four optimal sensor locations for thermocouple installation

However, the temperature ratio method has a limitation that one cluster among all clusters includes fairly many nodes compared to the others. Many nodes belonging to this cluster are insensitive nodes to input changes due to a relatively farther distance from the hot polymer and cooling channels compared to nodes in the other clusters. So, the linear relationship (i.e., constant ratio) between these nodes at which the small variation of the response occurs, could be more easily obtained. Despite introducing r_{mean} in Eq. (3-9) to prevent this problem, these insensitive nodes are more likely to meet the cut-off tolerance rate.

As another clustering method, the subtractive clustering algorithm [61] can also be considered. In this algorithm, each data point is assumed as a potential cluster center and a measure of the likelihood that each data point would define the cluster center is calculated based on the density of surrounding

data points. After selecting the data point with the highest potential to be the first cluster center, it subtracts all data points in the vicinity of the first cluster center that are within a pre-specified radius to find the next cluster and its center. This process is repeated until all data points are examined [62]. Although the subtractive clustering algorithm requires some parameters such as the radius and the ratios for acceptance and rejection, it has, similar to the proposed clustering method, the advantage of not defining the number of clusters in advance [63].

Therefore, the subtractive clustering algorithm and the temperature ratio method can be compared in terms of the clustering efficiency and the temperature estimation accuracy as a useful future work.

Chapter 4
Cooling performance of laminated die

4.1 Problem overview

The objectives of using conformal cooling with laminated tooling are to improve product quality by uniform cooling, and to reduce the cooling time through fast cooling. In order to study the efficiency in conformal cooling from the viewpoint of uniform and fast cooling, the temperature distribution at sensor locations identified in Chapter 3 and cooling time to reach the solidification temperature are investigated through the FEA analysis. The comparative results with the conventional cooling channels will show how the suggested conformal cooling channels can provide better cooling performance than the conventional design.

4.2 Conventional cooling channels and FE simulation

The cooling effects of conformal cooling channels (Fig. 2-4, pg 10) are compared to those of conventional cooling channels consisting of straight drilled holes into the mould as shown in Fig 4-1.

The FEA conducted for this comparison was based mainly on the second FEA for identifying optimal sensor locations (Section 3.3.1) except for boundary conditions. For example, the flow rate and temperature of the coolant are 7.13 gpm and 15 ^{o}C respectively, which are the operating conditions for the plastic injection moulding process, and the transient simulation time was set to 140 seconds to observe the thermal response over the time and estimate the cooling time required for the process with each type of cooling channel.

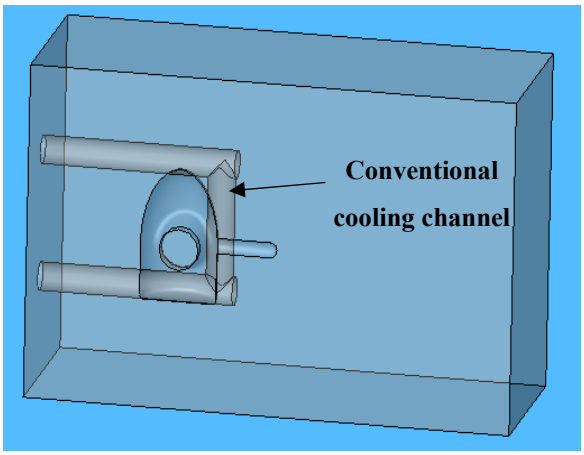


Figure 4-1 Injection mould tool with conventional (straight) cooling channels

4.3 Cooling performance of laminated die with conformal cooling channels

4.3.1 Uniformity of temperature distribution

To evaluate the cooling effect of each type of channel in terms of uniform cooling, the temperature distributions were compared. Figure 4-2 shows the temperature distributions inside moulds around conformal and conventional channels. As seen in this figure, lower temperatures in the mould with the conformal cooling channels are evenly distributed around the cavity and thus the part can be more uniformly cooled down. However, the conventional channels provide a uniform temperature distribution around not the cavity but around the cooling channel.

Figure 4-3 illustrates the difference in the temperature distribution on the symmetric half cut of moulds with conformal and conventional channels. From this figure, it is noted that an optimized cooling through conformal channels is attained by emitting more uniform heat from the part (cavity) to the cooling channel. The uniform cooling can be also judged by means of temperature deviations at some nodes near the cavity. The same five nodes (Fig. 4-4) located inside moulds were chosen to monitor the temperature deviation between these nodes in each cooling channel. Four nodes (Mo483, Mo484, Mo676, Mo771) were determined by the temperature ratio method and sensitivity analysis (Fig. 3-17) and one additional node (M) was located in the middle of the mould encompassing the cavity.

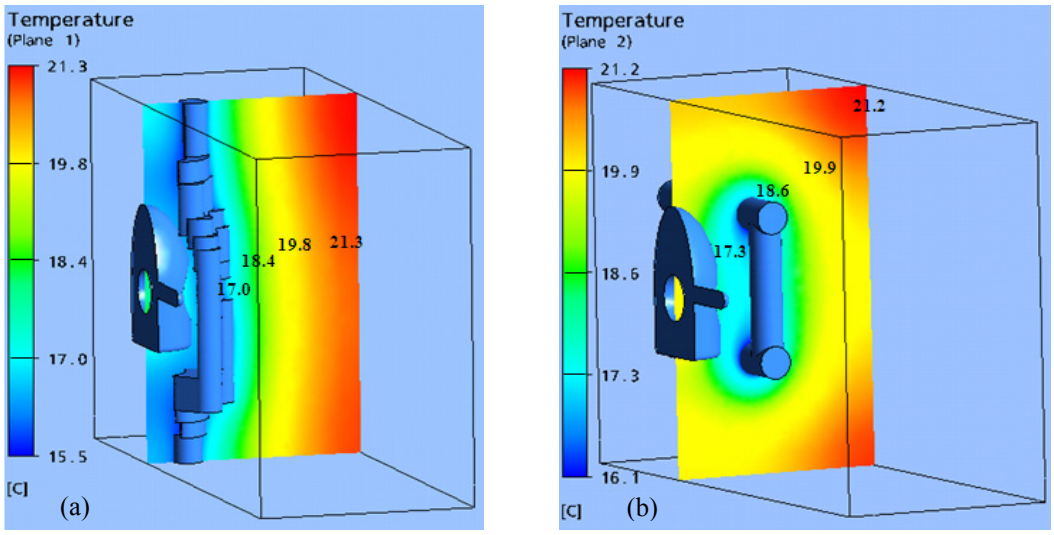


Figure 4-2 Temperature distributions inside moulds around conformal (a) and conventional channels (b)

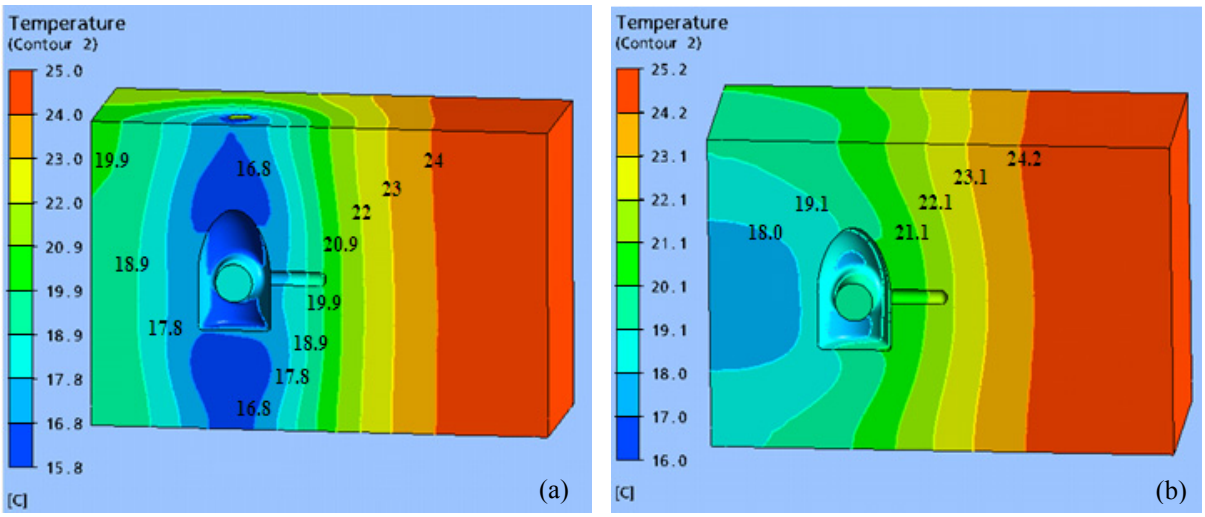


Figure 4-3 Temperature distributions on half cut of moulds with conformal (a) and conventional channels (b)

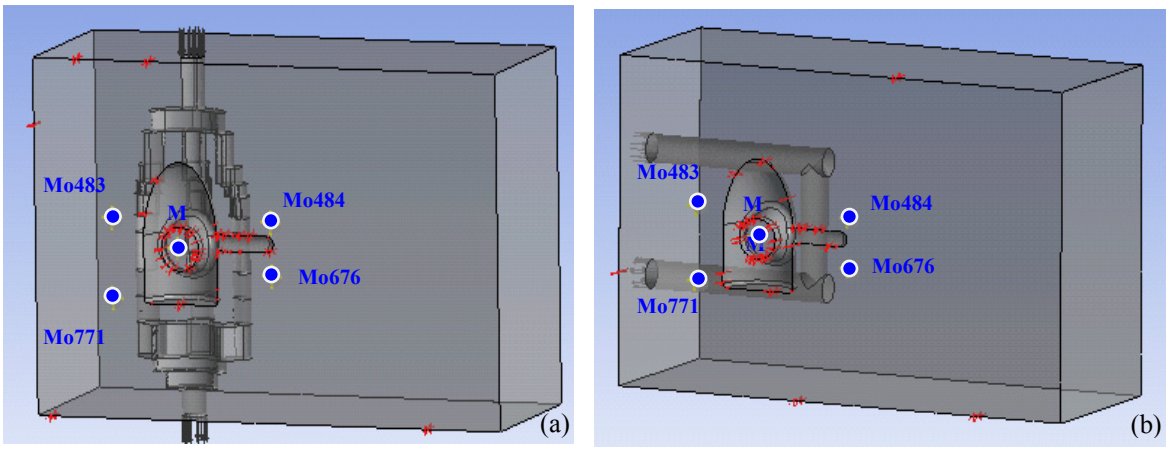


Figure 4-4 Five nodes near cavity to monitor temperature deviation in conformal (a) and conventional channels (b)

Figure 4-5 and 4-6 present the temperature profiles at each node with the conformal channels and the conventional channels, respectively. As depicted in the figures, the temperature deviations between nodes in the conformal channel mould are less than those in the conventional channel mould at all times. The average temperature at each node in Table 4-1 ensures less deviation of temperature between nodes with the conformal channels than the conventional channels. For example, the difference between the maximum (average at Mo484: 21.06 (°*C*)) and the minimum (average at Mo483: 19.97 (°*C*)) in the case of the conformal channels is 1.09 °*C*. This is smaller than the 3.42 °*C* with conventional channels, which is the difference between the maximum (M: 23.53 (°*C*)) and the minimum (Mo771: 20.11 (°*C*)).

Finally, the temperature deviation is investigated by using the following equation:

$$Td = \sqrt{\frac{\sum_{t=1}^{tn} (T_{\max}(t) - T_{\min}(t))^2}{tn}} \qquad (4\text{-}1)$$

where $T_{\max}(t)$ is the maximum temperature among the 5 nodes at time t; $T_{\min}(t)$ is the minimum temperature among the 5 nodes at time t; tn is the interval time (140 seconds). The much smaller

value of *Td*, shown in Table 4-2, for the conformal cooling channels, indicates the significantly lower temperature variation over the conventional channel design.

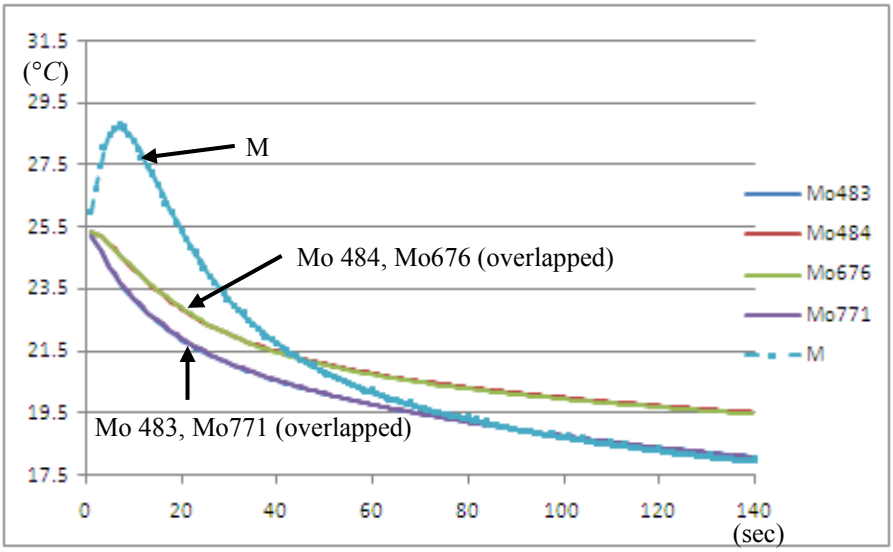


Figure 4-5 Temperature profiles at five nodes with conformal cooling channel mould

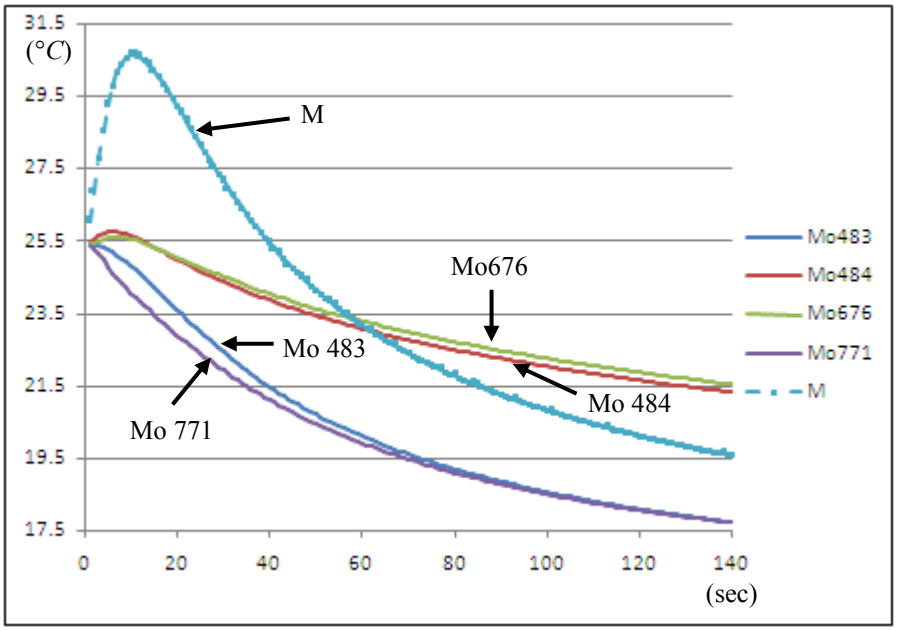


Figure 4-6 Temperature profiles at five nodes with conventional cooling channel mould

Table 4-1 Average temperature at five nodes for interval time (140 seconds)

	Node	Conformal channels	Conventional channels
Average temperature at each node (°*C*)	Mo483	19.97	20.35
	Mo484	21.06	23.08
	Mo676	21.04	23.24
	Mo771	19.98	20.11
	M	20.95	23.53

Table 4-2 Temperature deviation at five nodes for interval time (140 seconds)

Node	Conformal channels	Conventional channels
Td	2.08	4.29

From the results above, it is clear that the conformal cooling channels enable the temperature distribution of mould to be both lower and more uniform and therefore provide better cooling for the process.

4.3.2 Cooling time

Cooling time is the time to cool the part from melt temperature down to solidification (or de-moulding) temperature [64], [65]. Since the polymer (Santoprene 8211-45) used in the study has a solidification temperature of 85 °*C*, the time for the polymer to reach this temperature from its injection temperature is considered to be the cooling time. Figure 4-7 depicts the temperature variations of the parts with conformal and conventional channels over 140 seconds after injection.

The part with the conformal channels reaches the de-moulding temperature (85 °*C*) at 47 second whereas the one with the conventional channels needs 66 seconds to reach the same temperature. This corresponds closely to the actual cooling time (68 seconds) used for de-moulding in the current injection moulding process. The cooling time can then be reduced by 19 seconds (i.e., by almost 29%) by using the conformal cooling channels.

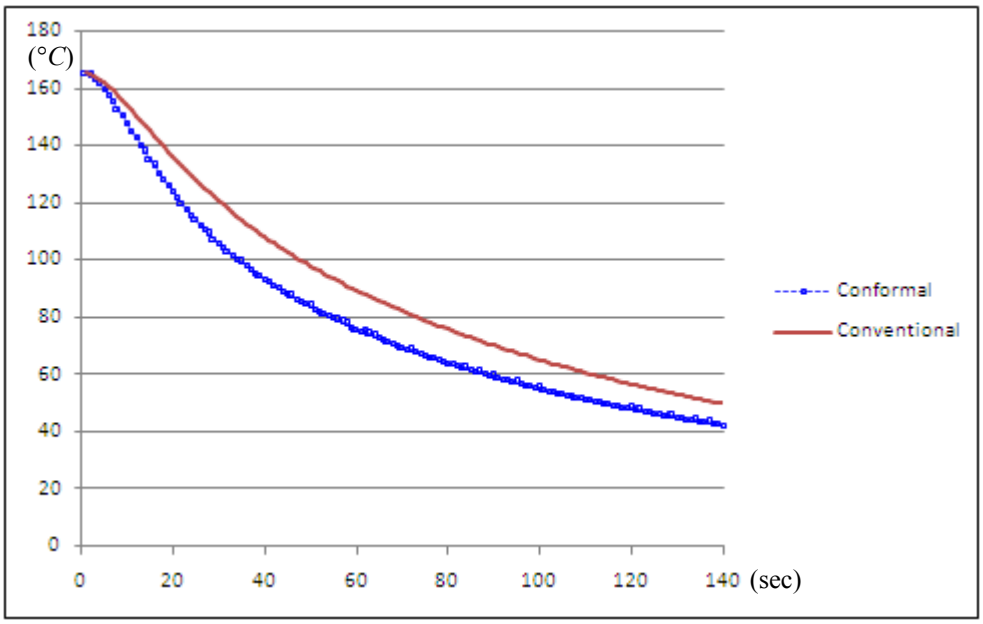


Figure 4-7 Temperature profiles of parts with conformal and conventional cooling channels

4.4 Summary

A comparative study using the finite element thermal analysis of a mould with conventional cooling channels versus one with conformal cooling channels has verified the efficiency of the conformal cooling channels on the cycle time for the plastic injection moulding process. The conformal cooling channels contribute to improving cooling on the mould by allowing a more uniform temperature distribution around the cavity. In addition, it reduces the cooling time and thus increases the production rate. From these perspectives, it is concluded that the introduction of conformal cooling channels using laminated tooling enables a more effective cooling for injection moulding process to be achieved.

Chapter 5
System identification of die thermal dynamics

5.1 Problem overview

As the first step in temperature control of laminated die systems, it is necessary to identify the dynamic behavior of the system to provide the groundwork for the application of control techniques.

Although accurate modeling of the thermal dynamics of dies is prerequisite for successful thermal control, it is a difficult task in practice due to the die's characteristics. For example, a die is a complex continuous system with cooling and heating channels causing the modeling and control to be quite complicated. Unmodeled thermal dynamics of dies (such as convection and radiation) also provide further analytical model challenges.

To deal with the limitations of simplified models aforementioned in Section 2.1 and inherent challenges of modeling in a die system, this study considers a neural network (NN) approach. By applying NN techniques, the thermal dynamics with uncertainties in a laminated die is modeled.

In this research, a combination of off-line and on-line learning techniques is used for the system identification. Figure 5-1 presents the entire procedure of the system identification using NNs through off-line and on-line training. Through the off-line learning using the data set gathered from the FEA, the NN is first trained so that a sensible output can be generated over the complete operating range [66]. The advantage of off-line training using the FEA is that enough data sets to adequately represent the system dynamics can be obtained from the FE simulations with high-speed computation, compared to experiments that cover a relatively limited range of data sets due to the limitation of possible operating conditions and geometric constraints (where data gathering is challenging).

Based on this initialized structure (i.e., initial weights and biases) of the NN from the off-line learning, the NN is re-trained for each new pattern (i.e., experimental data) from the real-world system. Since variations in processing conditions in the real-time process become more complex than those in simulations due to the injection material properties, cooling variables and environmental factors (such as variable convective heat transfer) from the experimental site, on-line training is additionally required to refine the model from the off-line training through the experimental data.

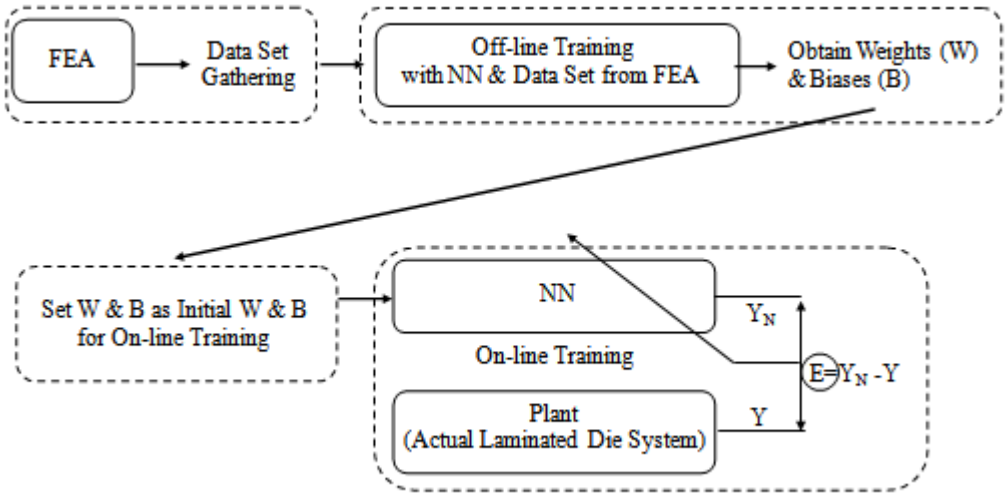


Figure 5-1 Procedure of system identification using NNs

5.2 Modeling of thermal dynamics

5.2.1 Various cycle times

The cycle-time for a plastic injection moulding process consists of the phases of injection, packing, holding, cooling and ejection. Since the cooling phase consumes a large portion of the cycle-time, cooling the polymer to solidification temperature [64], the cooling time plays a key role in the cycle-time. Previous studies dealing with system identification and thermal control of the injection moulding process have used a predetermined cooling time (thus cycle-time). However, system identification using a fixed cycle-time cannot cope with the wide range of thermal dynamic variations due to varying cycle-times. A control strategy based on this limited identification cannot properly control the die temperature. In this study, various cycle-times of the moulding process are considered in the thermal dynamics modeling.

5.2.2 Off-line training using NARX model and FEA

To capture a dynamic model to describe the temperature distribution in the die, the NARX model described in Eq. (3-12) was considered due to its merits (refer to Section 3.5.2). As an input of the NARX model, the coolant flow rate was chosen from previous studies regarding thermal analysis of

dies to report its significant effect on die temperature [45]. The temperature of the coolant is also a major factor in controlling the die temperature [10] but real plastic moulding processes use a coolant with a constant temperature (typical 15.5 ^{o}C) and thus, the coolant temperature was not utilized as an input in the NARX model (though the coolant temperature has a much greater effect than its flow rate (Fig. 3-5, 3-6)). Other parameters that could be potential sources of uncertainty in the model are considered as noise since their effects are relatively minor in comparison to those of the flow rate of the coolant, and these effects would be mitigated by the thermal management control system.

When applying the NARX model to our system, the outputs of the NARX model are the temperatures at the 4 nodes in Fig. 3-17 (pg 59). However, the rapid change of the temperature during the cycle makes it difficult to use this temperature profile as a set-point and thus an alternative variable is required as an output. This was chosen as the cycle average temperature [13], and is defined in Eq. (5-1).

$$T_a = \frac{\int_0^{Ct} T dt}{Ct} = \frac{\sum_0^M T_n}{M} \tag{5-1}$$

where t is time, T_a is the cycle average temperature, T is the instantaneous temperature at a specific location, Ct is the cycle-time, M is the total number of samples during one cycle, and T_n is the temperature at the n^{th} sample time. The use of T_a has the advantage of its reduced sensitivity to process noise.

By considering the structure requiring the model orders (i.e., past states), the past outputs of temperatures at each node are included as inputs to the NARX model. As mentioned before, our modeling is extended to deal with various cycle-times rather than a predetermined cycle-time that previous studies have considered. Therefore, the cycle-time for the moulding process is additionally included as an input variable to the model. The NARX model covering all above inputs (flow rate, cycle-time, past values of outputs at 4 nodes) and outputs (T_a at 4 nodes) is a multi-input multi-output (MIMO) system.

Data for training the NARX model off-line was obtained from the FEA, which is similar to the second FEA used for identifying optimum sensor locations, but different in terms of input (flow rate)-output data generation (this point will be described in Fig. 5-2). To improve the reliability of the FEA results, the mesh convergence study was carried out. Then, four nodes among vicinity nodes of the FEA model corresponding to four sensor locations in Fig. 3-17 (pg 59), were selected for output data generation. The FEA outputs at selected nodes were the closest to the actual values at four sensor locations obtained from an initial experiment for FEA model validation.

The following table shows input conditions for flow rate and cycle-time to generate the steady state outputs used for off-line training of the NARX modeling.

Table 5-1 Input conditions of FEA for NARX modeling

	Input variables	Range
Training	Flow rate (gpm)	0, 1, 3, 4, 5, 6, 8
	Cycle-time (sec)	61, 71, 81, 91

A portion of the training data sets is shown in Fig. 5-2, which is generated for node Mo484 using all flow rates in Table 5-1 and one cycle-time (91 seconds) from the four values of cycle-time.

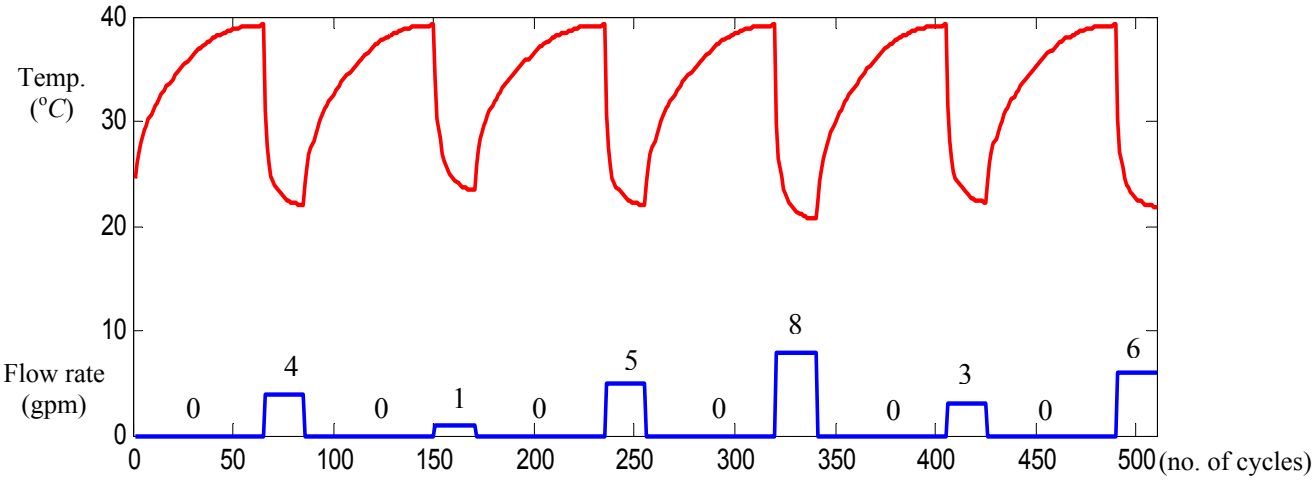


Figure 5-2 Training data using all flow rate ranges (0, 1, 3, 4, 5, 6, 8 gpm) and one cycle-time (91 sec) at node Mo484

From Fig. 5-2, it can be seen that for data gathering, each flow rate (input) for the given cycle-time stays constant until the steady state of the temperature at the node (output) is reached over many cycles [10]. Additional data sets for modeling the temperature at the same node (Mo 484) were generated in the same way by using the same flow rate ranges (0, 1, 3, 4, 5, 6, 8 gpm) with each different cycle-time (i.e., 61, 71, 81 seconds). This method was also applied for data generation at the remaining three nodes (Mo483, Mo676 and Mo771).

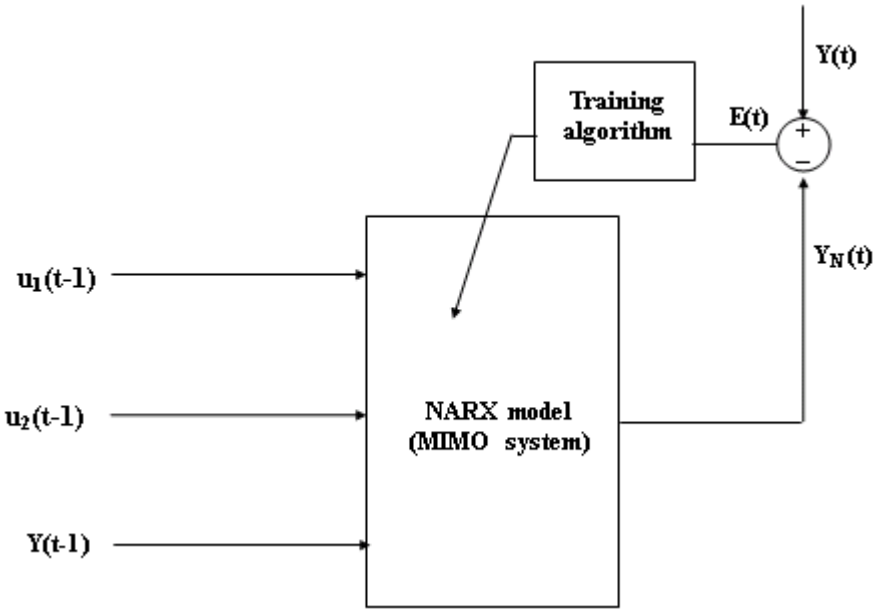
The next step is to determine the model order in the NARX model of Eq. (3-12) by applying the Lipschitz criterion [59] and using the training data sets. The obtained model orders are shown in the schematic of the NARX model (see Fig. 5-3).

For better results and a reduction of the calculation time [67], data were normalized by Eq. (5-2) and Eq. (5-3) before being used for the training and validation process [10].

$$y_{Nor,i}(t) = \frac{y_i^{\max} - y_i(t)}{y_i^{\max} - y_i^{\min}} \tag{5-2}$$

$$u_{Nor,m}(t) = \frac{u_m(t)}{u_m^{\max}} \tag{5-3}$$

where $y_{Nor,i}$, $y_i^{\max}$, $y_i^{\min}$ are the normalized maximum, minimum values of the output y (temperature) at the i^{th} node, respectively. $u_{Nor,m}$ and $u_m^{\max}$ are the normalized and maximum values of the m^{th} input, u_m (cycle-time or flow rate), respectively.



- Temperature at 4 nodes which are Mo483, Mo484, Mo676, Mo771:
  $Y(t)=[y_{Mo483}(t); y_{Mo484}(t); y_{Mo676}(t); y_{Mo771}(t)]$
- Neural network output: $Y_N(t)$
- System inputs: $u_1(t-1)$: cycle-time, $u_2(t-1)$: water flow rate, Y(t-1): past values of Y(t) with one cycle-time delay.

Figure 5-3 Schematic of NARX model

The Levenberg-Marquardt (LM) algorithm was adopted as a training algorithm, and the number of hidden layer nodes (6 nodes) was determined by trial and error, which generated the smallest mean square error (MSE). After training with the training data (2040 each covering all input conditions in Table 5-1) with 200 epochs, the NARX model's performance (i.e., accuracy) is presented in Fig. 5-4 and 5-5.

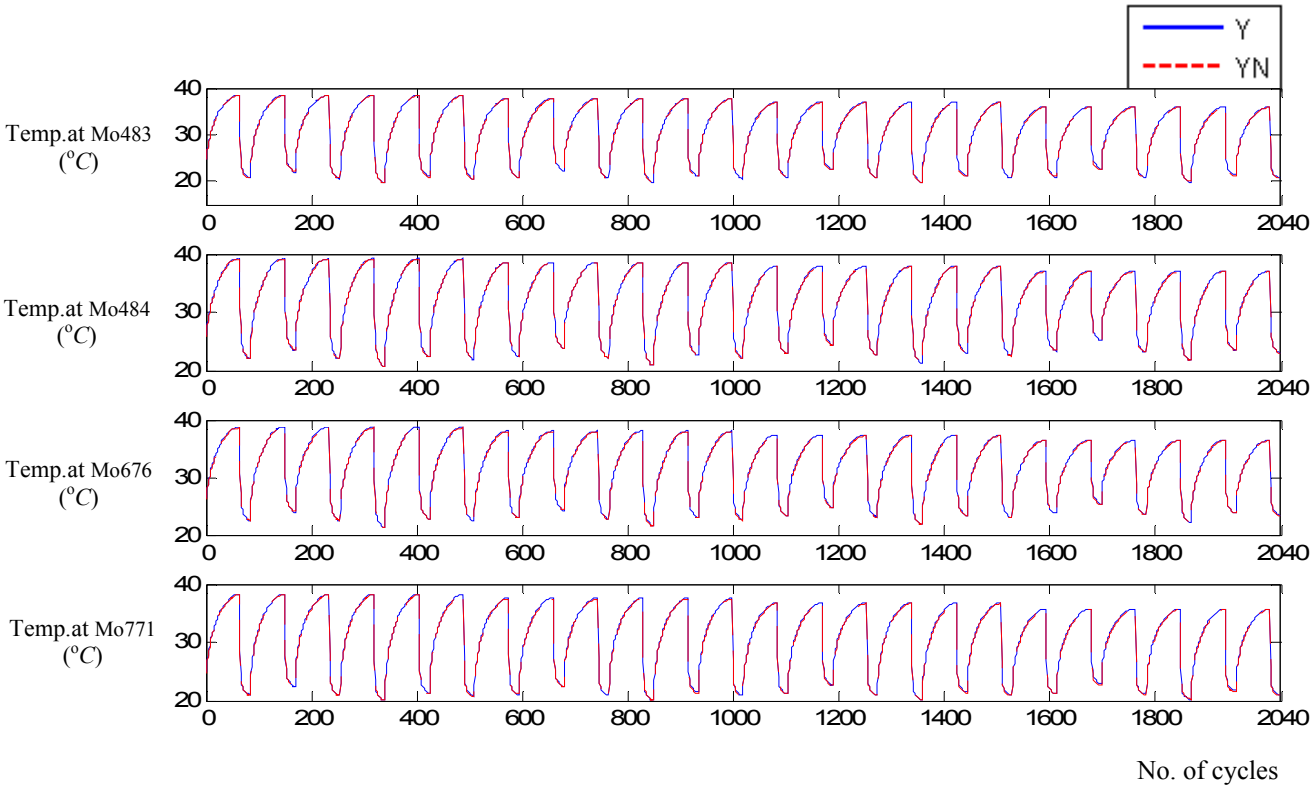


Figure 5-4 Off-line training results (comparison between model (Y_N) and actual outputs (Y))

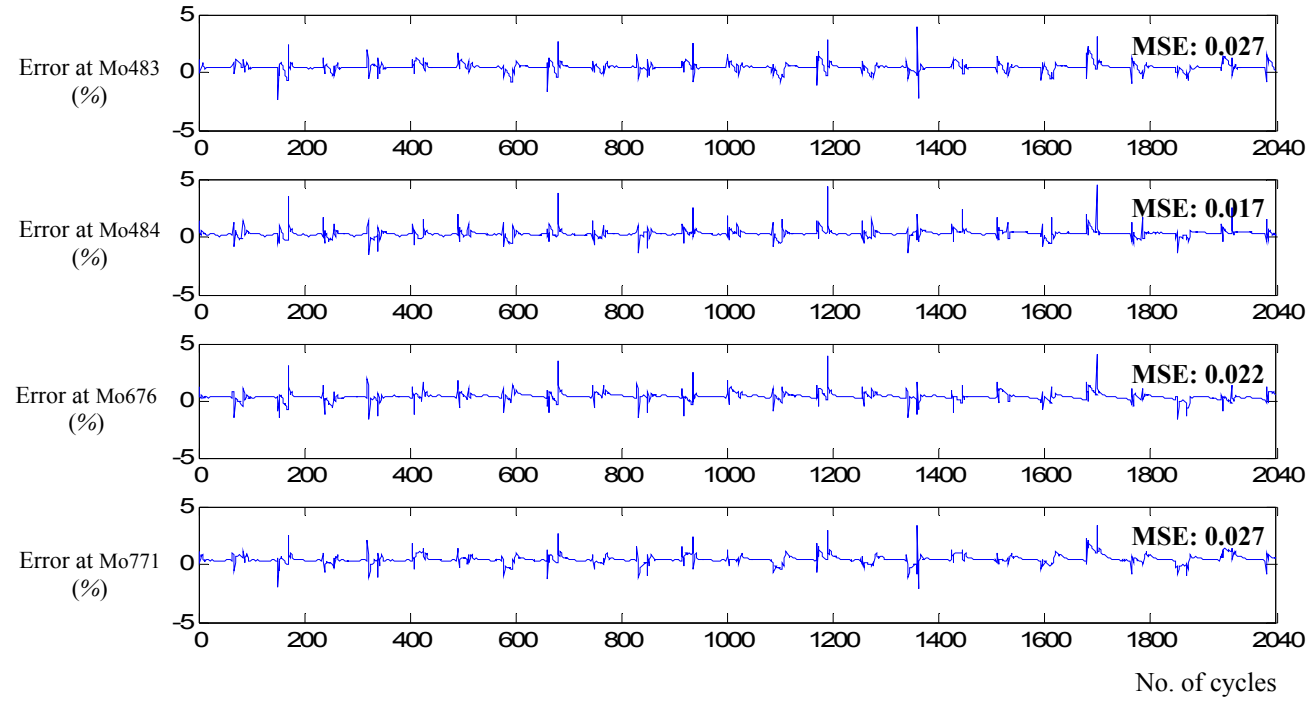


Figure 5-5 Off-line training results (error between Y_N and Y)

The MSE values in Fig. 5.5 exhibit negligible errors between the estimated outputs from the NARX neural network (Y_N) and the actual outputs (Y) from the FEA at all nodes.

5.3 Experimental test

5.3.1 On-line training algorithm

The selection of a proper learning algorithm is also an important issue for on-line training to get optimum weights (and biases) of the NN to represent the system dynamics. Some conjugate gradient based learning algorithms such as the Levenberg–Marquardt method used for off-line training in this study, have a critical drawback of requiring large computational demands and therefore, their use in on-line identification problems is not effective [68]. Sha and Bajic [68], [69] suggested an on-line learning algorithm based on optimized instantaneous learning rates of gradient descent rule. Because this algorithm is viable for on-line identification in terms of learning speed and training error of the on-line learning process, it is adopted in the study.

By referring to Eq. (2-2) (pg 8) and setting $b_{h0}^{w}(t) = w_{h0}(t)$ and $b_{m0}^{v}(t) = v_{m0}(t)$, the NN model can be described as:

$$\hat{Y}_{NN}(t+1) = \begin{bmatrix} \hat{y}_{NN\ 1}(t+1) \\ \hat{y}_{NN\ 2}(t+1) \\ \vdots \\ \hat{y}_{NN\ m}(t+1) \end{bmatrix} = \begin{bmatrix} f_2\left(\sum_{h=1}^{l} v_{1h}(t) f_1\left(\sum_{j=1}^{n} w_{hj}(t)(t) x_j(t) + b_{h0}^{w}(t)\right) + b_{10}^{v}(t)\right) \\ f_2\left(\sum_{h=1}^{l} v_{2h}(t) f_1\left(\sum_{j=1}^{n} w_{hj}(t) x_j(t) + b_{h0}^{w}(t)\right) + b_{20}^{v}(t)\right) \\ \vdots \\ f_2\left(\sum_{h=1}^{l} v_{mh}(t) f_1\left(\sum_{j=1}^{n} w_{hj}(t) x_j(t) + b_{h0}^{w}(t)\right) + b_{m0}^{v}(t)\right) \end{bmatrix}$$

$$
= \begin{bmatrix}
f_2\left(\sum_{h=1}^{l} v_{1h}(t) f_1\left(\sum_{j=1}^{n} w_{hj}(t) x_j(t) + w_{h0}(t)\right) + v_{10}(t)\right) \\
f_2\left(\sum_{h=1}^{l} v_{2h}(t) f_1\left(\sum_{j=1}^{n} w_{hj}(t) x_j(t) + w_{h0}(t)\right) + v_{20}(t)\right) \\
\vdots \\
f_2\left(\sum_{h=1}^{l} v_{mh}(t) f_1\left(\sum_{j=1}^{n} w_{hj}(t) x_j(t) + w_{h0}(t)\right) + v_{m0}(t)\right)
\end{bmatrix}
$$

$$
= \begin{bmatrix}
f_2\left(\sum_{h=0}^{l} v_{1h}(t) f_1\left(\sum_{j=0}^{n} w_{hj}(t) x_j(t)\right)\right) \\
f_2\left(\sum_{h=0}^{l} v_{2h}(t) f_1\left(\sum_{j=0}^{n} w_{hj}(t) x_j(t)\right)\right) \\
\vdots \\
f_2\left(\sum_{h=0}^{l} v_{mh}(t) f_1\left(\sum_{j=0}^{n} w_{hj}(t) x_j(t)\right)\right)
\end{bmatrix}
= \begin{bmatrix}
f_2\left[V_1^a(t) F_1(W^a(t) X^a(t))\right] \\
f_2\left[V_2^a(t) F_1(W^a(t) X^a(t))\right] \\
\vdots \\
f_2\left[V_m^a(t) F_1(W^a(t) X^a(t))\right]
\end{bmatrix}
$$

$$
= F_2\left[V^a(t) F_1(W^a(t) X^a(t))\right] \tag{5-4}
$$

where $x_j(t)$, $\hat{y}_{NN\,m}(t)$, $w_{hj}(t)$ and $v_{mh}(t)$ are the NN inputs, the NN outputs, the weights in the hidden layer and the weights in the output layer, respectively. $f_2(x) = x$, $f_1(x) = 2/(1+\exp(-x)) - 1$, and n, l and m are the number of inputs, neurons in the hidden layer and neurons in the output layer, respectively.

$$
X^a = \left[x_0(=1),\ x_1, \cdots, x_n\right]^T, \quad W^a = \begin{bmatrix}
w_{10} & w_{11} & \cdots & w_{1n} \\
w_{20} & w_{21} & \cdots & w_{2n} \\
\vdots & \vdots & \vdots & \vdots \\
w_{l0} & w_{l1} & \cdots & w_{ln}
\end{bmatrix}, \quad V^a = \begin{bmatrix}
v_{10} & v_{11} & \cdots & v_{1l} \\
v_{20} & v_{21} & \cdots & v_{2l} \\
\vdots & \vdots & \vdots & \vdots \\
v_{m0} & v_{m1} & \cdots & v_{ml}
\end{bmatrix},
$$

$$
F_1(W^a(t) X^a(t)) = \left[f_1\left((net_h(t))_0\right), f_1\left((net_h(t))_1\right), \cdots, f_1\left((net_h(t))_l\right)\right]^T, \text{ and } \quad net_h(t) = \sum_{j=0}^{n} w_{hj}(t) x_j(t).
$$

To derive the optimal learning rate, consider the following error increment equation:

$$\begin{aligned}\Delta e(t+1) = e(t+1) - e(t) &= \left(y(t+1) - \hat{y}_{NN}(t+1)\right) - \left(y(t) - \hat{y}_{NN}(t)\right) \\ &= \Delta y(t+1) - \Delta \hat{y}_{NN}(t+1) \cong -\Delta \hat{y}_{NN}(t+1)\end{aligned} \tag{5-5}$$

where $\Delta y(t+1) = y(t+1) - y(t)$, $\Delta \hat{y}_{NN}(t+1) = \hat{y}_{NN}(t+1) - \hat{y}_{NN}(t)$, $y(t)$ is the actual output, and $\hat{y}_{NN}(t)$ is the NN output. Assume $\left|\Delta y(t+1)\right| << \left|\Delta \hat{y}_{NN}(t+1)\right|$, namely the absolute value of the change of the actual output is much smaller than the absolute value of the change of the NN output during the training process.

This assumption is realistic for many processes due to energy limitations in practical systems, whereas such limitations are not imposed on the output of the NN. If this condition is not satisfied, the NN identification system will not be able to adapt sufficiently fast to the change in the actual output, and thus identification of the process model will not be possible.

Next, the error increment is approximated as:

$$e(t+1) \approx e(t) - \Delta \hat{y}_{NN}(t+1) = e(t) - \eta \xi(t) e(t) \tag{5-6}$$

where $\xi(t) = F_1^T F_1 I_{m\times m} + \bar{V}^a \bar{F}_1' \bar{F}_1' \left(\bar{V}^a\right)^T (X(t)^T X(t))$, $I_{m\times m}$ is a $m \times m$ identify matrix, $\bar{F}_1'\left(W^a(t) X^a(t)\right) = diag\left[f_1'\left((net_h(t))_1\right), f_1'\left((net_h(t))_2\right), \cdots, f_1'\left((net_h(t))_l\right)\right]$, F_1 is defined in Eq. (5-4), $\bar{V}^a = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1l} \\ v_{21} & v_{22} & \cdots & v_{2l} \\ \vdots & \vdots & \vdots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{ml} \end{bmatrix}$, and m is the number of neurons in the output layer.

Using Eq. (5-6) and the definition of the error cost function: $E(t) = \frac{1}{2}[e(t)^T e(t)]$, the equation for $E(t+1)$ becomes:

$$
\begin{aligned}
E(t+1) &= 0.5e(t+1)^T e(t+1) \\
&= 0.5[e(t)-\eta\xi(t)e(t)]^T[e(t)-\eta\xi(t)e(t)]
\end{aligned}
\tag{5-7}
$$

In order to find an optimal value of the learning rate at t, $\eta^{op}(t)$, which minimizes $E(t+1)$, the following first and second order conditions in Eq. (5-7) are required:

$$
\begin{aligned}
&\left.\frac{dE(t+1)}{d\eta}\right|_{\eta=\eta^{op}(t)} = -0.5[\xi(t)e(t)]^T[e(t)-\eta^{op}(t)\xi(t)e(t)] - 0.5[e(t)-\eta^{op}(t)\xi(t)e(t)]^T[\xi(t)e(t)] = 0 \\
&\left.\frac{d^2E(t+1)}{d\eta^2}\right|_{\eta=\eta^{op}(t)} = [\xi(t)e(t)]^T[\xi(t)e(t)] > 0
\end{aligned}
\tag{5-8}
$$

The optimized learning rate to meet above the conditions is:

$$
\eta^{op}(t) = \frac{[\xi(t)e(t)]^T e(t)}{[\xi(t)e(t)]^T[\xi(t)e(t)]} \tag{5-9}
$$

Using (5-9), weights can be updated in accordance with the following increments:

$$
\begin{aligned}
\Delta W^a(t) &= -\eta^{op}(t)(-\overline{F}_1'(t)\left(\overline{V}^a\right)^T F_2'(t)e(t)X(t)^T) = \eta^{op}(t)\overline{F}_1'(t)\left(\overline{V}^a\right)^T e(t)X(t)^T \\
\Delta V^a(t) &= -\eta^{op}(t)(-F_1(t)e(t)^T F_2'(t))^T = \eta^{op}(t)e(t)F_1(t)^T
\end{aligned}
\tag{5-10}
$$

where $F_2'(t)=1$ due to $F_2(x)=x$.

5.3.2 Training using experimental data and on-line training algorithm

Experiments were carried out for improving the model off-line trained in Section 5.2.2, and validation of the model was finalized through experimental data set. For this, an experimental setup was designed as follows:

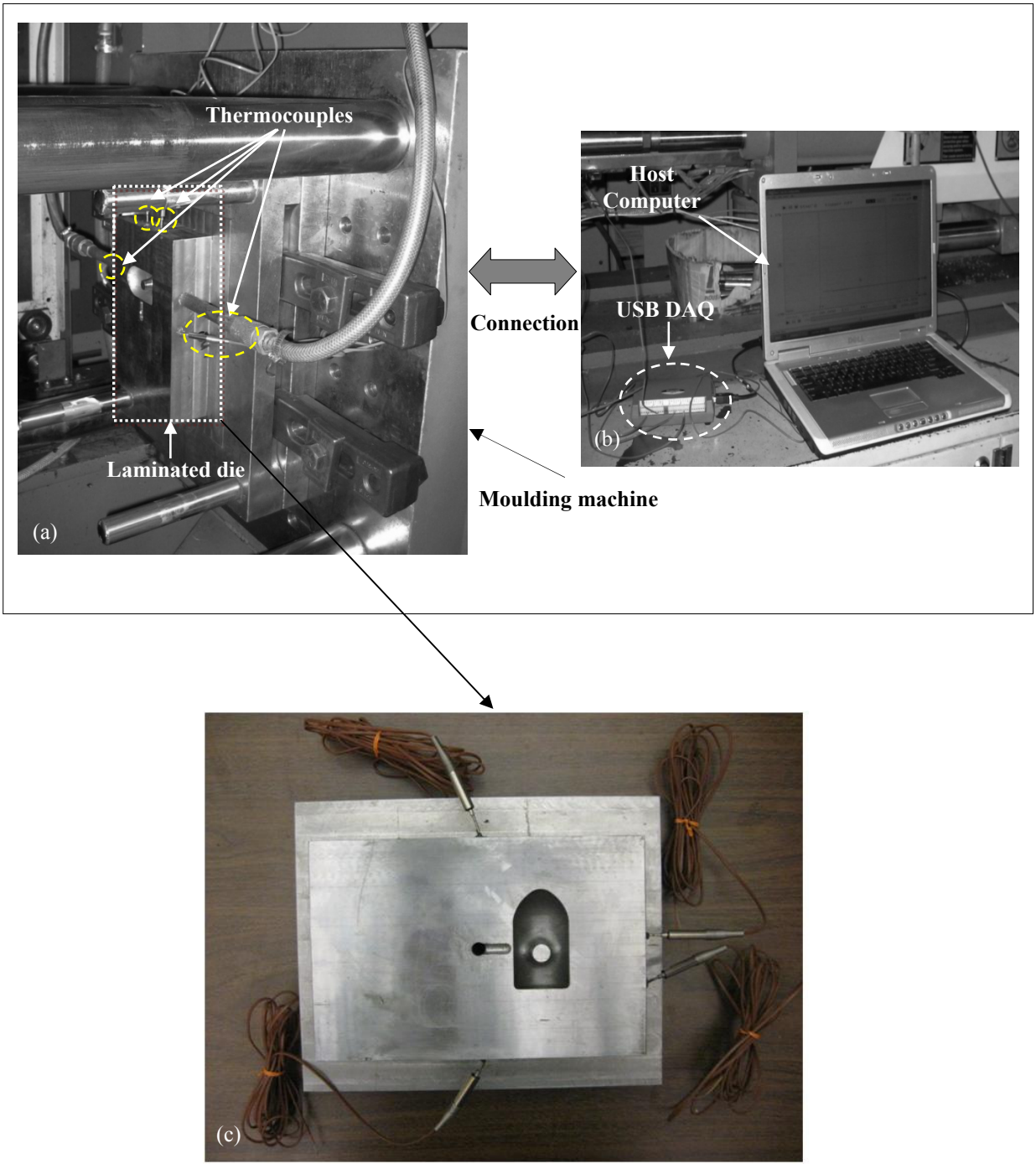


Figure 5-6 Layout of experimental setup ((a): installation of laminated die on moulding machine, (b): host pc with USB DAQ and (c): laminated die with thermocouples)

As represented in Fig. 5-6-(a), the laminated die was installed on the moulding machine at WebPlas Inc. (http://www.webplas.ca). Four thermocouples (type K) inserted in the die were used to measure the temperature at 4 nodes (refer to Fig. 3-17 (pg 60)).

The thermocouples were connected to a USB data acquisition module (DAQ) (see Fig. 5-6-(b)). The temperature distribution acquired by DAQ was sent to the host pc where data was monitored and saved.

The die was cooled by coolant circulating through a chiller with an operating pressure of 85 psi (6894 Pa). For experimental validation, the cycle-time for all phases (i.e., injection, packing, holding, cooling and ejection) was varied from 91 to 61 seconds. The flow rate of the coolant was controlled by the supply valves. Through the on-line learning algorithm in Eq. (5-10) and data set (1019 each) generated from the experiments with the input conditions in Table 5-2, additional training was conducted. The weights and biases obtained from the off-line training with the LM algorithm (Section 5.2.2) were used as initial weights and biases for training using the on-line learning algorithm. This setting of the weights and biases is possible because the NN for on-line algorithm based training has the same structure as that of the NN used for off-line training. By doing so, it is possible to reduce the time taken to find the optimized NN structure by means of the real-time (on-line) training. Results of on-line algorithm based training are given in Fig. 5-7 and 5-8.

Table 5-2 Input conditions of training using on-line learning algorithm

	Input variables	Range
Training using on-line learning algorithm	Flow rate (gpm)	0, 0.79, 6.54, 7.04
	Cycle-time (sec)	61, 71, 81, 91

After training, the NARX model was tested with experimental validation data that was obtained by other input conditions (flow rate: 0—7.13—0.5—2 gpm (which are not used in Table 5-2 for on-line training algorithm); cycle-time: 91—61 seconds over 120 cycles in Fig. 5-9).

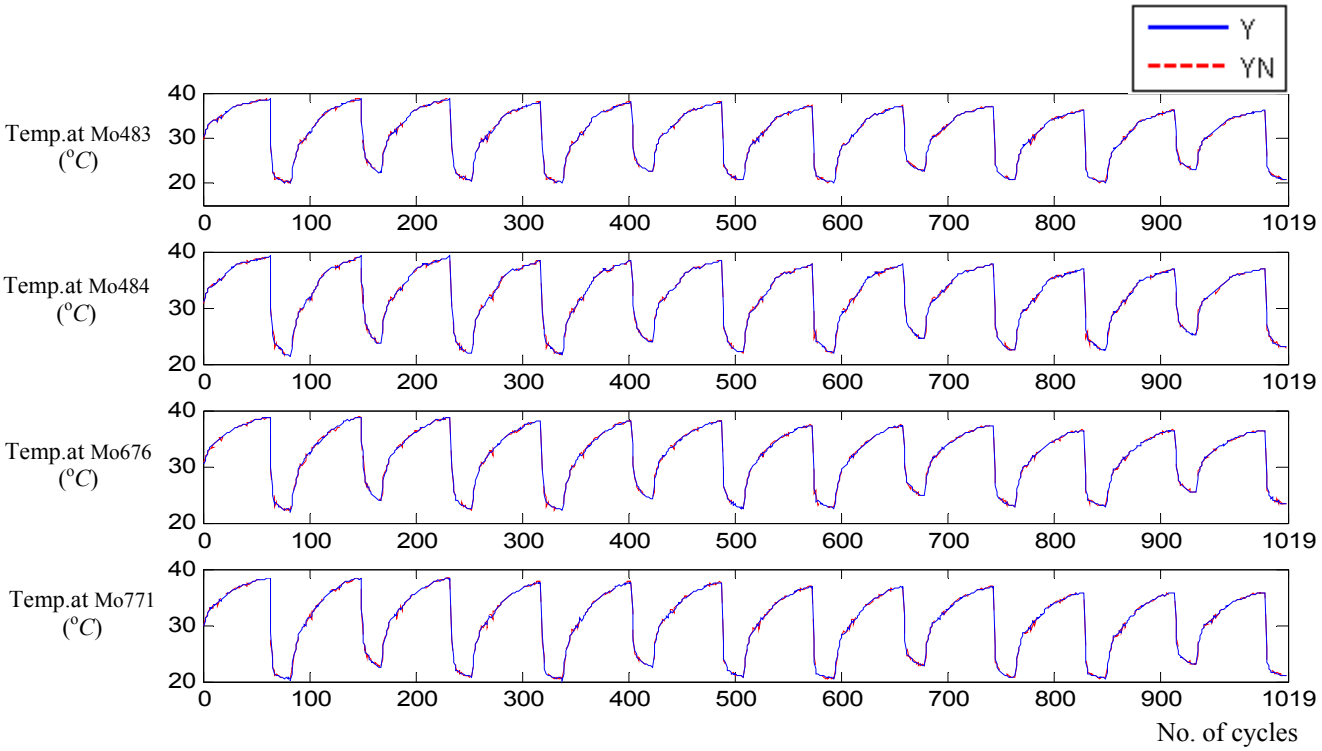


Figure 5-7 Results of training using on-line learning algorithm (comparison between model (Y_N) and actual outputs (Y))

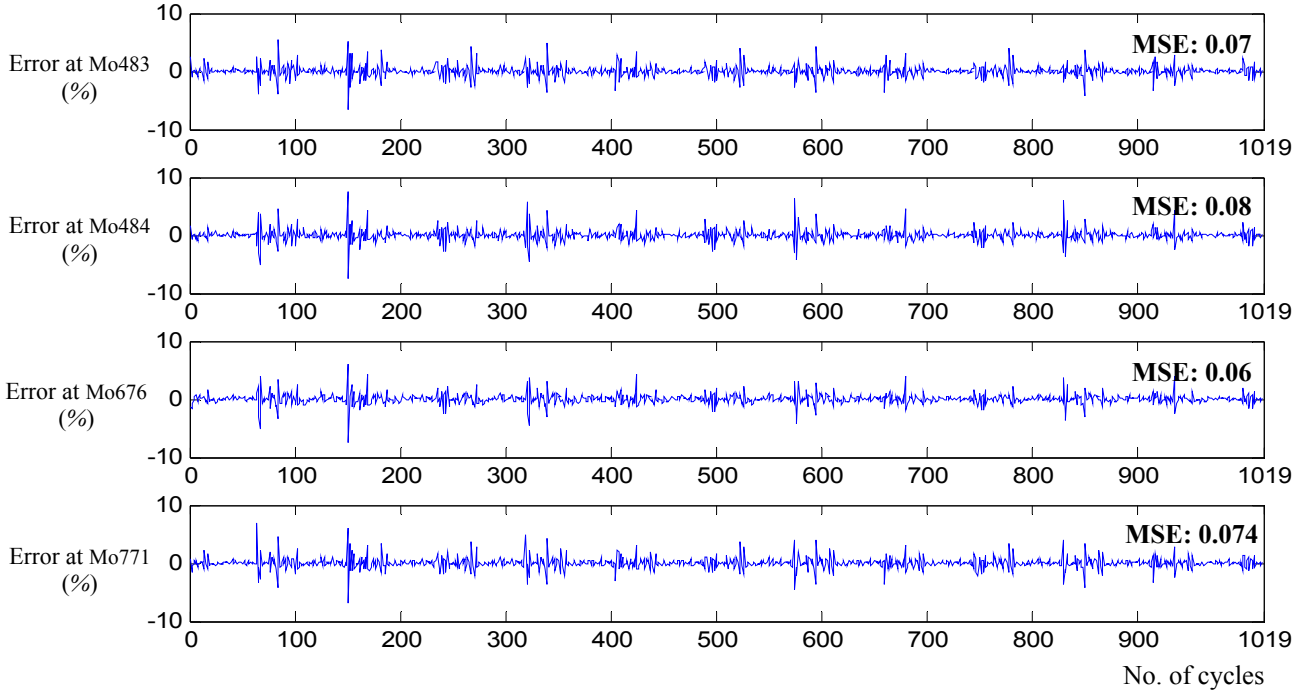


Figure 5-8 Results of training using on-line learning algorithm (error between Y_N and Y)

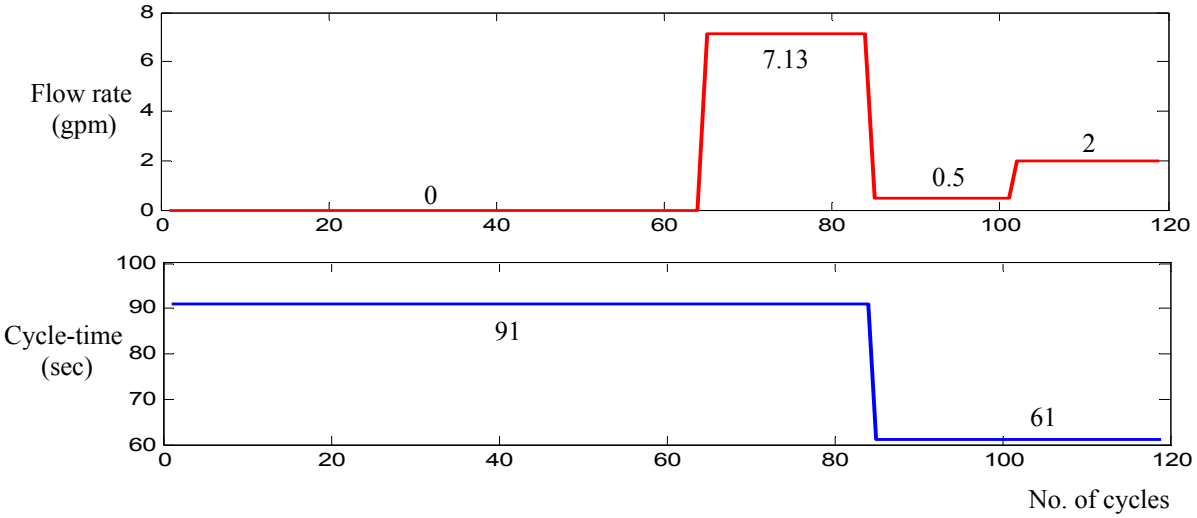


Figure 5-9 Input conditions for experimental validation

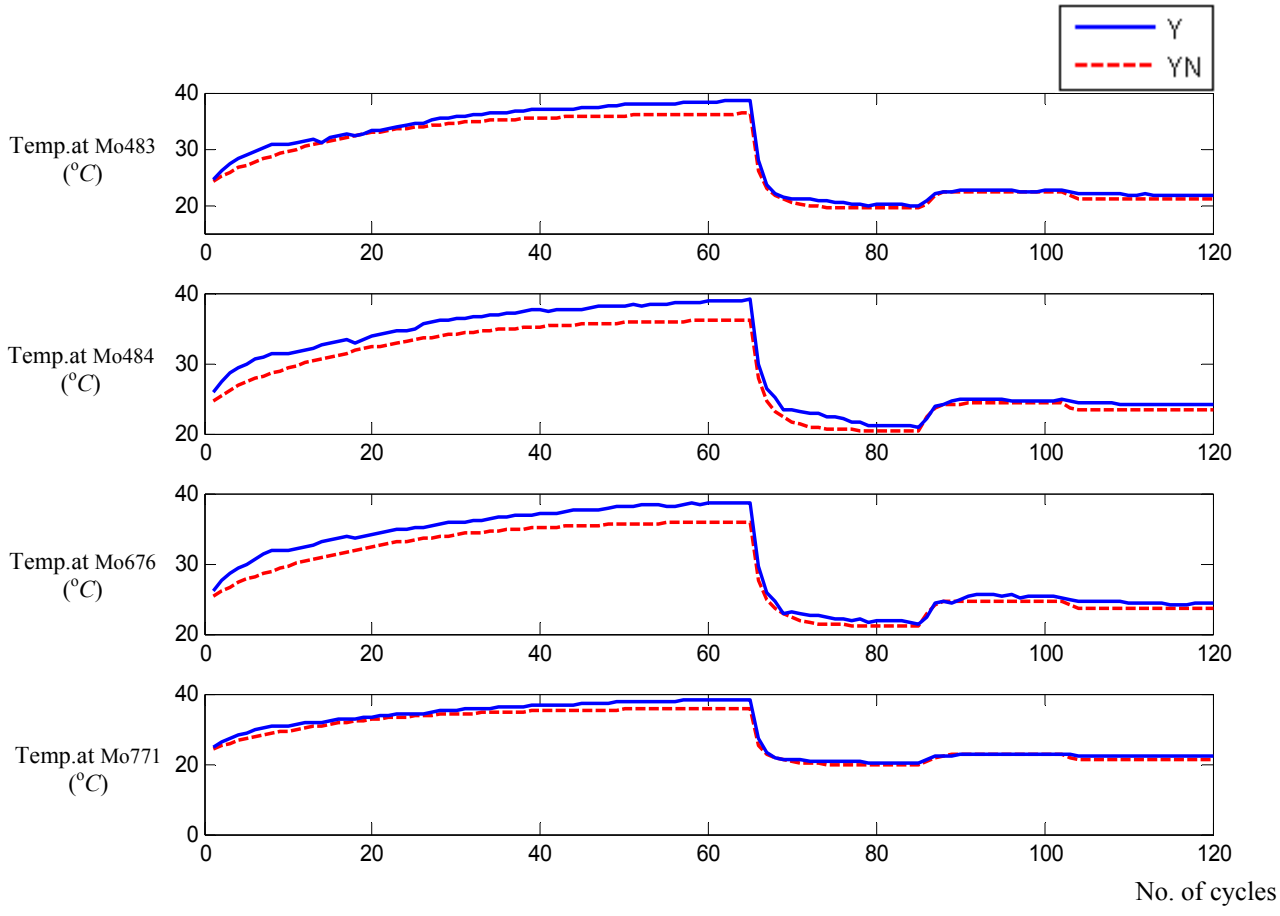


Figure 5-10 Experimental validation results (comparison between model (Y_N) and actual outputs (Y))

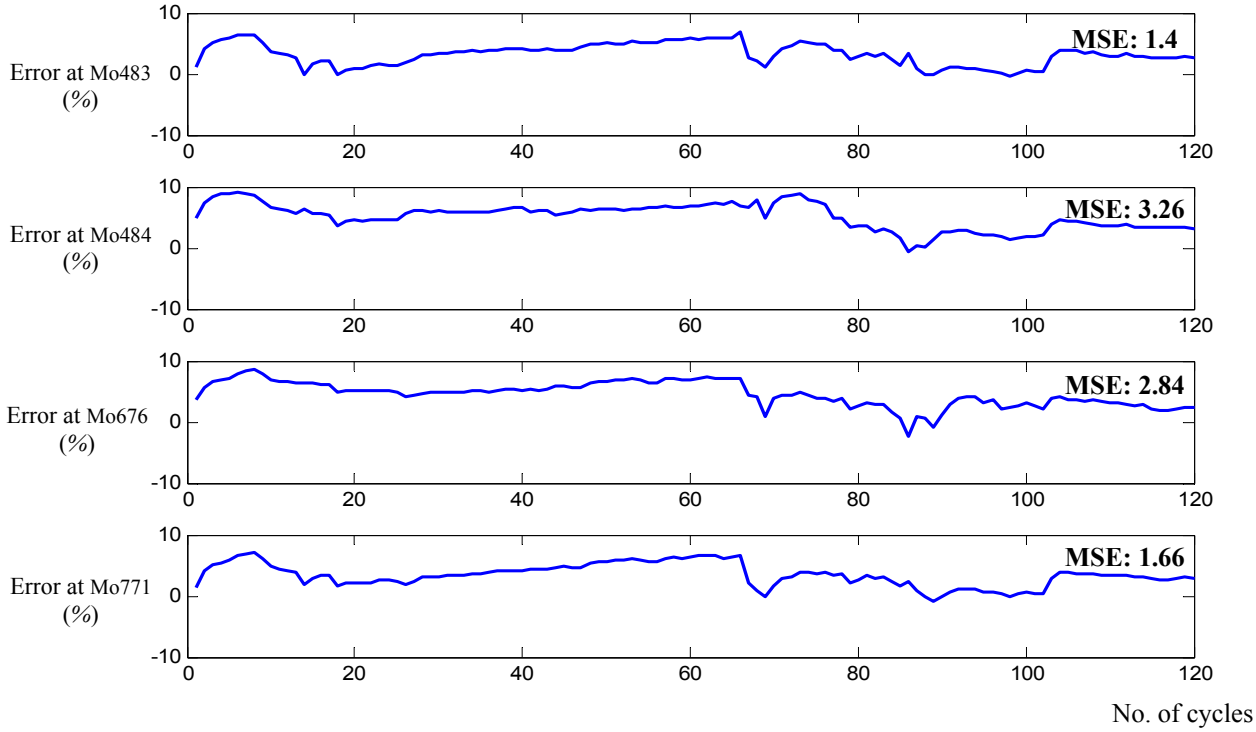


Figure 5-11 Experimental validation results (error between Y_N and Y)

The outputs of the NARX model and the experimental system are compared in Fig. 5-10 and 5-11. From both figures, it can be seen that the NARX model's outputs almost match the experimental data. However, the experimental outputs are slightly higher than the NARX model outputs at node Mo484 and Mo676 for 0 gpm compared to other nodes. Since Mo484 and Mo676 are closer to the narrow injection mould channel (pg 59) into which molten plastic is forced to inject under extremely high pressure, it is possible that the actual temperature at these nodes are higher than the NARX model's prediction especially for 0 gpm. Therefore, a more accurate FEA and a greater number of on-line data sets at these nodes are required to improve the model when there is no coolant.

5.4 Summary

The goal in this chapter was to identify the thermal dynamics of a die system to provide the groundwork for die temperature control. For this, system identification was conducted using data sets obtained from both FEA and experiments, and NN techniques dealing with uncertain dynamics of the die system.

For modeling, the off-line training was carried out first using the off-line algorithm (Levenberg-Marquardt algorithm) and data from the FEA to give initial structure of the NN model (NARX). Then the model was updated by means of the on-line learning algorithm (optimized learning rate based algorithm) and experimental data from the actual injection moulding process. Through validating the model with additional experimental data, the NARX model showed the ability to accurately predict the temperature response at selected sensor locations. After the model validation, the NARX model was then used for controller development.

Chapter 6
Controller design for thermal management

6.1 Problem overview

In the previous section, a model was derived using the NN based system identification method with training data sets to handle system uncertainties. This kind of system identification is particularly effective as a platform for designing adaptive control algorithms [18].

Adaptive controllers have been widely used for industrial applications due to their capabilities in adjusting the system and control parameters for any uncertainty in the control process [29]. However, because conventional adaptive controllers require a linear model, these controllers may not work properly for a complex system with poorly understood dynamics.

Therefore, an NN control and fuzzy logic control will be more appropriate to deal with the temperature control of our die systems due to their features explained in Section 2.4 and 6.2.2, respectively. As NN controller design approaches, self-adaptive PID neural control, direct adaptive inverse control and additive feedforward control will be attempted, which are the most commonly used categories of neural control (refer to the categories outlined in Section 2.4.2).

The designed controller's performance will be evaluated by how well it controls the cavity-wall temperature for product quality under different cycle-times.

6.2 Controller development

6.2.1 Control of cavity-wall temperature

Cooling time is the time that the part cools from melt temperature to demoulding temperature, at which point it is ejected from the cavity. Therefore, the cavity-wall temperature after part ejection can indicate the quality of the final product. However, because the cavity wall is an area where locating sensors is challenging, the temperature measured at four other nodes can be used as proxy for the cavity-wall temperature by utilizing the relation between them. This relation could be obtained from the training data sets used for NARX modeling, and is presented in Fig. 6-1.

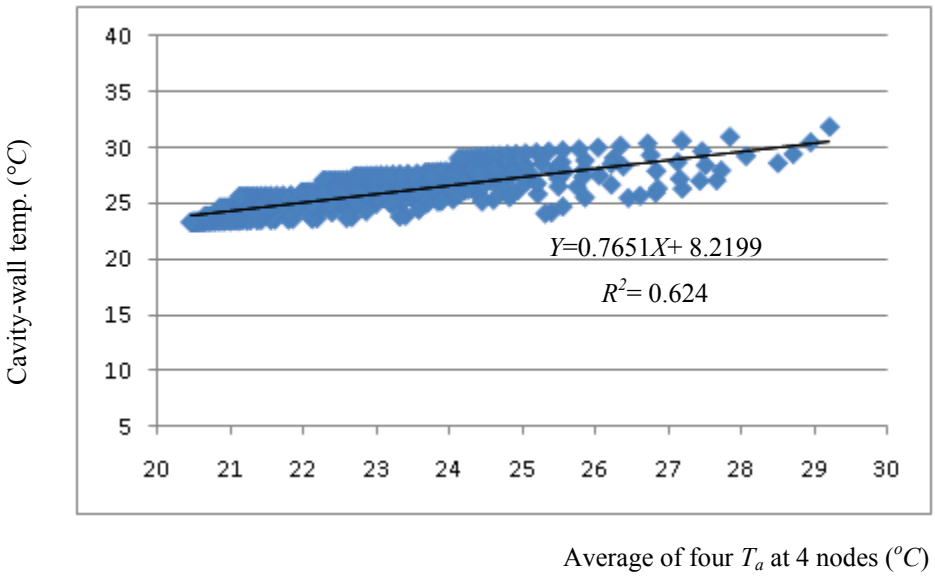


Figure 6-1 Linear relation between an average of four T_a at 4 nodes and cavity-wall temperature right after ejection

Figure 6-1 shows a linear relationship between an average of the four T_a at the four nodes and cavity-wall temperature immediately after ejection. Since an average of four T_a is relatively insensitive to noise, this value is more suitable to use for analyzing the cavity-wall temperature rather than each individual T_a. An R-squared value of 0.624 (R=0.787) represents a strong relationship between an average of four T_a (range: 20.45 – 29.21 ^oC) and cavity-wall temperature after ejection (range: 23.26 – 31.77 ^oC).

When the part reaches the desired demoulding temperature for this material (Santoprene 8211-45: 85 ^oC), the cavity-wall temperature is around 25 ^oC, which corresponds to 22 ^oC for an average of four T_a. Therefore, 22 ^oC will be used as an objective value to control the quality of moulding process (or product quality).

6.2.2 Fuzzy logic control

Fuzzy logic control is an effective approach in uncertain systems for which it is difficult to obtain a mathematical model. By capturing the knowledge of humans into the fuzzy rules, fuzzy logic enables

to implement a controller without an in-depth knowledge on the system dynamics. Figure 6-2 shows the structure of a fuzzy control system used for this study.

In Fig. 6-2, the dynamic model (NARX) was derived through off-line and on-line training described in the previous chapter, and thus reflects the actual plant behavior. Fuzzified input variables are the desired temperature of an average of four T_a, error, and cycle-time. Besides the error variable commonly used in fuzzy control, the desired temperature and cycle-time variables are additionally considered.

As one of the input variables in fuzzy control, the cycle-time affects T_a at each node (and thus average of four T_a). Specifically, a short cycle-time increases T_a when the water is provided (i.e., non zero gpm of the flow rate) whereas it decreases T_a in the case of the non-supply of water. Thus, the flow rate as a fuzzified output variable should be controlled differently along with the cycle-time.

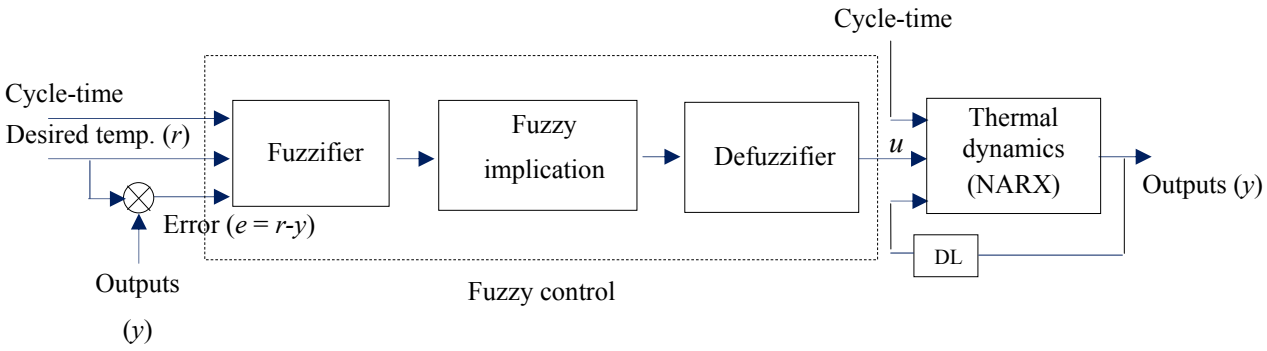


Figure 6-2 Fuzzy control structure

The desired temperature is also introduced as another input variable. This feature distinguishes our control design from that previously developed in fuzzy thermal control area of die systems [11], [70], [71] which use only error and error-change as input variables. As seen in Fig. 5-2 (pg 71), each flow rate cannot make the temperature at certain nodes drop below a certain value (e.g., 23.5 ^oC and 20.7 ^oC for 1gpm and 8gpm, respectively with 91 seconds of cycle-time at Mo484) once it reaches a steady state value. This means each desired temperature value to be controlled needs a different flow rate

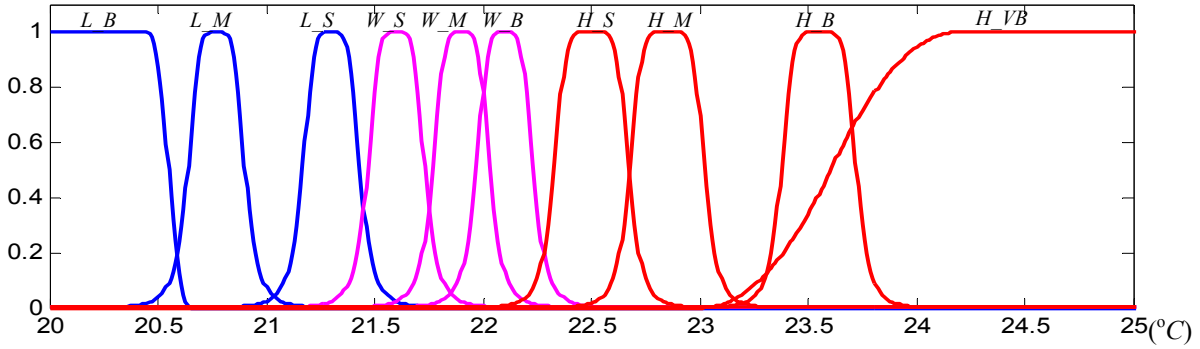
suitable for the corresponding value. Therefore, only error and error-change terms are not enough to control different levels of T_a. For example, if there is the same magnitude of error (e.g., 1 ^oC) between actual temperature and desired value at different levels of desired T_a (e.g., 23.5 ^oC and 20.7 ^oC at Mo484), then the control strategy requires a different operating flow rate to track a different desired temperature (e.g., operating flow rates should vary around 1gpm and 8gpm for maintaining the desired values of 23.5 ^oC and 20.7 ^oC, respectively despite the same error of 1 ^oC).

Using the input and output variables, the fuzzy control rules can be expressed as:

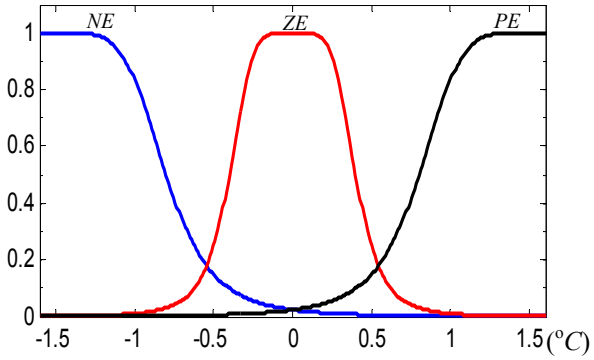
$$\text{Rule: If } DT \text{ is } DT_i \text{ and } ET \text{ is } ET_j \text{ and } Ct \text{ is } Ct_k, \text{ THEN } Q = Q_r \tag{6-1}$$

where DT is the desired average of four T_a, ET is the error between actual temperature and desired value, Ct is the cycle-time, Q is the flow rate of the coolant. *i = LB, ..., HVB*; *j = NE, ZE, PE*; *k* =61, …, 91seconds; *r = Off*, …, *H_b* (refer to the membership functions in Fig. 6-3). All fuzzy rules applied to the thermal control can be drawn out as Table 6-1. For the 9th fuzzy rule in Table 6-1 as an example, if *DT* is *L_S* (Low_Small) and *ET* is *PE* (Positive Error) and *Ct* is 61sec, then *Q=H_b* (*b* (big) level of *H* (High) flow rate).

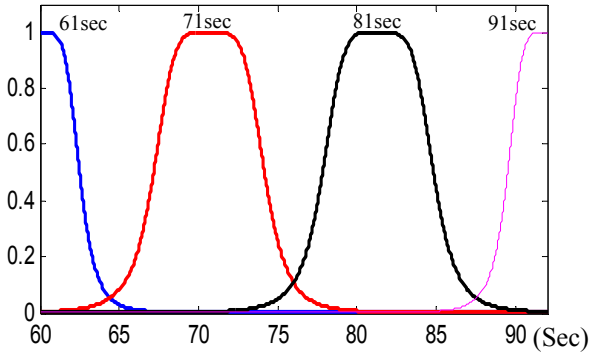
The membership functions and control rules in this fuzzy logic control are obtained by trial and error design procedures. The Mamdani method [72] is used for fuzzy implication (i.e., if-then statement in Eq. (6-1)) and the centroid method [18] is used for defuzzification of a fuzzy control inference.



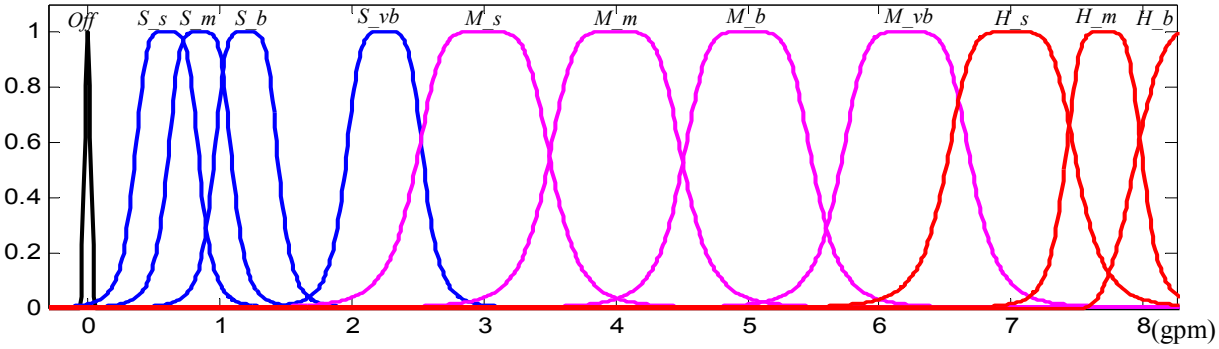
(a) Desired temperature (*DT*)



(b) Error (*ET*)



(c) Cycle-time (*Ct*)



(d) Flow rate (*Q*)

Figure 6-3 Membership functions of fuzzy inputs (a), (b), (c) and output (d)

Table 6-1 Fuzzy control rules

Rule No.	*DT*	*ET* \ *Ct*	61 sec	71 sec	81 sec	91 sec
1	*L_B* (Low_Big)	*NE* (Negative Error)	*H_b* [L]	*H_b*	*H_m* [K]	*H_m*
2		*ZE* (Zero Error)	*H_b*	*H_b*	*H_m*	*H_m*
3		*PE* (Positive Error)	*H_b*	*H_b*	*H_m*	*H_s* [J]
4	*L_M* (Low_Medium)	*NE*	*H_b*	*H_b*	*H_m*	*H_m*
5		*ZE*	*H_b*	*H_b*	*H_m*	*H_s*
6		*PE*	*H_b*	*H_b*	*H_s*	*M_vb* [I]
7	*L_S* (Low_Small)	*NE*	*H_b*	*H_m*	*H_m*	*H_s*
8		*ZE*	*H_b*	*H_m*	*H_s*	*M_vb*
9		*PE*	*H_b*	*H_m*	*M_vb*	*M_b* [H]
10	*W_S* (Warm_Small)	*NE*	*H_m*	*H_m*	*H_s*	*M_vb*
11		*ZE*	*H_s*	*H_s*	*M_vb*	*M_b*
12		*PE*	*M_vb*	*M_b*	*M_b*	*M_m* [G]
13	*W_M* (Warm_Medium)	*NE*	*H_s*	*H_s*	*M_vb*	*M_b*
14		*ZE*	*M_vb*	*M_vb*	*M_b*	*M_m*
15		*PE*	*M_b*	*M_b*	*M_m*	*M_s* [F]
16	*W_B* (Warm_Big)	*NE*	*M_vb*	*M_b*	*M_m*	*M_m*
17		*ZE*	*M_b*	*M_m*	*M_m*	*M_s*
18		*PE*	*M_m*	*M_s*	*M_s*	*S_vb* [E]
19	*H_S* (High_Small)	*NE*	*M_b*	*M_b*	*M_m*	*M_s*
20		*ZE*	*M_m*	*M_m*	*M_s*	*S_vb*
21		*PE*	*M_s*	*S_vb*	*S_vb*	*S_b* [D]
22	*H_M* (High_Medium)	*NE*	*M_m*	*M_m*	*S_vb*	*S_vb*
23		*ZE*	*M_s*	*S_vb*	*S_b*	*S_b*
24		*PE*	*S_vb*	*S_b*	*S_b*	*S_b*
25	*H_B* (High_Big)	*NE*	*M_s*	*S_vb*	*S_vb*	*S_vb*
26		*ZE*	*S_vb*	*S_b*	*S_b*	*S_b*
27		*PE*	*Off*	*Off*	*Off*	*Off*
28	*H_VB* (High_Very Big)	*NE*	*S_m* [C]	*S_m*	*S_m*	*S_s* [B]
29		*ZE*	*Off* [A]	*Off*	*Off*	*Off*
30		*PE*	*Off*	*Off*	*Off*	*Off*

^{A} means *zero* flow rate; $^{B,C,D\,E}$ mean *s* (small), *m* (medium), *b* (big), *vb* (very big) levels of *S* (Small) flow rate; ^{F,G,H,I} mean *s* (small), *m* (medium), *b* (big), *vb* (very big) levels of *M* (Medium) flow rate; ^{J,K,L} mean *s* (small), *m* (medium), *b* (big) levels of *H* (High) flow rate.

6.2.3 Self-tuning PID neural control

In this section, self-tuning PID control schemes in which neural networks based on backpropagation sand radial basis function are used to tune the parameters of the controller are described.

6.2.3.1 PID based on BP neural network learning

As an alternative approach, a self-tuning PID controller with the backpropagation (BP) learning algorithm is applied. This PID controller overcomes a limitation of conventional PID controllers in that the determination of control parameters is not easy in nonlinear or uncertain systems, by the self-learning ability of tuning PID parameters online [73].

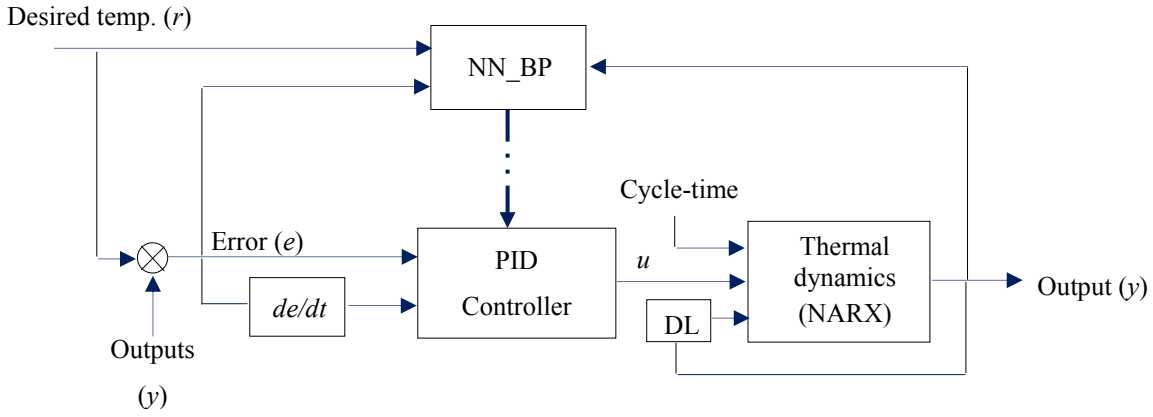


Figure 6-4 Self-adaptive PID control with BP neural network learning

The schematic diagram of the self-adaptive PID neural controller based on BP neural network learning is given in Fig. 6-4. The incremental PID control algorithm is given as follows:

$$\begin{aligned} u(k) &= u(k-1) + \Delta u(k) \\ &= u(k-1) + K_P \Delta e(k) + K_I e(k) + K_D \Delta^2 e(k) \end{aligned} \tag{6-2}$$

where $u(k)$ is the control variable at the time instant k, and K_P, K_I, K_D are the proportional, integral and derivative gains respectively. $e(k) = r(k) - y(k)$ is the current control error; $\Delta e(k) = e(k) - e(k-1)$; $\Delta^2 e(k) = e(k) - 2e(k-1) + e(k-2)$.

A three-layer neural network with BP was used with the following structure:
The inputs and outputs in the input layer are given by

$$\begin{aligned} x_j(k) &= [x_1(k),\ x_2(k),\ x_3(k)] \\ &= [r(k),\ y(k),\ e(k)] \\ o_j^{(i)}(k) &= \begin{cases} x_j(k), & j = 1,\ 2,\ 3 \\ 1, & j = 4 \end{cases} \end{aligned} \tag{6-3}$$

where $x_j(k)$ and $o_j^{(i)}(k)$ are the inputs and outputs of the input layer. $r(k)$ and $y(k)$ are the desired values of the average of the four T_a and the output of NARX model, respectively. *i* denotes the input layer.

The inputs and outputs in the hidden layer and these in the output layer are defined in Eq. (6-4) and Eq. (6-5), respectively.

$$\begin{aligned} net_l^{(h)}(k) &= \sum_{j=1}^{n} w_{lj}^{(h)} o_j^{(i)}(k) \\ o_l^{(h)}(k) &= \begin{cases} f[net_l^{(h)}(k)], & l = 1,\ 2,\ ...,\ p-1 \\ 1, & l = p \end{cases} \end{aligned} \tag{6-4}$$

where $net_l^{(h)}$, $o_l^{(h)}$, $w_{lj}^{(h)}$ are the inputs, outputs, and weights in the hidden layer. $net_r^{(o)}$, $o_r^{(o)}$, $w_{rl}^{(o)}$ are the inputs, outputs, and weights in the output layer. f and g are tanh(*x*) and (1+tanh(*x*))/2, respectively. *h* and *o* denote the hidden layer and the output layer, respectively.

Using the gradient-descent algorithm, the weights in the output and hidden layer are updated by the following equations based on [74] (Section 2.3.3).

$$\begin{aligned} \Delta w_{rl}^{(o)}(k) &= \eta \delta_r^{(o)} o_l^{(h)}(k) + \gamma \Delta w_{rl}^{(o)}(k-1) \\ \delta_r^{(o)} &= e(k) \frac{\partial y(k)}{\partial \Delta u(k)} \frac{\partial \Delta u(k)}{\partial o_r^{(o)}(k)} g'[net_r^{(o)}(k)], \quad r=1, 2, 3 \end{aligned} \tag{6-5}$$

$$\begin{aligned} \Delta w_{lj}^{(h)}(k) &= \eta \delta_l^{(h)} o_j^{(i)}(k) + \gamma \Delta w_{lj}^{(h)}(k-1) \\ \delta_l^{(h)} &= f'[net_l^{(h)}(k)] \sum_{r=1}^{3} \delta_r^{(o)} w_{rl}^{(o)}(k), \quad l=1, 2, ..., p-1 \end{aligned} \tag{6-6}$$

where η is the leaning speed rate and γ is the momentum factor for convergence. g' and f' are the derivatives of g and f.

6.2.3.2 PID based on RBF neural network

Another type of neural network using the radial basis function (RBF) is considered for the design of the self-tuning PID controller, due to the advantages of the RBF neural network such as simple structure, faster learning algorithms and effective mapping between a controlling system's input and output [75].

Figure 6-5 describes the structure of the self-adaptive PID neural controller based on an RBF neural network. The main function of this control technique is to first identify the plant dynamics and secondly to adjust the PID parameters adaptively based on RBF neural network identification (specifically, Jacobian information, $\frac{\partial y}{\partial \Delta u}$ (see Eq. (6-13)).

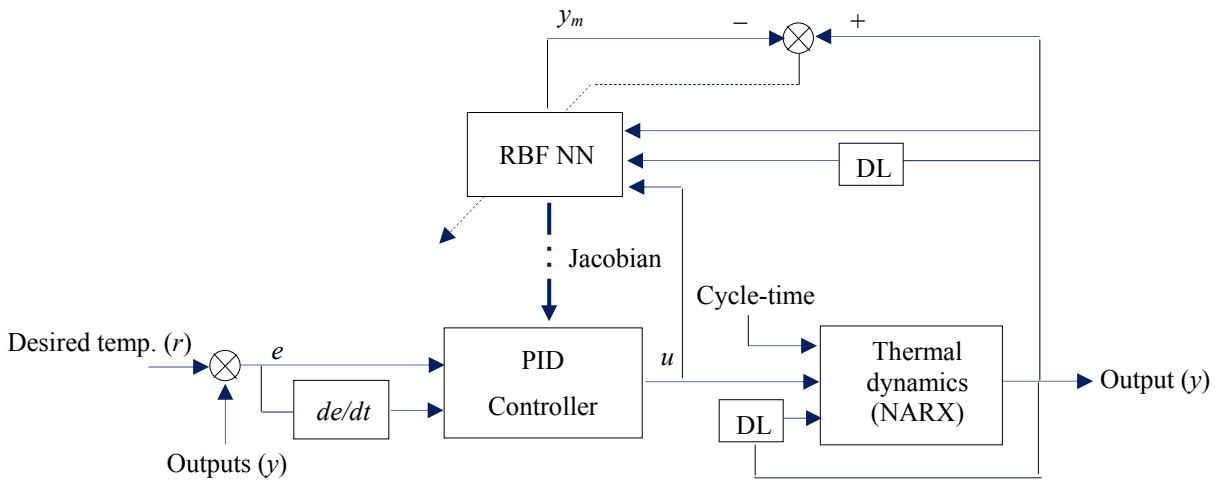


Figure 6-5 Self-adaptive PID control with RBF neural network

The input vector, output vector in the hidden layer, and weight vector between hidden layer and output layers in the RBF network are respectively:

$$\begin{aligned} X &= [x_1,\ x_2,\ ...,\ x_n]^T \\ &= [\Delta u(k),\ y(k),\ y(k-1)]^T \\ H &= [h_1,\ h_2,\ ...,\ h_m]^T \\ W &= [w_1,\ w_2,\ ...,\ w_m]^T \end{aligned} \tag{6-7}$$

where X, H and W are the input vector, output vector in the hidden layer and the weight vector between hidden and output layers. $\Delta u(k)$ and $y(k)$ are defined in Eq. (6-2) and Eq. (6-3), respectively. Each element (h_j) of vector H has a format of the Gaussian kernel function given by:

$$\begin{aligned} h_j &= \exp\left[\left(\left\|X - C_j\right\|^2\right) / 2b_j^2\right], \quad j = 1,\ ...\ , m \\ C_j &= [c_{1j},\ c_{2j},\ ...,\ c_{ij},\ ....,\ c_{nj}]^T \end{aligned} \tag{6-8}$$

where h_j is the Gaussian kernel function, C_j and b_j are the center and width of h_j.

The output of the RBF network, y_m is expressed as:

$$y_m = w_1 h_1 + w_2 h_2 + \cdots + w_m h_m \tag{6-9}$$

To update W, b_j and C_j, the cost function required is:

$$J(k) = \frac{1}{2}\left[y(k) - y_m(k) \right]^2 \tag{6-10}$$

By using the gradient-descent technique and Eq. (6-11), W, b_j and C_j are updated according to the following algorithms [63], [76].

$$\begin{aligned}
w_j(k) &= w_j(k-1) - \eta^* \frac{\partial J}{\partial w_j(k)} + \alpha^* \left(w_j(k-1) - w_j(k-2) \right) + \beta^* \left(w_j(k-2) - w_j(k-3) \right) \\
b_j(k) &= b_j(k-1) - \eta^* \frac{\partial J}{\partial b_j(k)} + \alpha^* \left(b_j(k-1) - b_j(k-2) \right) + \beta^* \left(b_j(k-2) - b_j(k-3) \right) \\
c_{ij}(k) &= c_{ij}(k-1) - \eta^* \frac{\partial J}{\partial c_{ij}(k)} + \alpha^* \left(c_{ij}(k-1) - c_{ij}(k-2) \right) + \beta^* \left(c_{ij}(k-2) - c_{ij}(k-3) \right)
\end{aligned} \tag{6-11}$$

where η^* is the leaning speed rate. α^* and β^* are the momentum factors.

By the gradient descent method, the adjustment of PID parameters [76] is given by

$$\begin{aligned}
K_P(k) &= K_P(k-1) + \eta^{**} e(k) \frac{\partial y}{\partial \Delta u} \Delta e \\
K_I(k) &= K_I(k-1) + \eta^{**} e(k) \frac{\partial y}{\partial \Delta u} e \\
K_D(k) &= K_D(k-1) + \eta^{**} e(k) \frac{\partial y}{\partial \Delta u} \Delta^2 e(k)
\end{aligned} \tag{6-12}$$

where η^{**} is the leaning speed rate for PID parameter update and $\frac{\partial y}{\partial \Delta u}$ is the Jacobian information of controlled plant.

6.2.4 Adaptive inverse control

As another type of NN based adaptive control, direct adaptive inverse control and additive feedforward control are introduced in the following section.

6.2.4.1 Direct adaptive inverse control

Direct inverse control is one approach for designing the neural controller. The popularity of this approach is its simplicity (i.e., leading to an overall transfer function of unity), a faster execution and tolerance to a range of noises [18]. In this section, direct adaptive inverse control is utilized in order to achieve good control performance in the presence of uncertainties in a laminated die.

Based on the schematic of the NARX model represented in Fig. 5.3 with one additional time delay of the water flow rate variable (i.e., $u_2(k\text{-}1)$ in Eq. (6-14)) , the controlled system can be described as:

$$y_N(k+1) = f(y(k),\ u_1(k),\ u_2(k),\ u_2(k-1)) \tag{6-13}$$

where $y(k)$ is the process output (i.e., $y(k)=[y_{Mo483}(k);\ y_{Mo484}(k);\ y_{Mo676}(k);\ y_{Mo771}(k)]$), $y_N(k)$ is the neural network output, $u_1(k)$ is the cycle-time, $u_2(k)$ is the water flow rate and f is the NARX model. 1 in $u_2(k\text{-}1)$ means one cycle-time delay and k is the discrete time index.

Then, the inverse neural model from Eq. (6-14) can be expressed as:

$$u_2(k) = f^{-1}(y_N(k+1),\ y(k),\ u_1(k),\ u_2(k-1)) \tag{6-14}$$

where f^{-1} is the inverse neural model.

Using Eq. (6-15), a schematic of direct adaptive inverse control is represented in Fig. 6-6. In the schematic, $y_N(k+1)$ is replaced by the desired temperature, $r(k+1)$ [77]. The inverse neural model (AINC in Fig. 6-6) is trained adaptively in real-time control using the on-line learning algorithm (i.e., optimized learning rate based algorithm) in Section 5.3.1.

The initial weights and biases of the inverse neural model were given by following the same procedure through which the thermal dynamics model (NARX) was obtained previously. Specifically, the inverse neural model was first identified by off-line training with the same FEA data and Levenberg-Marquardt algorithm, and then was updated through identical experimental data and on-line learning algorithm used in Chapter 5. Then, weights and biases provided after training using the on-line algorithm were utilized as an initial structure of the inverse neural model for real-time control.

Data of $y(k)$, $u_1(k)$ and $u_2(k)$ in the schematic was normalized by Eq. (5-2) and (5-3), and the node number of hidden layer in AINC (in Fig. 6-6) was chosen as eight by trial and error.

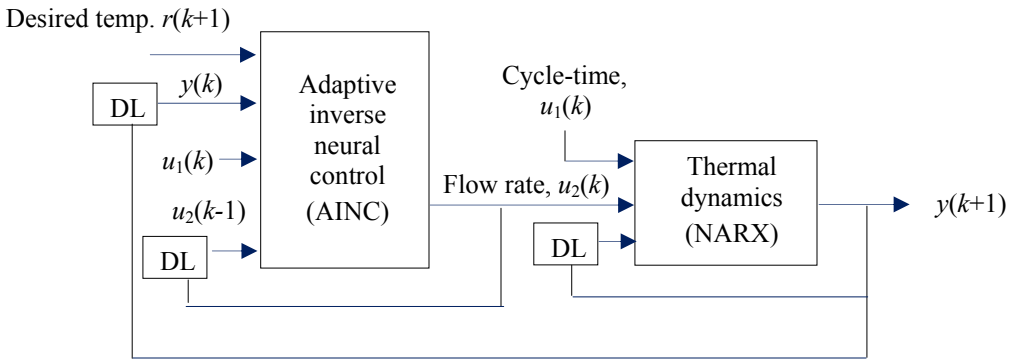


Figure 6-6 Direct adaptive inverse control

6.2.4.2 Additive feedforward control

To improve the performance of direct adaptive inverse control, Additive Feedforward Control (AFC) is provided. AFC is a control strategy whose principle is to add an additional inverse control (direct adaptive inverse control in our case) to an existing controller such as PID [78]. In other words, AFC combines a model based feedforward controller and a PID controller (i.e., PID feedback loop).

Whereas the pure AFC (Fig. 6-7) as a type of AFC does not receive any information from the plant, the mixed AFC takes the feedback information (see the dash-line in Fig. 6-8) from the plant [79]. In the present work, the mixed AFC is chosen.

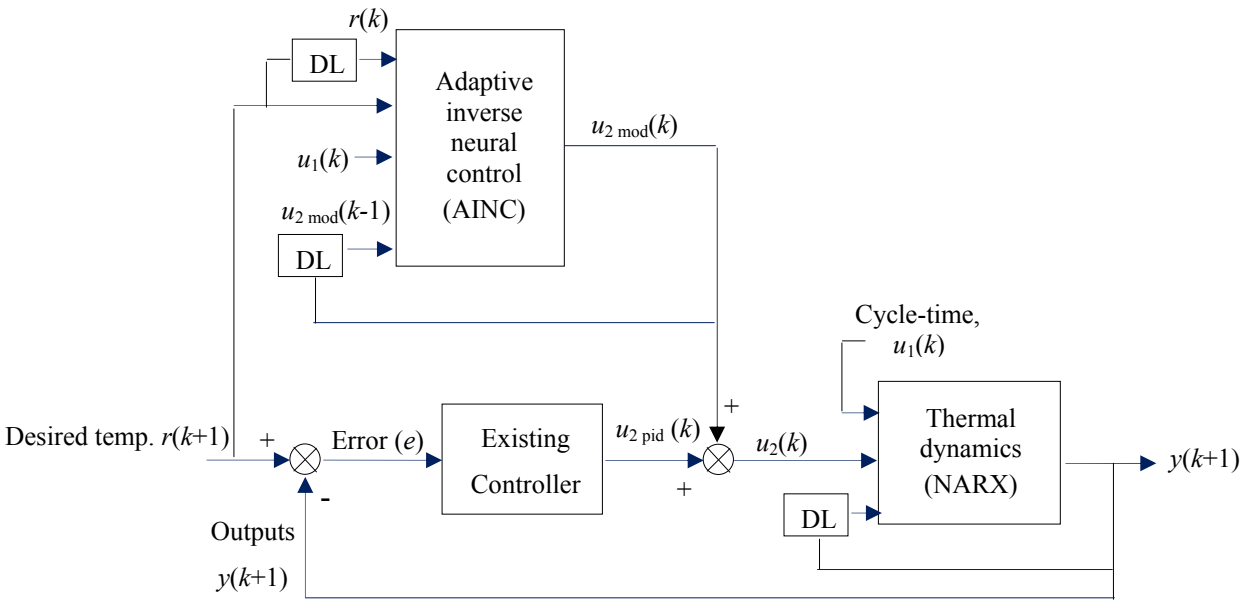


Figure 6-7 Pure additive feedforward control

Compared to the case of direct adaptive inverse control in Fig. 6-6, the complete control signal of the flow rate, $u_2(k)$ consists of the control signal, $u_{2\ \mathrm{mod}}(k)$ from the inverse model and the control signal, $u_{2\ \mathrm{pid}}\ (k)$ from the existing controller. As an existing controller, we applied the conventional PID controller whose parameters are tuned manually (i.e., by trial and error), and self-adaptive PID controllers with BP and RBF introduced in Section 6.2.3.

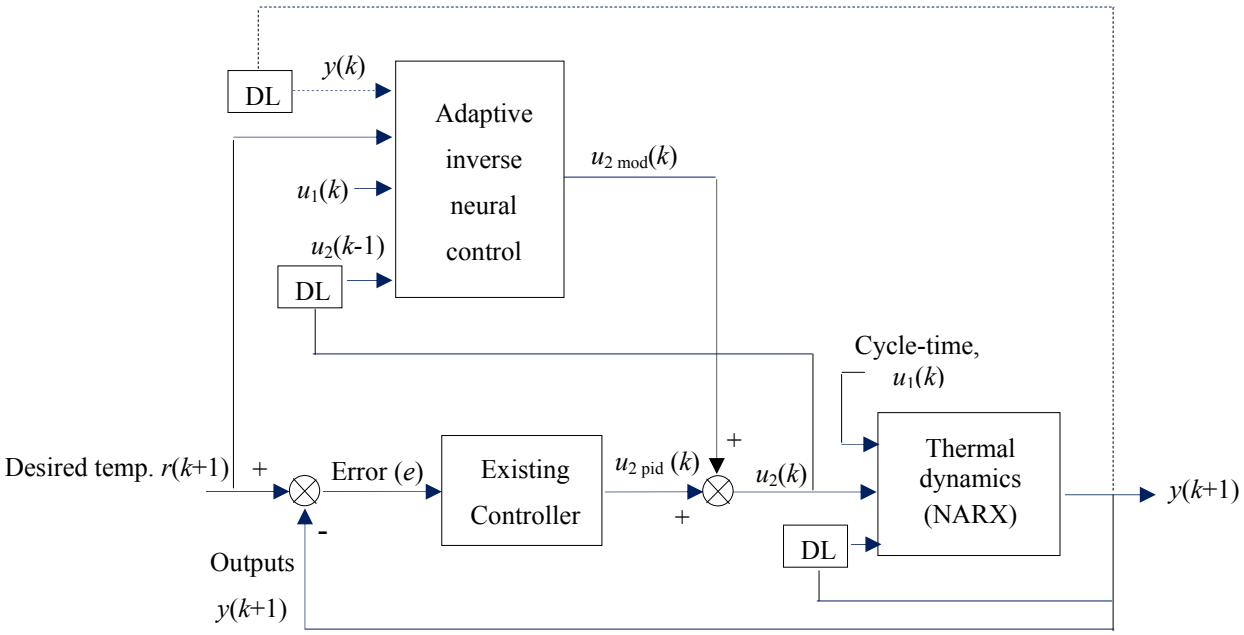


Figure 6-8 Mixed additive feedforward control

6.3 Control performance

The performance of each controller developed in Section 6.2 is compared in terms of tracking accuracy and response time. First of all, the tracking performance of the fuzzy-logic controller and the self-adaptive PID controllers with BP and RBF were evaluated using MATLAB/Simulink. Figure 6-9 shows the variation of the average of four T_a (at 4 nodes) in the NARX model (thermal dynamics) with each controller. For the first range (0-6000 second), the cycle-time of 91 seconds (thus 66 cycles) is considered with multi-setpoints of 25 ^oC and 22 ^oC (which corresponds to the demoulding temperature). Then, the set-point is increased to 24 ^oC and kept constant during 6000 – 12000 second with a different cycle-time of 71 seconds (total 85 cycles). Then multi-setpoints of 22 ^oC and 25 ^oC are used during 12000 – 18000 second with the cycle-time of 61 seconds (total 99 cycles).

Although the fuzzy-logic controller shows an acceptable control performance for all cycle-times, there is some error in tracking during 6000 – 12000 second (error: -0.075 ^oC) and 12000 – 18000 second (errors: -0.1 ^oC for 22 ^oC).

Self-tuning PID controllers with BP and RBF show better tracking performance for all ranges. However, the PID controller with the BP reacts relatively slowly. For example, it takes 41 cycles (= 192 – 151 cycles) for the PID controller with BP to reach 22 ^{o}C right after changing the cycle-time from 71 to 61 seconds, whereas those with RBF and fuzzy logic require 8 and 7 cycles, respectively.

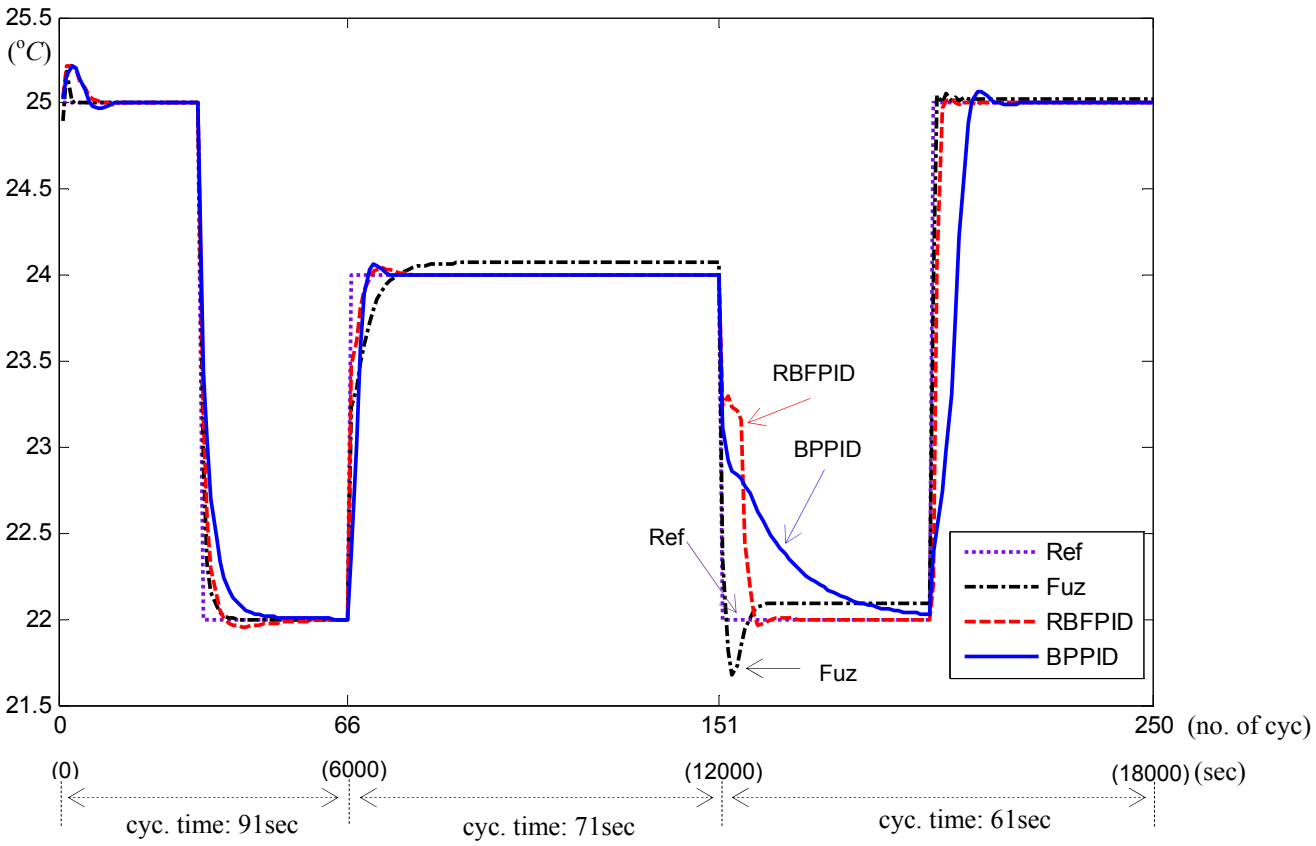


Figure 6-9 Performance of fuzzy-logic controller (Fuz), self-adaptive PID controller with BP (BPPID) and self-adaptive PID controller with RBP (RBFPID) for multi-setpoints with various cycle-times

The flow rate (control variable) of each controller is presented in Fig. 6-10, corresponding to the controller performances described in Fig. 6-9.

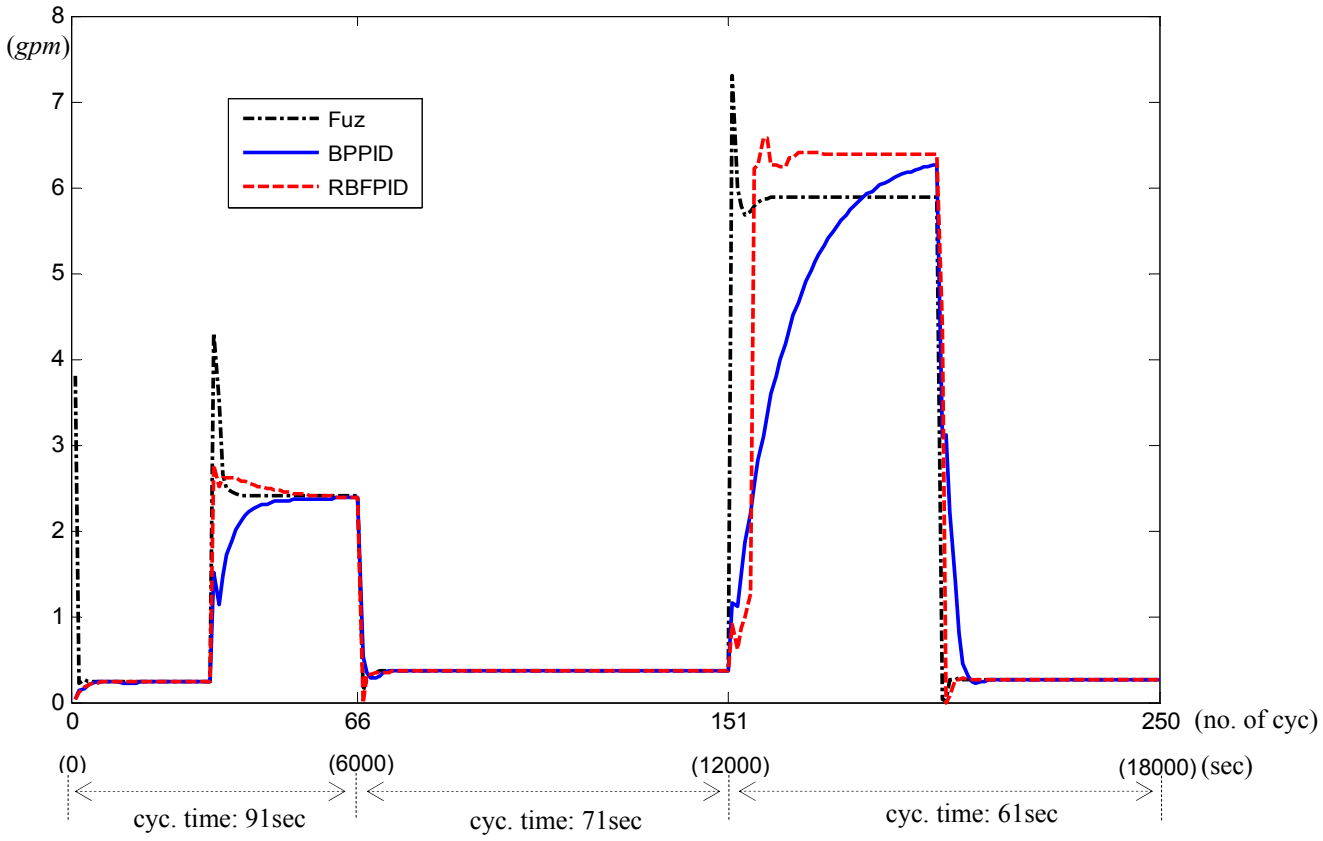
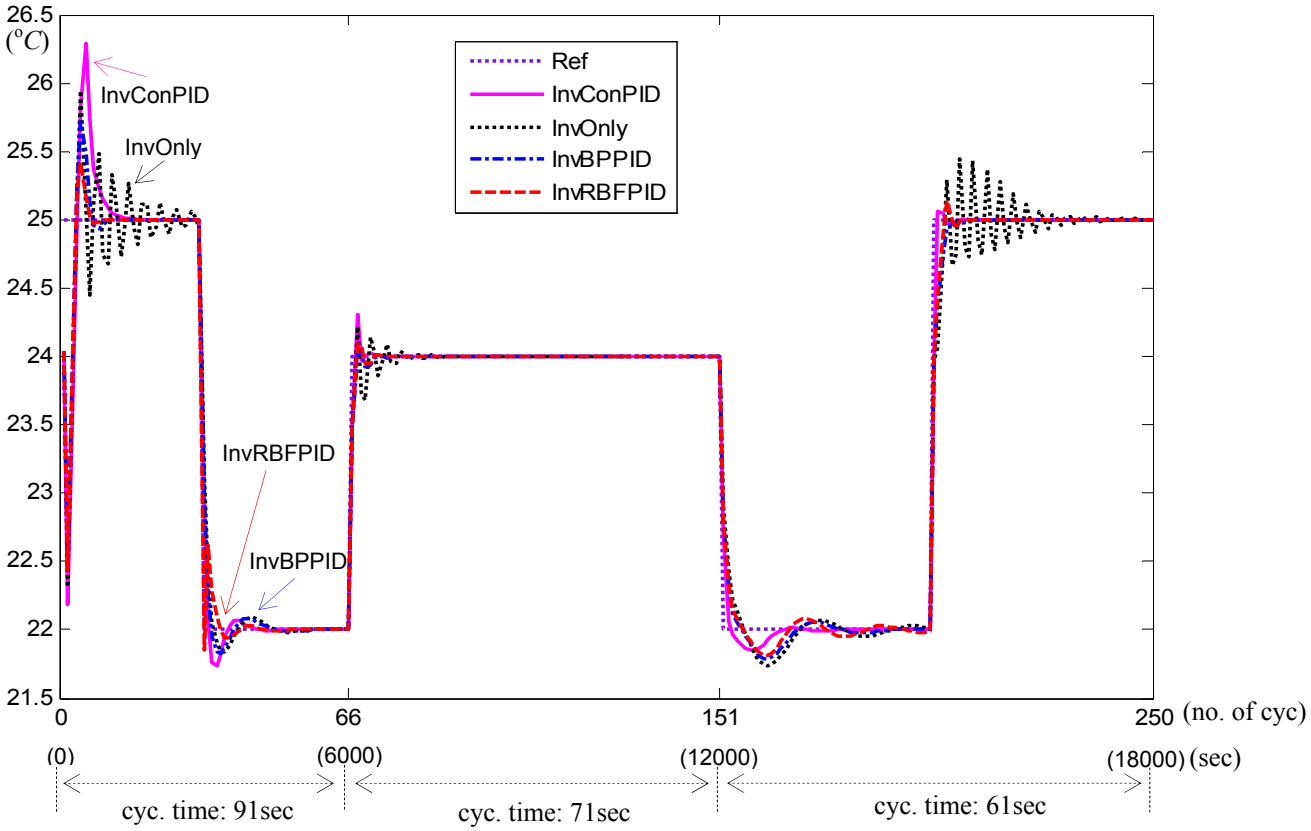


Figure 6-10 Coolant flow rate (control variable) of fuzzy-logic controller (Fuz), self-adaptive PID controller with BP (BPPID) and self-adaptive PID controller with RBP (RBFPID) for multi-setpoints with various cycle-times

By adopting four control techniques (direct adaptive inverse control, AFC with self-adaptive PID with BP, AFC with self-adaptive PID with RBF and AFC with conventional PID) aforementioned in Section 6.2.4, the performance of these controllers was evaluated. In Fig. 6-11, the direct adaptive inverse control by itself has less accurate tracking performance compared to the AFCs with PID controllers because it shows some oscillations around 25 ^{o}C (0 – 6000 second) , 24 ^{o}C (6000 – 12000 second) and 25 ^{o}C (12000 – 18000 second). The PID controllers added to the direct adaptive inverse control improve the performance, but the AFCs having adaptive PID with BP and RBF are superior to the one with the conventional PID in terms of accuracy and response time. These results are well described in the second subfigure in Fig. 6-11. Also, the control variable (flow rate) corresponding to the results in Fig. 6-11 is shown in Fig. 6-12.



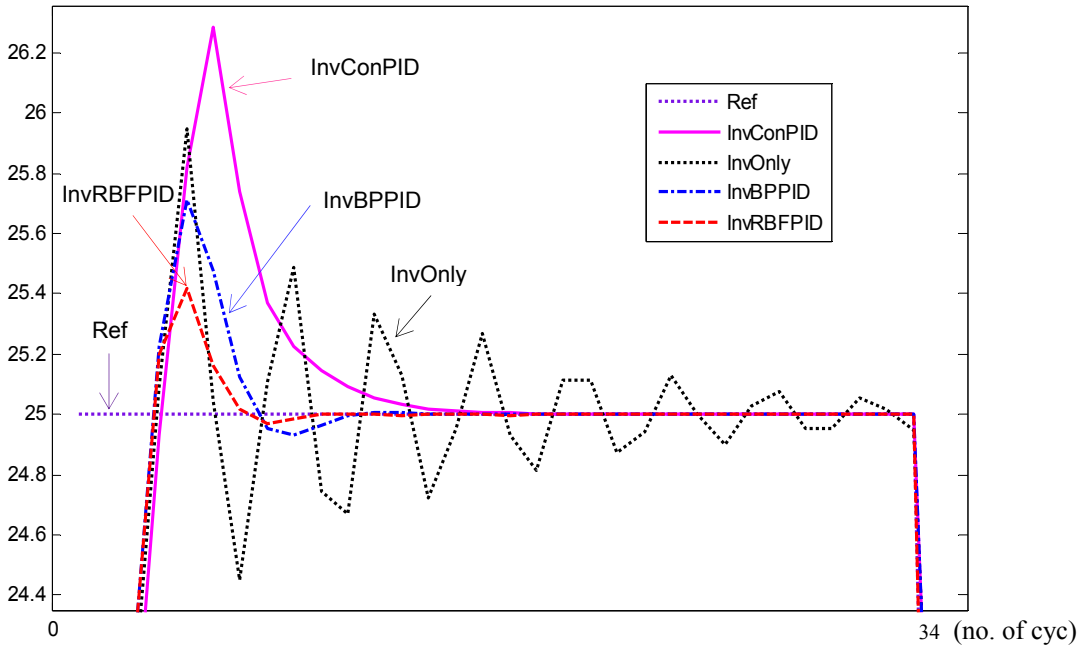


Figure 6-11 Performance of direct adaptive inverse controller (InvOnly), AFC with self-adaptive BP PID (InvBPPID), AFC with self-adaptive RBF PID (InvRBFPID) and AFC with conventional PID (InvConPID) for multi-setpoints with various cycle-times

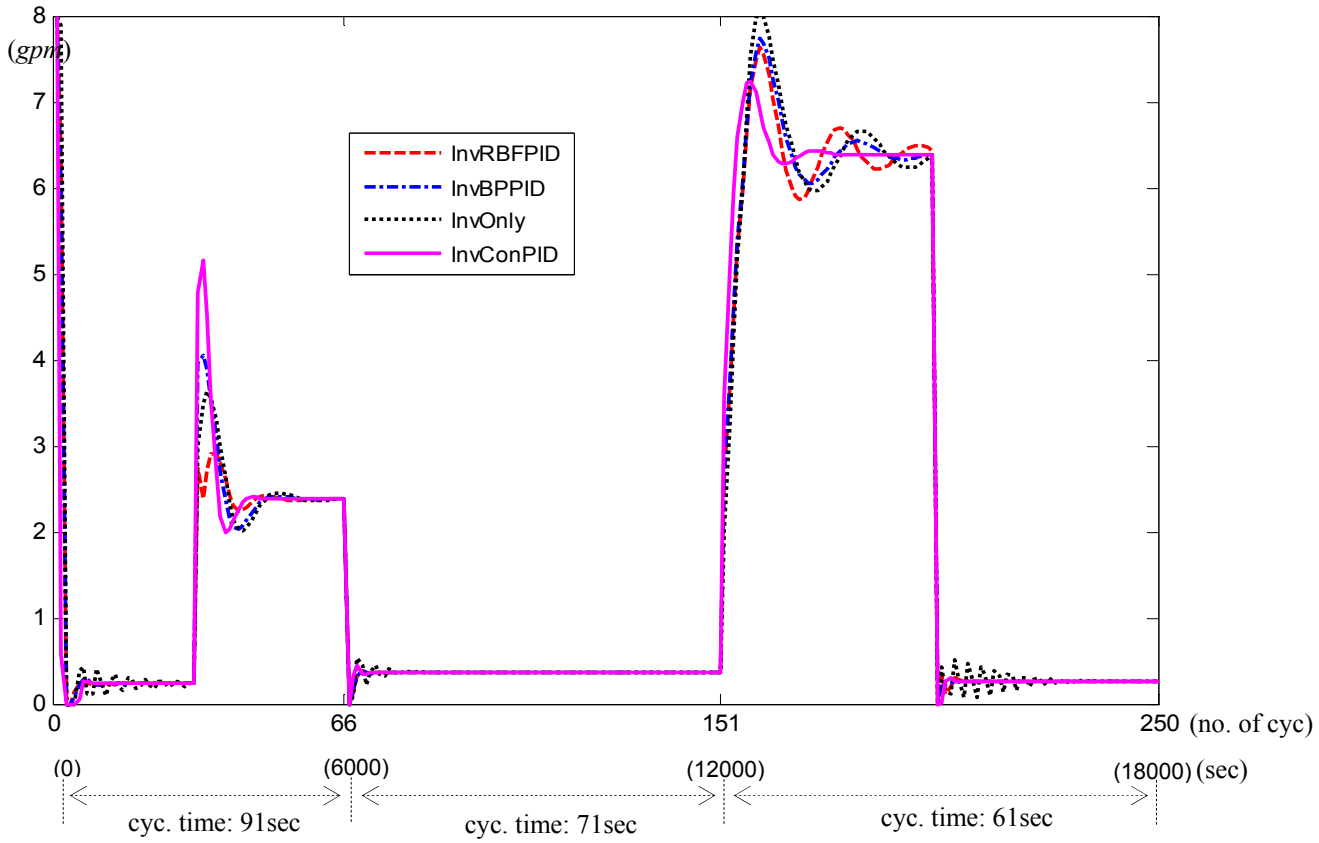


Figure 6-12 Coolant flow rate (control variable) of direct adaptive inverse controller (InvOnly), AFC with self-adaptive BP PID (InvBPPID), AFC with self-adaptive RBF PID (InvRBFPID) and AFC with conventional PID (InvConPID) for multi-setpoints with various cycle-times

Tracking errors of all controllers shown in Fig. 6-9 and 6-11 (except the fuzzy-logic controller and the direct adaptive inverse control having the least accuracy in each figure) are presented in Table 6-2. Through comparing the mean squared error (MSE) over 18000 seconds, it is noted that an addition of inverse control to an existing controller (PID) in the AFC contributes to improving tracking accuracy. For example, MSE values in the cases of inverse control with self-adaptive PID with BP (0.0511) and RBF (0.0470) are less than those of only self-adaptive PID with BP (0.1746) and RBF (0.0757).

Table 6-2 Mean squared error (MSE) of each controller

Controller type	MSE
Self-adaptive PID with BP	0.1746
Self-adaptive PID with RBF	0.0757
AFC with conventional PID	0.0654
AFC with self-adaptive PID with BP	0.0511
AFC with self-adaptive PID with RBF	0.0470

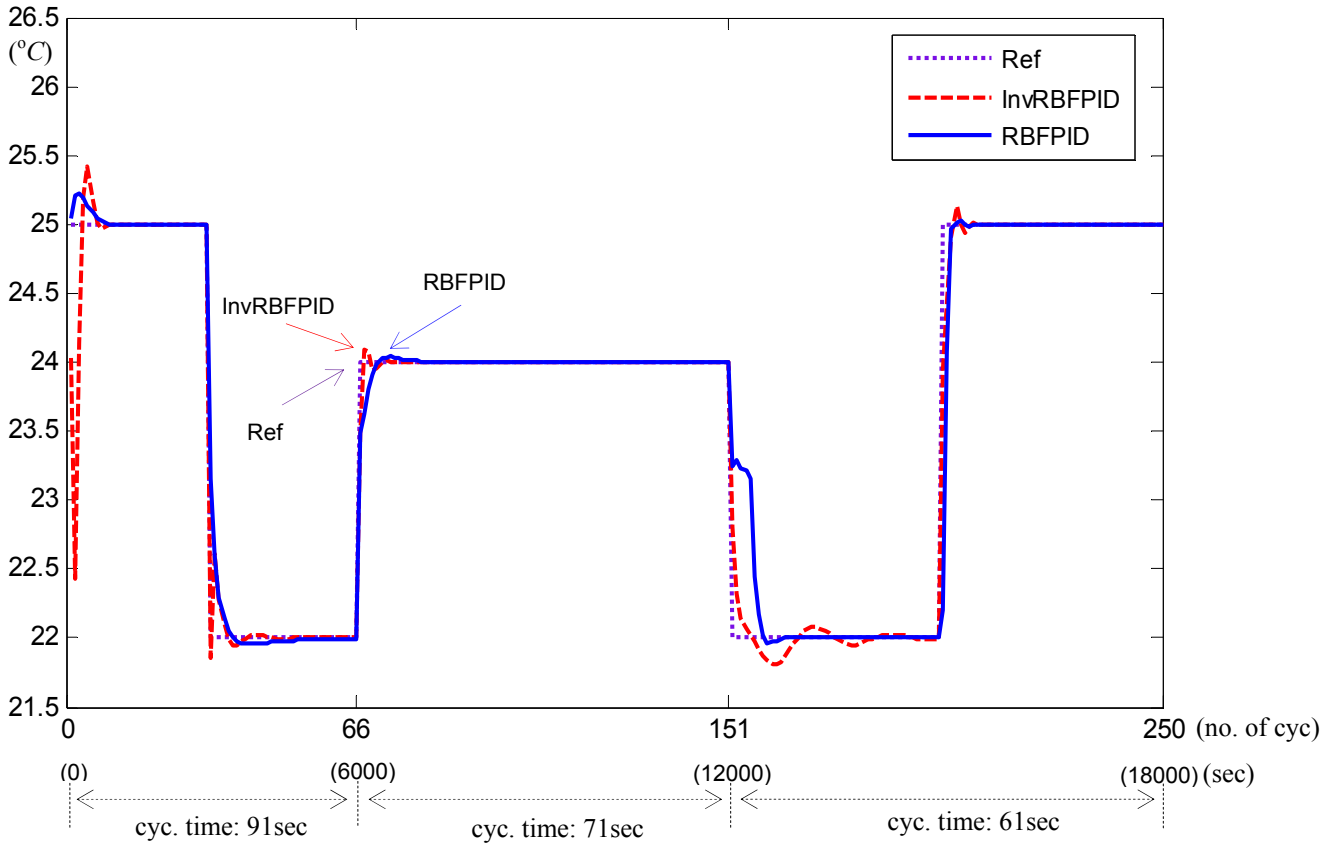


Figure 6-13 Comparison of controller performance between self-adaptive PID with RBF and inverse control with self-adaptive PID with RBF for multi-setpoints with various cycle-times

Finally, a comparative study (Fig. 6-13) between self-adaptive PID with RBF and inverse control with self-adaptive PID with RBF that are the best control strategies in Fig. 6-9 and 6-11, respectively, was carried out in terms of response time. Except at the beginning of the running time, inverse control with self-adaptive PID with RBF shows faster response time especially whenever the setpoint rapidly changes.

6.4 Summary

To cover various cycle-times (various thermal dynamics) of moulding process, a fuzzy logic controller needs the refined membership functions for the output variable (flow rate) as seen in Fig. 6-3. This makes the controller design more complicated. In spite of having a refinement of membership functions, it is difficult to provide an accurate performance for all cycle-times. The self-tuning PID neural controllers show better performance than this fuzzy-logic controller. Between self-tuning PID controller with RBF and one with BP, the RBF based self-tuning PID has faster response than the others.

The direct adaptive inverse controller and additive feedforward controller (AFC) (adding the direct adaptive inverse control to PID controller (either conventional PID or self-adaptive PID)) were also considered. AFC shows better quality of tracking accuracy and response time than the adaptive inverse control only, and AFC with self-adaptive PID outperforms one with conventional PID control whose parameters are tuned manually. A comparison of results of tracking error (MSE) and response time verifies that AFC with self-tuning PID for both BP and RBF showed more accurate control performance than self-tuning PID itself. From all obtained results, AFC with RBF based self-tuning PID is superior to the others in terms of accurate tracking of cavity-wall temperature and faster response under thermal dynamics with various cycle-times in the injection mould process.

The performance of all controllers was evaluated by simulation. Therefore, although the AFC with RBF based self-tuning PID shows the best results, self-tuning PID control (with either RBF or BP) or fuzzy control may be more practical in real-time since an addition of inverse control can increase the computational burden. Therefore, a careful validation of controller performance through real-time implantation is required as further work.

Chapter 7
Conclusion

7.1 Summary

This thesis addresses the thermal control strategies for plastic injection moulding in a laminated die, which includes methodologies for finding sensor locations as the first step, modeling the dynamics as the second step and designing a temperature controller as the last step. These methodologies are based on finite element analysis, clustering, sensitivity analysis and neural network techniques.

First of all, sensor locations were identified by applying a clustering method (using the temperature ratio) and sensitivity analysis. Sensor locations were obtained by conducting finite element analysis of plastic injection moulding process using ANSYS CFX 11 software.

For the modeling of thermal dynamics of a die as the next step, neural network (NN) techniques with off-line and on-line training algorithms were adopted to tackle the problem of various cycle-times for moulding process and uncertain dynamics of the die. The off-line training using the data set gathered from the finite element analysis provided an initialized structure of the NN for modeling. Then, experimental data set was used in an on-line training algorithm to improve the model. The identified model was validated using experimental tests and showed a good result in predicting the temperature distributions at selected sensor locations.

Based on this model, several temperature controllers including fuzzy-logic, self-adaptive PID controllers using BP and RBF, direct adaptive inverse neural-control and additive feedforward control (by combining a feedforward control (direct adaptive inverse control) and self-adaptive PID control) were designed as the last step. Through a comparative study for each controller's performance, the fuzzy controller showed less accurate tracking despite refined membership functions for output variables to cope with various cycle-times. By utilizing learning algorithms to tune control parameters on-line, both BP and RBF self-adaptive PID controllers exhibited good control performance. Between two self-adaptive PID controllers, the self-tuning PID controller with RBF exhibits faster response than the one with BP.

Direct adaptive inverse neural-control using the optimized learning rate based algorithm, and additive feedforward control having both adaptive inverse control and PID control were revised. Direct adaptive inverse neural-control showed less accuracy in tracking by presenting some

oscillations. Additive feedforward control showed improved performance over direct adaptive inverse controller itself, fuzzy controller and self-adaptive PID controllers. Among all designed controllers, additive feedforward control with the RBF based self-tuning PID was the best from the viewpoints of tracking accuracy and response rate for the quality of plastic injection moulding process with diverse cycle-times.

Additionally, this study verified that conformal cooling channels provide more efficient cooling than conventional straight cooling channels in terms of uniform temperature and cooling time. This result confirms the advantages of using conformal cooling channels as a cooling strategy.

7.2 Contributions

From the results, it can be seen that this study provides a control strategy for a better thermal management in the plastic injection moulding process by covering three themes as follows:

First, a method for determining optimal sensor locations to estimate the temperature distribution was introduced, which has not been addressed in the literature dealing with die systems. This work has a prerequisite for an effective modeling and controller design development.

Secondly, the use of both off-line and on-line algorithms using NN techniques and the inclusion of multiple-cycle times were considered as a method for system identification. It provided accuracy of the model considering uncertainties and thermal dynamic variations of the die. Conventional linear approximation methods with a fixed cycle-time usually result in either modeling error or limited identification.

Thirdly, various controllers by means of fuzzy-logic and neural control techniques were developed and the best controller among them for achieving cavity-wall temperature control was identified.

By combining these methodologies, this study can fundamentally contribute to formulating a unique framework for mould thermal control. Also, since the overall methods in the study can be applicable to the thermal management problem of general systems, these methods can be extended to other relevant industrial problems including plastic processing.

7.3 Future work

Future work includes the following tasks:

- The optimal sensor location study can be expended to deal with the effect of heat transfer on measurement noise and cost effective aspect for sensor configuration.

- Sensor locations identified by other clustering methods such as the grid and subtractive clustering algorithms can be compared with those by the method proposed in this study for efficiency of the clustering algorithm and accuracy of the temperature estimation.

- Although the derived model well matched with the experimental outputs, the developed controller's performance was only tested through simulation. Thus, experimental verifications are needed for further evaluation.

- To better verify cooling effects of conformal cooling channels with respect to the cycle-time, the quality of the product after ejection can be examined in terms of porosity and properties.

- This study is limited to the injection moulding process for plastic materials (Santoprene 8211-45) which has thermal properties of lower conductivity and higher specific heat than metallic materials. Therefore, the application of thermal control using conformal cooling channels to metal casting (e.g., Aluminum) may be considered as future extensions to this work.

Appendix A

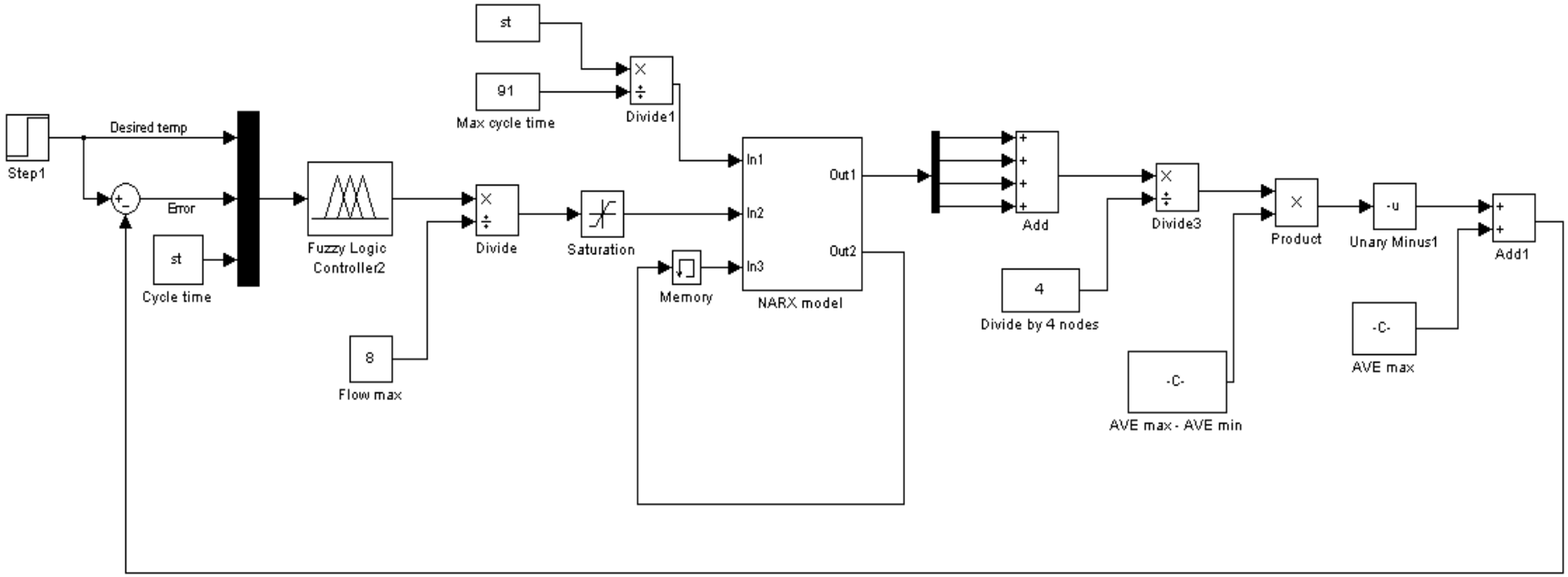


Figure A-1 Simulink model of fuzzy-logic controller

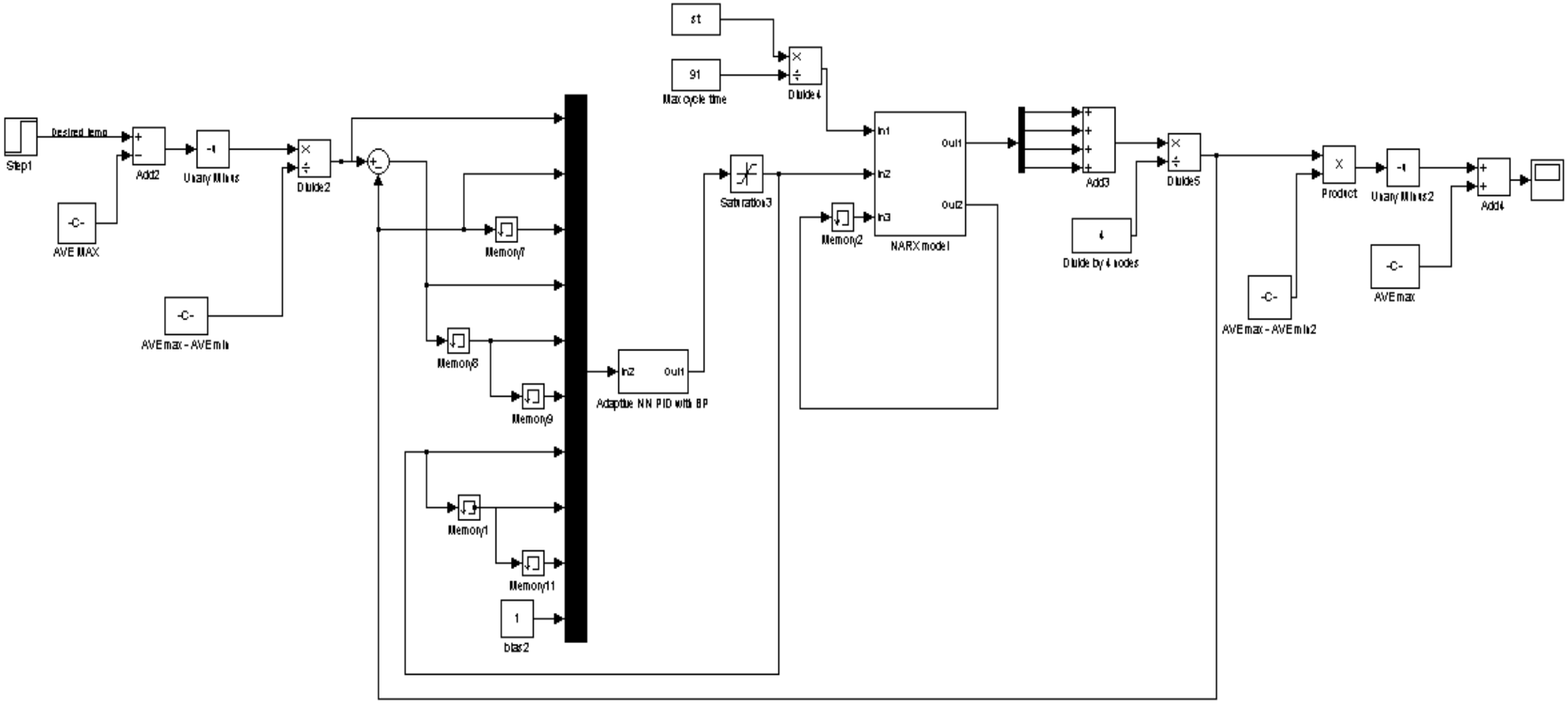


Figure A-2 Simulink model of self-adaptive PID controller with BP neural network learning

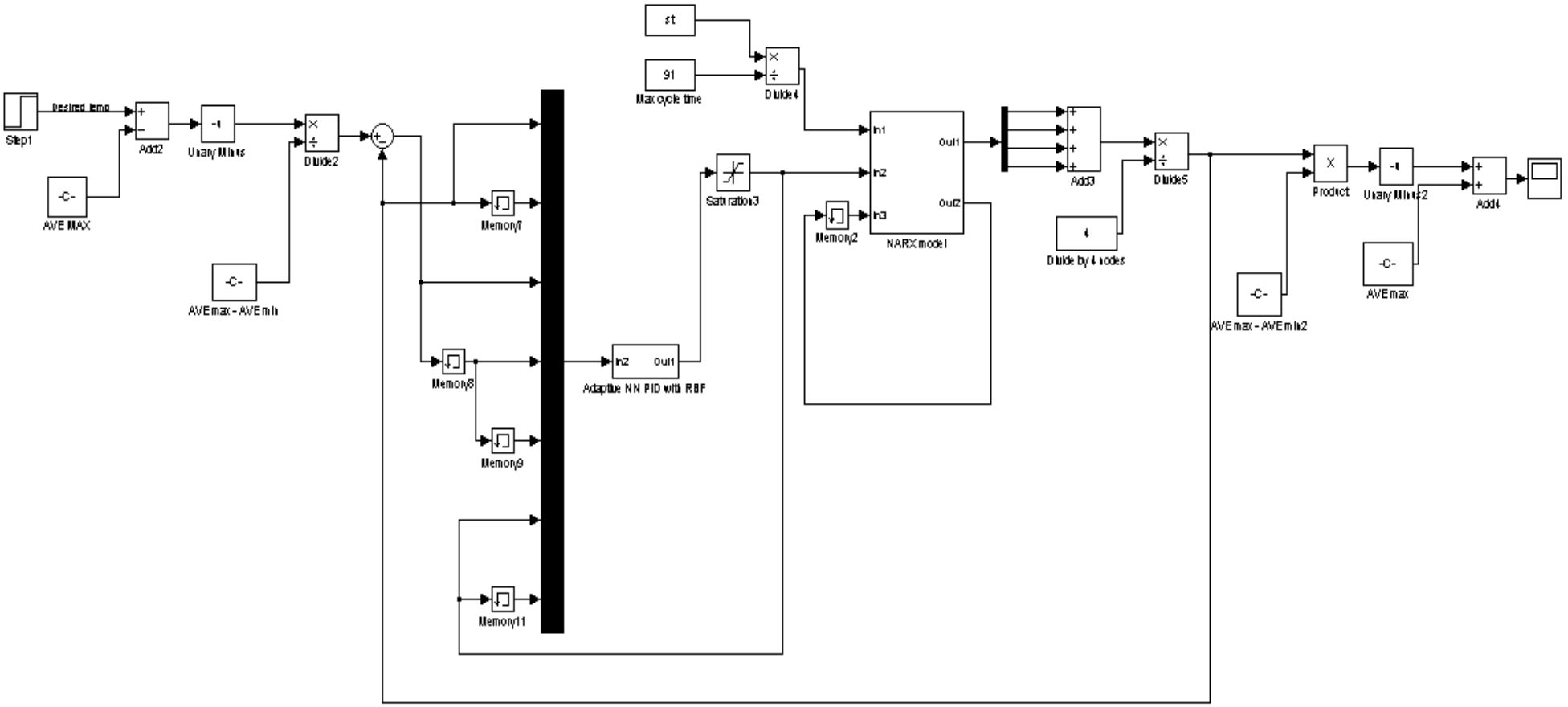


Figure A-3 Simulink model of self-adaptive PID controller with RBF neural network

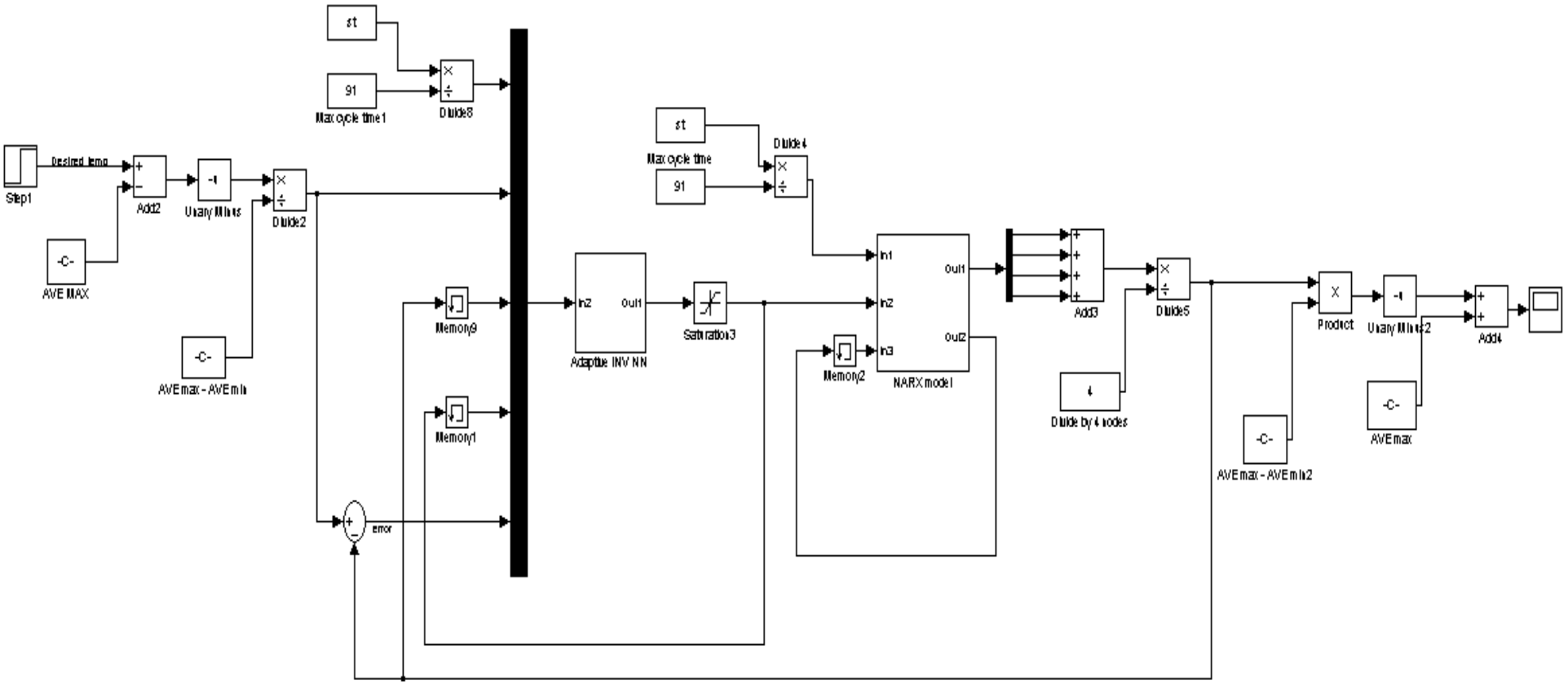


Figure A-4 Simulink model of direct adaptive inverse control

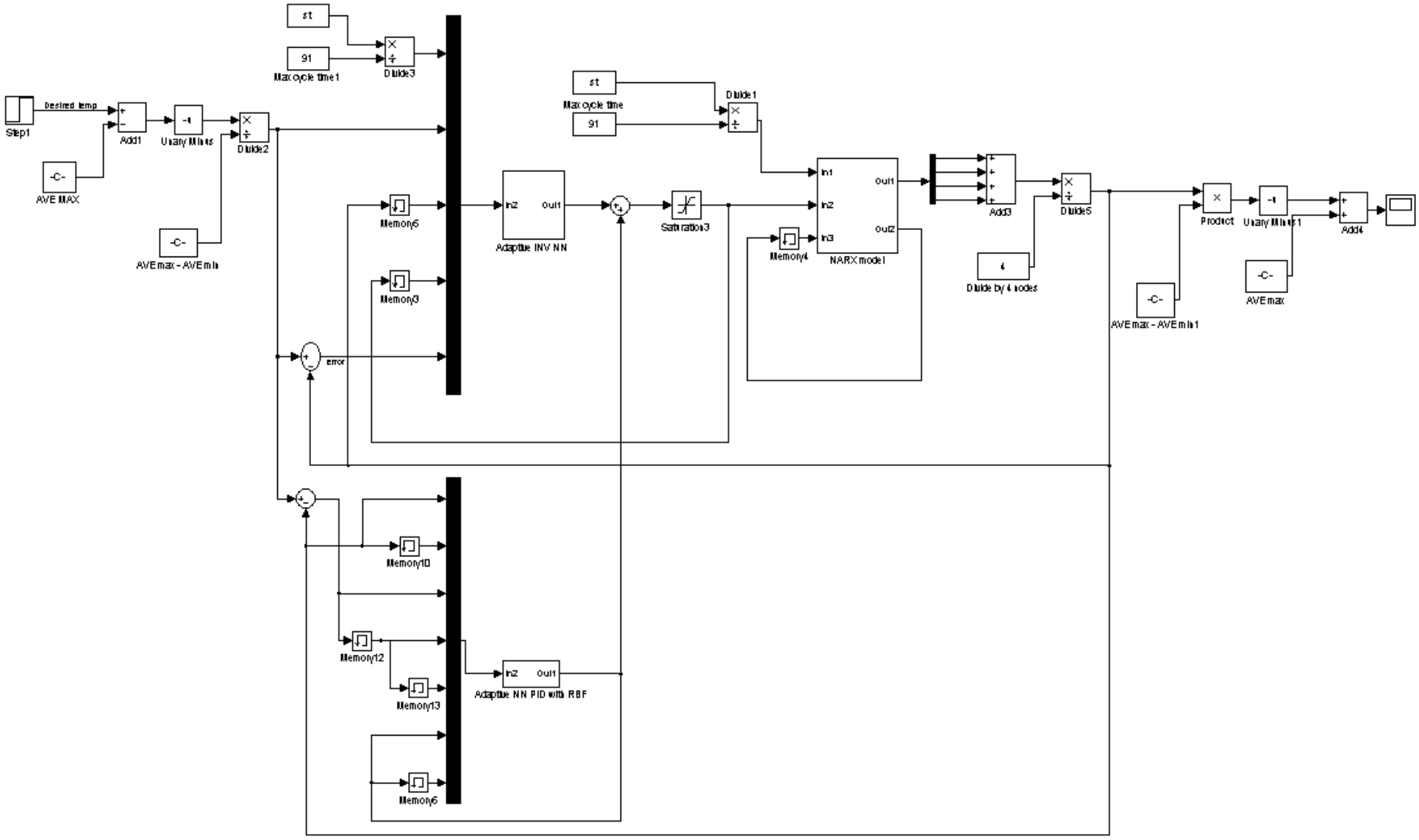


Figure A-5 Simulink model of additive feedforward controller (inverse control with self-adaptive RBF PID control)

Appendix B

- MATLAB code used in Fig. 3-16

```
% 4  locations K-means with representative nodes
Tot_opt_result=[];
%Tot_opt_result_train=[];

hidnode=10;

%=======================================
% Training data set
%=======================================

ym1es=load ('m0572train.txt');
ym2es=load ('m0577train.txt');
ym3es=load ('m1244train.txt');
ym4es=load ('m1249train.txt');
ym5es=load ('m1916train.txt');
ym6es=load ('m1921train.txt');

ym1re=load ('m0257train.txt');
ym2re=load ('m1033train.txt');
ym3re=load ('m1894train.txt');
ym4re=load ('m2155train.txt');

um1=load ('WaterTemptrain312ea.txt');
um2=load ('FlowRatetrain312ea.txt');

nf = 1;
ng = 1;

na = 1;
nb = 1;
```

```
nc = 1;
nd = 1;

naa = 1;
nbb = 1;
ncc = 1;
ndd = 1;
nee = 1;
nff = 1;

offset =max(max(max(max(max(max(max(max(max(max(na,nb),nc),nd),naa),nbb),ncc),ndd), nee),nff),
nf),ng)+1;

Nt=length(ym1es);

yt1es = ym1es(1:Nt);
yt2es = ym2es(1:Nt);
yt3es = ym3es(1:Nt);
yt4es = ym4es(1:Nt);
yt5es = ym5es(1:Nt);
yt6es = ym6es(1:Nt);

yt1re = ym1re(1:Nt);
yt2re = ym2re(1:Nt);
yt3re = ym3re(1:Nt);
yt4re = ym4re(1:Nt);

ut1 = um1(1:Nt);
ut2 = um2(1:Nt);

phitnor=[];
ytonor=[];
ytoall=[];

for t=offset:1:Nt
```

```
phit = [yt1es(t-1)  yt2es(t-1) yt3es(t-1) yt4es(t-1) yt5es(t-1) yt6es(t-1) ut1(t-1)  ut2(t-1) yt1re(t-1)  yt2re(t-1)
yt3re(t-1) yt4re(t-1) ];
pnorf = [max(yt1es) max(yt2es) max(yt3es) max(yt4es) max(yt5es) max(yt6es) max(ut1)  max(ut2) max(yt1re)
max(yt2re) max(yt3re) max(yt4re) ];
phit1=phit./pnorf ;
phitnor = [phitnor; phit1];

yto = [yt1es(t) yt2es(t) yt3es(t) yt4es(t) yt5es(t) yt6es(t)];
ynorf = [max(yt1es) max(yt2es) max(yt3es) max(yt4es) max(yt5es) max(yt6es)];
yto1=yto./ynorf;
ytonor = [ytonor; yto1];
ytoall=[ytoall;yto];

end;

epochs=200;
goal=0;
layer_sizes = [hidnode 6];
function_types={'tansig','purelin'};
training_method='trainlm';

PR = [min(phitnor)' max(phitnor)'];

phitnor = phitnor';
ytonor = ytonor'
ytoall=ytoall';

net = newff(PR,layer_sizes,function_types,training_method);
net.trainParam.goal=goal;
net.trainParam.epochs = epochs;
net = train(net,phitnor,ytonor);
% Simulaiton for training data
yhatt1 = sim(net,phitnor);
```

```
for i=1:1:6
yhatt(i,:)=yhatt1(i,:)*ynorf(i);
end;

%Tot_mse_train= mse(ytoall - yhatt);
%Opt_result_train=[hidnode, Tot_mse_train]
%Tot_opt_result_train=[Tot_opt_result_train; Opt_result_train];


%=======================================
% Testing data set
%=======================================

ym1tes=load ('m0572test.txt');
ym2tes=load ('m0577test.txt');
ym3tes=load ('m1244test.txt');
ym4tes=load ('m1249test.txt');
ym5tes=load ('m1916test.txt');
ym6tes=load ('m1921test.txt');

ym1tre=load ('m0257test.txt');
ym2tre=load ('m1033test.txt');
ym3tre=load ('m1894test.txt');
ym4tre=load ('m2155test.txt');

um1t=load ('WaterTemptest48ea.txt');
um2t=load ('FlowRatetest48ea.txt');

Nts=length(ym1tes);

y1tses = ym1tes(1:Nts);
y2tses = ym2tes(1:Nts);
y3tses = ym3tes(1:Nts);
y4tses = ym4tes(1:Nts);
y5tses = ym5tes(1:Nts);
```

```
y6tses = ym6tes(1:Nts);

y1tsre = ym1tre(1:Nts);
y2tsre = ym2tre(1:Nts);
y3tsre = ym3tre(1:Nts);
y4tsre = ym4tre(1:Nts);
u1ts = um1t(1:Nts);
u2ts = um2t(1:Nts);

phittsnor=[];
ytsall=[];

for t=offset:1:Nts

phitts  = [y1tses(t-1)  y2tses(t-1) y3tses(t-1) y4tses(t-1) y5tses(t-1) y6tses(t-1)  u1ts(t-1) u2ts(t-1) y1tsre(t-1)
y2tsre(t-1) y3tsre(t-1) y4tsre(t-1) ];
ptsnorf = [max(y1tses)  max(y2tses) max(y3tses) max(y4tses) max(y5tses) max(y6tses)  max(u1ts) max(u2ts)
max(y1tsre)  max(y2tsre) max(y3tsre) max(y4tsre) ];
phitts1=phitts./ptsnorf ;
phittsnor = [phittsnor; phitts1];

ytots = [y1tses(t) y2tses(t) y3tses(t) y4tses(t) y5tses(t) y6tses(t)];
ytsall=[ytsall;ytots];

end;

ytsall=ytsall';
phittsnor = phittsnor';

yhatts1 = sim(net,phittsnor);
ytsnorf = [max(y1tses) max(y2tses)  max(y3tses) max(y4tses)  max(y5tses) max(y6tses)];

for i=1:1:6

yhatts(i,:)=yhatts1(i,:)*ytsnorf(i);
```

```
end;

% MSE for testing data set
Tot_mse= mse(ytsall-yhatts);
string('============= 4 locations K-means with representative nodes ==================')
Opt_result=[hidnode, Tot_mse]
Tot_opt_result=[Tot_opt_result; Opt_result];
```

References

[1] Dargusch, Matthew S., Dour, G., Schauer, N., Dinnis, C. and Savage, G., 2006, "The influence of pressure during solidification of high pressure die cast aluminium telecommunications components", Journal of Materials Processing Technology, Vol. 180(1-3), pp. 37-43.

[2] Gibbons, G., Hansell, R., Norwood, A. and Dickens, P., 2003, "Rapid laminated die-cast tooling", Assembly Automation, Vol. 23(4), pp. 372-81.

[3] Dickens, P., 1995, "Research developments in rapid prototyping", Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Vol. 209, pp. 261-266.

[4] Ahari, H., 2011, "Optimization of Laminated Dies Manufacturing", PhD thesis, University of Waterloo.

[5] Xu, X., Sachs, E. and Allen, S., 2001, "The design of conformal cooling channels in injection molding tooling", Polymer Engineering and Science, Vol. 41(7), pp. 1265-1279.

[6] Laws, K., Gun, B. and Ferry, M., 2006, "Effect of die-casting parameters on the production of high quality Mg-base bulk metallic glass samples", Materials Science & Engineering A, Vol. 425(1-2), pp. 114-120.

[7] Bishenden, W. and Bhola, R., 1999, "Die temperature control", Transaction of the 20th International Die Casting Congress and Exposition, North American Die Casting Association, Cleveland, USA, pp. 161−164.

[8] Kong, L., She, F., Gao, W., Nahavandi, S. and Hodgson, P., 2008, "Integrated optimization system for high pressure die casting processes", Journal of Materials Processing Technology, Vol. 201(1-3), pp. 629-634.

[9] Gorbach, P. and Shutts, B., 2002, "The use of temperature control units in die casting dies", Die Casting Engineer, Vol. 46(6), pp. 50-55.

[10] Dubay, R., Pramujati, B. and Hernandez, J., 2005, "Cavity temperature control in injection plastic molding", Mechatronics and Automation, 2005 IEEE International Conference, Vol. 2, pp. 911-916.

[11] Yang, T., Chen, X. and Hu, H., 2007, "A fuzzy PID thermal control system for die casting processes", 22nd IEEE International Symposium on Intelligent Control, Singapore, pp. 389-394.

[12] Yang, T., Hu, H., Chen, X., Chu, Y. and Cheng, P., 2007, "Thermal analysis of casting dies with local temperature controller", The International Journal of Advanced Manufacturing Technology, Vol. 33(3-4), pp. 277-284.

[13] Gao, F., Patterson, W. and Kamal, M., 1993, "Dynamics and control of surface and mold temperature in injection molding", International Polymer Processing, Vol. 8, pp. 147-157.

[14] Vetter, R., Maijer, D., Huzmezan, M. and Meade, D., 2004, "Control of a die casting simulation using an industrial adaptive model-based predictive controller", Proceedings of the TMS Fall Extraction and Processing Conference, pp. 235-246.

[15] Dubay, R., 2002, "Self-optimizing MPC of melt temperature in injection moulding", ISA Transactions, Vol. 41(1), pp. 81-94.

[16] Tsai, C. and Lu, C., 1998, "Multivariable self-tuning temperature control for plastic injection molding process", IEEE Transactions on Industry Applications, Vol. 34(2), pp. 310-18.

[17] Lu, C. and Tsai, C., 2001, "Adaptive decoupling predictive temperature control for an extrusion barrel in a plastic injection molding process", IEEE Transactions on Industrial Electronics, Vol. 48(5), pp. 968-975.

[18] Karray, F. and De Silva, C., 2004, "Soft computing and intelligent systems design", Addison Wesley Publishing, Pearson Education, Chap. 6.

[19] Rai, J., Lajimi, A. and Xirouchakis, P., 2008, "An intelligent system for predicting HPDC process variables in interactive environment", Journal of Materials Processing Technology, Vol. 203(1-3), pp. 72-79.

[20] Yarlagadda, P., 2000, "Prediction of die casting process parameters by using an artificial neural network model for zinc alloys", International Journal of Production Research, Vol. 38(1), pp. 119-139.

[21] Krimpenis, A., Benardos, P., Vosniakos, G. and Koukouvitaki, A., 2006, "Simulation-based selection of optimum pressure die-casting process parameters using neural nets and genetic algorithms", International Journal of Advanced Manufacturing Technology, Vol. 27(5-6), pp. 509-517.

[22] Faessler, A. and Loher, M., 1996, "Quality control in die casting with neural networks, Neuro-Fuzzy Systems, International Symposium", Proceedings of the Symposium on Neuro-Fuzzy Systems, Lausanne, Switzerland, pp. 147-153.

[23] Lau, H., Wong, T. and Pun, K., 1999, "Neural-fuzzy modeling of plastic injection molding machine for intelligent control", Expert Systems with Applications, Vol. 17(1), pp. 33-43.

[24] Werbos, P., 1974, "Beyond regression: New tools for prediction and analysis in the behavioral sciences", PhD thesis, Harvard University.

[25] Rumelhart, D., Hinton, G. and Williams, R., 1986, "Learning representations by back-propagating errors", Nature, Vol. 323 (6088), pp. 533–536.

[26] Hagan, M. and Menhaj, M., 1994, "Training feedforward networks with the Marquardt algorithm", IEEE Transactions on Neural Networks, Vol. 5(6), pp. 989-993.

[27] Wilamowski, B., Iplikci, S., Kaynak, O. and Efe, M., 2001, "An algorithm for fast convergence in training neural networks", Proceedings of the International Joint Conference on Neural Networks, Vol. 3, pp. 1778-1782.

[28] Omatu, S., Maruzuki, K. and Rubiyah, Y., 1996, "Neuro-control and its applications", Springer, London, Chap. 4.

[29] Song, G., Longman, R. and Mukherjee, R., 1999, "Integrated sliding-mode adaptive-robust control", IEE Proceedings of Control Theory and Applications, Vol. 146(4), pp. 341-347.

[30] Abdelrahman, M., Tantawy, M. and Bayoumi, M. 2006, "Adaptive Neuro-Fuzzy Control Approach for Spacecraft Maneuvers", International Journal on Automatic Control and System Engineering, Vol. 6 (2), pp. 49-56.

[31] Wesley-Smith, I., 2006, "A parallel artificial neural network implementation", Proceedings of the National conference on graduate research (NCUR), University of North Carolina at Asheville.

[32] Psaltis, D., Sideris, A., and Yamamura, A., 1988, "A multi- layered neural network controller", IEEE Control Systems Magazine, Vol. 8, pp. 17-21.

[33] Kawato, M., Uno, Y., Isobe, M. and Suzuki, R., 1988, "Hierarchical neural network model for voluntary movement with application to robotics", IEEE Control Systems Magazine, Vol. 8 (2), pp. 8–16.

[34] Akhyar, S. and Omatu, S., 1992, "Neuromorphic self-tuning PID controller", Proceedings of IEEE International Conference on Neural Networks, pp. 552-557.

[35] Narendra, K. and Parthasarathy, K., 1990, "Identification and control of dynamical systems using neural networks", IEEE Transactions on Neural Network, Vol. 1(1), pp. 4-27.

[36] Demetriou, M., 2005, "Robust sensor location optimization in distributed parameter systems using functional observers", Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference, Seville, Spain, pp. 7187 – 7192.

[37] Udwadia, F., 1994, "Methodology for optimum sensor locations for parameter identification in dynamic systems", Journal of Engineering Mechanics, Vol. 120(2), pp. 368-390.

[38] Fadale, T., Nenarokomov, A. and Emery A., 1995, "Two approaches to optimal sensor locations", ASME Journal of Heat Transfer, Vol. 117(2), pp. 373-379.

[39] Papadimitriou, C., 2005, "Pareto optimal sensor locations for structural identification", Computer Methods in Applied Mechanics and Engineering, Vol. 194, pp. 1655-1673.

[40] Papadimitriou, C., Beck, J. and Au, S., 2000, "Entropy-based optimal sensor location for structural model updating", Journal of Vibration and Control, Vol. 6(5), pp. 781-800.

[41] Rao, A. and Anandakumar, G., 2007, "Optimal placement of sensors for structural system identification and health monitoring using a hybrid swarm intelligence technique", Smart Materials and Structures, Vol. 16, pp. 2658-2672.

[42] Yao, L., Sethares, W. and Kammer, D., 1993, "Sensor placement for on orbit modal identification via a genetic algorithm", AIAA Journal, Vol. 31(10), pp. 1167–1169.

[43] Dour, G., Dargusch, M. and Davidson, C., 2006, "Recommendations and guidelines for the performance of accurate heat transfer measurements in rapid forming processes", International Journal of Heat and Mass Transfer, Vol. 49(11-12), pp. 1773-1789.

[44] Hamasaiid, A., Dour, G., Dargusch, M., Loulou, T., Davidson, C. and Savage, G., 2008, "Heat-transfer coefficient and in-cavity pressure at the casting-die interface during high-pressure die casting of the magnesium alloy AZ91D", Physical Metallurgy and Materials Science, Vol. 39(4), pp. 853-864.

[45] Hu, H., Chen, F., Chen, X., Chu, Y. and Cheng, P., 2004, "Effect of cooling water flow rates on local temperatures and heat transfer of casting dies", Journal of Materials Processing Technology, Vol. 148, pp. 439-451.

[46] http://www.matweb.com

[47] http://www.santoprene.com.

[48] ANSYS CFX-Solver Modeling Guide, ANSYS CFX Release 11.0, December 2006.

[49] Hutton, D., 2004, "Fundamentals of finite element analysis", McGraw-Hill, New York, Chap 6.

[50] Hafner, M., Schüler, M., Nelles, O. and Isermann, R., 2000, "Fast neural networks for diesel engine control design", Control Engineering Practice, Vol. 8(11), pp.1211-1221.

[51] Zheng, C. and Bennett, G., 2002, "Applied contaminant transport modeling", Wiley-Interscience, New York, Chap. 12.

[52] Choi, K. and Kim, N., 2005, "Structural sensitivity analysis and optimization 1: Linear systems", Springer-Verlag, New York, Chap. 1.

[53] Chen, B., Tai, P., Harrison, R. and Pan, Y., 2005, "Novel hybrid hierarchical-k-means clustering method (h-k-means) for microarray analysis", Proceedings of 2005 IEEE Computational Systems Bioinformatics Conference, pp. 105-108.

[54] Jun, S. and Oh, I., 2007, "A co-evolutionary k-means algorithm", International Journal of Soft Computing, Vol. 2(5), pp. 624-627.

[55] Moore, C., 1992, "Selection of controlled and manipulated variables", Practical Distillation Control, Van Nostrand Reinold, New York, pp. 140-177.

[56] Jolliffe, I., 2002, "Principal component analysis, Series: Springer series in statistics", Springer, New York, Chap. 2.

[57] Zamprogna, E., Barolo, M. and Seborg, D., 2005, "Optimal selection of soft sensor inputs for batch distillation columns using principal component analysis", Journal of Process Control, Vol. 15(1), pp. 39-52.

[58] Wang, J. and Chen, Y., 2006, "A fully automated recurrent neural network for unknown dynamic system identification and control", IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Vol. 53(6), pp. 1363-1372.

[59] He, X. and Asada, H., 1993, "A new method for identifying orders of input–output models for nonlinear dynamic systems", Proceedings of the American Control Conference, San Francisco (CA), pp. 2520-2523.

[60] Yu, D., Gomm, J. and Yu, D., 2005, "A model order and time-delay selection method for mimo non-linear systems and its application to neural modeling", International Journal of Information and Systems Sciences, Vol. 1(1), pp. 39-60.

[61] Chiu, S., 1994, "Fuzzy model identification based on cluster estimation", Journal of Intelligent & Fuzzy Systems, Vol. 2 (3), pp. 267–278.

[62] Alata, M., Molhim, M. and Ramini, A., 2008, "Optimizing of fuzzy c-means clustering algorithm using GA", Proceedings of World Academy of Science on Engineering and Technology, Vol. 29 (5), pp. 224~229.

[63] Vernieuwe, H., Bernard, B. and Verhoest, N., 2006, "Comparison of clustering algorithms in the identification of takagi-sugeno models: A hydrological case study", Fuzzy Sets and Systems, Vol. 157 (21), pp. 2876-2896.

[64] Potsch, G. and Michaeli, W. 2007, "Injection molding: An introduction", Hanser/Gardner Publications, Chap. 4.

[65] Saifullah, A. and Masood, S., 2007, "Finite element thermal analysis of conformal cooling channels in injection moulding", Proceedings of the 5th Australasian Congress on Applied Mechanics, Brisbane, Australia, pp. 337-341.

[66] McLoone, S. and Irwin, G., 1997, "Fast parallel off-line training of multilayer perceptrons", IEEE Transactions on Neural Networks, Vol. 8(3), pp. 646-653.

[67] Sola, J. and Sevilla, J., 1997, "Importance of input data normalization for the application of neural networks to complex industrial problems", IEEE Trans. Nuclear Science, Vol. 44, pp. 1464-1468.

[68] Sha, D. and Bajic, V., 1999, "On-line adaptive learning rate BP algorithm for multi-layer feed-forward neural networks", Journal of Applied Computer Science, Vol. 7(2), pp. 67-82.

[69] Sha, D. and Bajic, V., 2002, "An on-line hybrid learning algorithm for multilayer perceptron in identification problems", Computers and Electrical Engineering, Vol. 28, pp. 587-598.

[70] Gong, C., Feng, Y., Wang, J. and Song, Y., 2007, "A fuzzy-PID control system of PTFE sintering furnace based on Lonworks", 26th China Control Conference, Vol. 4, pp. 266-269.

[71] Yang, T., Hu, H., Chen, X., Chu, Y., and Cheng, P., 2005, "An intelligent control system for die thermal management", Proceedings of Materials Science and Technology, Vol. 1, pp. 11-20.

[72] Mamdani, E., 1976, "Advances in the linguistic synthesis of fuzzy controllers", International Journal of Man-Machine Studies, Vol. 8, pp. 669–678.

[73] Guo, B., Liu, H., Luo, Z. and Wang, F., 2009, "Adaptive PID controller based on BP neural network", International Joint Conference on Artificial Intelligence, pp. 148-150.

[74] Wang, J., Zhang, C. and Jiang, Y., 2008, "Adaptive PID control with BP neural network self-tuning in exhaust temperature of micro gas turbine", 3rd IEEE Conference on Industrial Electronics and Applications, pp. 532-537.

[75] Zhang, M., Wang, X. and Liu, M., 2005, "Adaptive PID control based on RBF neural network identification", 17th IEEE International Conference on Tools with Artificial Intelligence, pp. 681-683.

[76] Jiang, J., Wen, S., Zhou, Z. and He, H., 2008, "Fuzzy barrel temperature PID controller based on neural network", 2008 International Congress on Image and Signal Processing (CISP), Vol. 1, pp. 90-94.

[77] Kajan, S., 2008, "Neural controllers for nonlinear systems in MATLAB", 16th Annual Conference Technical Computing, Prague, Czech Republic, ISBN 978-80-7080-692-0.

[78] Dias, F. M., Antunes, A. and Mota, A., 2005, "Additive internal model control: A new control strategy", International Transactions on Computer Science and Engineering, Vol. 25, pp. 1-12.

[79] Sørensen, O., 1994, "Neural networks in control applications", PhD Thesis, Department of Control Engineering, Institute of Electronic Systems, Aalborg University, Denmark.