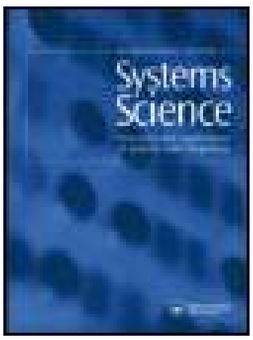


International Journal of Systems Science

ISSN: 0020-7721 (Print) 1464-5319 (Online) Journal homepage: http://www.tandfonline.com/loi/tsys20

On identification of Hammerstein and Wiener model with application to virtualised software system

Dharma Aryani, Liuping Wang & Tharindu Patikirikorala

To cite this article: Dharma Aryani, Liuping Wang & Tharindu Patikirikorala (2016): On identification of Hammerstein and Wiener model with application to virtualised software system, International Journal of Systems Science, DOI: 10.1080/00207721.2016.1244303

To link to this article: http://dx.doi.org/10.1080/00207721.2016.1244303

Published online: 17 Nov 2016.

INTERNATIONAL JOURNAL OF SYSTEMS SCIENCE, 2016
http://dx.doi.org/10.1080/00207721.2016.1244303

Taylor & Francis
Taylor & Francis Group

On identification of Hammerstein and Wiener model with application to virtualised software system

Dharma Aryani[a,b], Liuping Wang[a] and Tharindu Patikirikorala[c]

[a]School of Electrical and Computer Engineering, RMIT University, Melbourne, VIC, Australia; [b] Department of Electrical Engineering, The State Polytechnic of Ujung Pandang, Makassar, Indonesia; [c]Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, VIC, Australia

ABSTRACT

This paper proposes a system identification method for estimating virtualised software system dynamics within the framework of a Hammerstein–Wiener model. Building on the authors' previous work in identification and control of the software systems, the approach utilises frequency sampling filter structure to describe the linear dynamics and B-spline curve functions for the inverse static output nonlinearity. Furthermore, the issue on parameter selection for B-spline model approximation of scatter data is addressed by using a data clustering method. An experimental test-bed of virtualised software system is established to generate real observational data which are used to confirm the performance of the proposed approach. The identification results have shown that the model efficacy is increased with the proposed approach because the dimension of the nonlinear model can be significantly reduced while maintaining the desired accuracy.

ARTICLE HISTORY
Received 20 April 2015
Accepted 28 September 2016

KEYWORDS
Nonlinear system identification; virtualised software system; B-spline estimation; Hammerstein–Wiener; clustering analysis; frequency sampling filter

1. Introduction

In the embedded system and enterprise domains, software systems are the key instrument to maintain more efficient and dynamic interactions across computing infrastructures. The nature of software systems has emerged into a pattern of utility service models because a shared resources environment is maintained in the system level to provide services for multiple users (Andrzejak, Arlitt, & Rolia, 2002; Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009; Weiss, 2007). Accordingly, useful access is given for end users to utilise the consolidated and configurable computing resources. Extensive developments in software system evoke the increasing popularity of virtualisation technology as reliable framework for a high performance computing generation. Virtualisation is a technique to simplify the computing environment by means of partitioning hardware platforms into individual units called VMs (Barham et al., 2003). Each VM is independent and allowed to run on different operating systems or to use multiple applications, where each of them is a duplication of the real system. As a result, software systems could be constructed into agile and cost-efficient structures, and their reliability would increase because their performance quality and the complex resources management could be optimised (Armbrust et al., 2009). Moreover, virtualisation offers operational benefits for better resources utilisation and performance isolation while providing features for resource provisioning alternatives (Azeez et al., 2010). These benefits allow the resources to be dynamically allocated among the VMs. Over many approaches for resource provisioning, proportional sharing is one of the suitable ways for resource allocation (Chenyang, Abdelzaber, Stankovic, & Son, 2001; Patikirikorala, Wang, Colman, & Han, 2011a, 2014). Its provisioning mechanism is performed by specifying certain ratio of shares between VMs during runtime with regard to the targeted performance objectives or specific preferences for Quality of Services (QoS) properties.

For years, there has been a growing attention on the research about software system management. Hellerstein (2004), Karamanolis, Karlsson, and Zhu (2005), and Patikirikorala, Colman, Han, and Wang (2012a) highlighted the importance of model estimation and control system approach to performance management and monitoring in software systems. A number of studies have emphasised control system applications for relative performance differentiation scheme, in which the resources provisioning for each customer is manipulated to support different classes or priority levels between customers. This scheme has been investigated in the works by Lu, Abdelzaher, Lu, and Tao (2002), Pan, Mu, Wu, and Yao (2008), Padala, Hou, Shin, and Zhu (1996), Patikirikorala et al. (2011a), Patikirikorala, Wang, Colman, and Han (2011b, 2012b), Patikirikorala, Wang, and Colman (2011c) and Patikirikorala et al. (2014), implementing feedback control to automate the resource and performance management. In applying such dynamic control systems, the main objective is to manipulate the system management with respect to the reference conditions, where input variables are adjusted until the measured outputs converge to the desired values in the presence of disturbances during the runtime. Whereas, it has been analyzed by Lu, Abdelzaher, Stankovic, and Son (2001) and from a survey by Patikirikorala, Colman, Han, and Wang (2012a) that in an application of relative guarantee management, nonlinearities exist in the relationship of input and output parameters.

In order to employ control engineering techniques in virtualised software systems, their characteristics including both linear and nonlinear dynamics should be identified. However,

CONTACT Dharma Aryani s3351269@student.rmit.edu.au

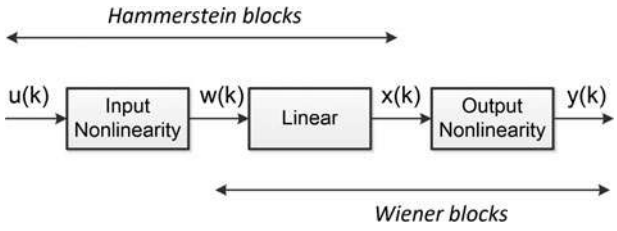


Figure 1. Hammerstein–Wiener structure.

the majority of the existing results has not considered the nonlinear dynamics of of shared resources systems. Researchers have demonstrated studies for automated management of resources and QoS performance using linear models (Karlsson, Zhu, & Karamanolis, 2005; Kusic & Kandasamy, 2006; Lu, Abdelzaher, Lu, Sha, & Liu, 2003; Lu, Saxena, & Abdelzaher, 2001; Lu et al., 2002; Padala et al., 2007; Pan et al., 2008; Wu, Lilja, & Bai, 2005). The argument behind the decision to use linear models only was the common assumption that linear models can be estimated with less computational efforts and they are often considered sufficient to be used as the basis for controller design in an operating region. However, further studies by Zhu, Wang, and Singhal (2006), Padala, Hou, Shin, and Zhu (2010) and Patikirikorala et al. (2011a) have observed the nonlinear relationship between resources provisioning and performance properties. Ignoring the nonlinear characteristics of a software system may degrade the management quality because its overall behaviour is not captured as a whole. To overcome these limitations, nonlinear system identification is proposed for two reasons: (1) to obtain proper models to characterise both the static and dynamic input–output relationships, and (2) to identify the nonlinear properties of the models in a wide range of operating points.

Billings and Fakhouri (1979, 1982) asserted that system identification of nonlinear systems can be characterised by synthesising linear and static nonlinear elements. Their original idea has evolved into Hammerstein–Wiener model estimation which becomes a favourable and generic approach for nonlinear dynamical model estimation (Ljung, 1999). In Hammerstein–Wiener, the nonlinear dynamic components (as shown in Figure 1) could be modelled in its inversion form Pajunen (1992), Kalafatis, Wang, and Cluett (1997), and Hong and Mitchell (2006). The purpose of capturing nonlinear dynamic in terms of its inversion is for easy compensation to the nonlinearity dynamics (Bloemen, Van den Boom, & Verbruggen, 2000; Fruzzetti, Palazoglu, & McDonald, 1997; Hong, Mitchell & Chen, 2012). In a recent research by Patikirikorala et al. (2011a, 2014), dynamics of shared resources environment are identified in the structure of Hammerstein–Wiener models. However, the estimation was implemented in a two-steps procedure which required extra efforts and a complex estimation.

This paper aims to improve system identification for linear and nonlinear dynamics of virtualised software systems. Building on the authors' previous work in identification and control, the improved approach utilises frequency sampling filter (FSF) structure to describe the linear dynamics and B-spline curve functions for the inverse static output nonlinearity. Furthermore, the issue on parameter selection for B-spline model

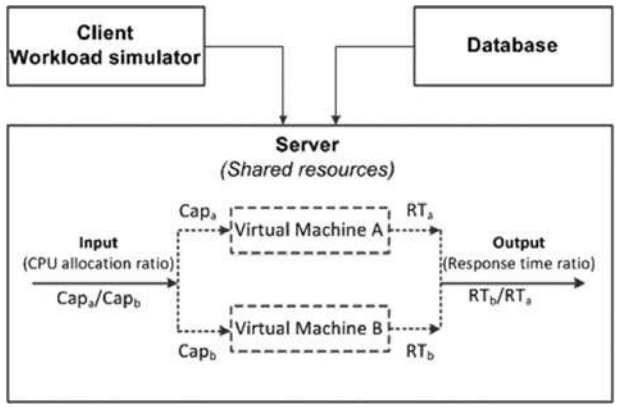


Figure 2. Virtualised software system.

approximation of scatter data is addressed by using a data clustering method. Another advancement in the current work is the use of K-means clustering method as an algorithm for curve parameters selection which provides more precise knot points position. With the proposed approach, the estimation of linear and nonlinear model parameters is performed in a straightforward manner. A new experimental test-bed of virtualised software system is established to generate real observational data which are used to confirm the performance of the proposed approach.

The structure of the paper is organised as follows. Section 2 covers the process description of virtualised software system test-bed. Section 3 addresses the characteristics of relative performance management. Section 4 formulates the proposed identification algorithms. The identification results using experimental data from the test-bed are demonstrated in Section 5. Finally, the results are concluded in Section 6.

2. Process description

It is known that the environment of software system is complex and lack of first principal or prior knowledge about the real physical process (Padala et al., 2010). For experimental purpose, an architecture of virtualised software system is built on the scenario of multitier applications by implementing RUBiS. It is an online auction site benchmark in three tiers application of e-commerce website which models the behaviour of ebay.com. RUBiS has been used in several existing studies about software system management (e.g. Padala et al., 2007; Patikirikorala et al., 2014). In practice, a common pool of server resources is shared and each tier is hosted in a virtual machine (VM). This type of shared resources environment has been engaged to embedded system and business domains due to the efficiency of infrastructure utilisation and maintenance cost in data centre.

The test-bed, as shown in Figure 2, consists of a server, a database and a client workload simulator machine connected in an isolated network. Server is the shared infrastructure for two VMs which will utilise the resources proportionally. Therefore, the server is functioning as the host machine while the two VMs are the guest machines. For virtualisation, a software layer called 'hypervisor' is implemented to create VM by virtually partitioning the host machine hardware and to manage guests operating systems. In this study, the hypervisor is *Xen*2.6, an open

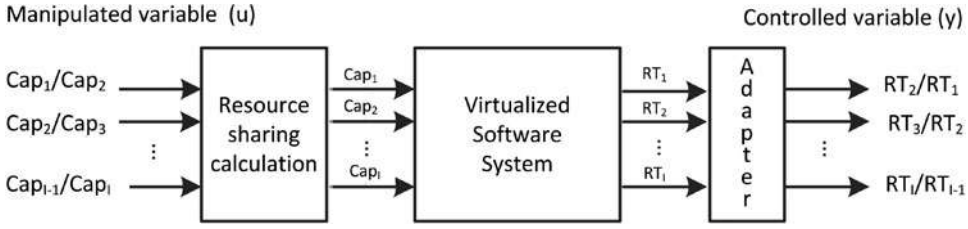


Figure 3. Structure of relative performance management scheme.

source native hypervisor. This hypervisor comes with a credit-based scheduler function that gives advantage for resource allocation. The scheduler dynamically allocates a certain share of CPU capacity to each VM. In order to adjust this function, an actuator was installed to feed the desired ratio of CPU to the system. Since the system sending intensive client requests, one CPU is dedicated only for the management of VM (*dom*0) while all VMs are pinned to one CPU. This setting provides an efficiency for CPU capacity provisioning among VMs. In addition, a sensor program was attached to each VM to calculate the response time of incoming requests. Also, Apache Httpd 2.2 server is installed in all machines for customer application settings. The host and guest machines are running on *CentOS* (Community Enterprise operating system). RUBiS workload generator is set up to simulate the workloads for each VM and the database of RUBiS benchmark was deployed in the other machine.

3. Characteristics of relative performance management

Dealing with a shared resources environment to achieve proper functional services in runtime is a very complex task (Padala et al., 2007; Patikirikorala et al., 2011a). The performance objective of the service provider is to maintain response time on the reference levels or subject to the priority degree of the clients/users. In order to achieve this objective, the available resources need to be manipulated to give equivalent responses for the incoming workloads from the clients. For such systems, the performance stability is affected by three conditions. First, unpredictable conditions or the change of system dynamics may occur in the operating stages, including demand or workload changes, which are most likely to be disturbances to the system. Second, complex preferences for performance objective require complicated procedures to manage the QoS metrics of a software system, such as response time. Third, resource demands between each tier or consumer might be different and correlated to each other. Thus, the pool of shared resources needs to be attained dynamically by adjusting the resources according to a client requirement while acknowledging the requirement of other users. These issues can be addressed efficiently on the basis of dynamic resources management. However, the effectiveness of dynamic resources management between multiple users in a shared environment is mainly determined by the accuracy of the model prediction. In a resources management, the average response time for each client is maintained under varying workloads based on absolute or relative guarantee scheme. The key point that distinguishes a relative scheme from a absolute scheme is the existence of relative relationship between the clients. This relative relationship can be captured by their ratio value for convenience of analysis. From control system point of view, this means that a single input and single output system can be configured for a case of two clients scenario. In the same spirit, Lu et al. (2003) and Patikirikorala et al. (2011a) confirmed that relative performance management scheme is very useful for service differentiation on overloaded systems.

A relative management scheme specifies the relative importance of clients and controls the ratio of QoS parameters to the desired levels. Each consecutive client is paired and the ratios are computed between the pairs. In this paper, the CPU capacity is allocated to dynamically manage the response time as the controlled variable of QoS parameter. Figure 3 illustrates the input and output variables in the framework of a relative management objectives for I client classes.

3.1. Input and output variables

The input and output data pairs for system identification are generated from the test-bed with an experiment of two VMs (VM_a and VM_b). To represent relative management scheme, input and output variables are defined as the ratio values of the related variables from two VMs. The input parameter is resource allocation ratio representing the CPU capacity entitlement for VM_a over the entitlement for VM_b while the output is the response time ratio from the time measurement of each VMs to respond to the workload requests. These parameters are the main metrics which indicate the end-users experience. This selection of input–output variables has been confirmed in a survey study by Patikirikorala, Colman, Han, and Wang (2012a) for being widely used for software system modelling. In addition, when adaptive control approaches will be implemented, the input should be persistently excited to guarantee the parameters convergence to their true value (Astrom & Wittenmark, 1995; Ioannou & Sun, 1996; Ljung, 1999).

The characteristics of input and output variables of this test-bed are depicted in Figure 4. It shows the relationship between CPU allocation and response time to confirm the presence of nonlinear dynamics in virtualised software system. Furthermore, the incoming workloads changes affect the performance of system dynamics, so that these will be considered as disturbances to the system on further study for control system experimentation.

In a two VMs scenario, when VM_b requires higher priority than VM_a, the response time of VM_b needs to be maintained faster than the response time of VM_a. In order to achieve this objective, the CPU capacity allocation for VM_b should be larger than the allocation for VM_a. It is considered that at least

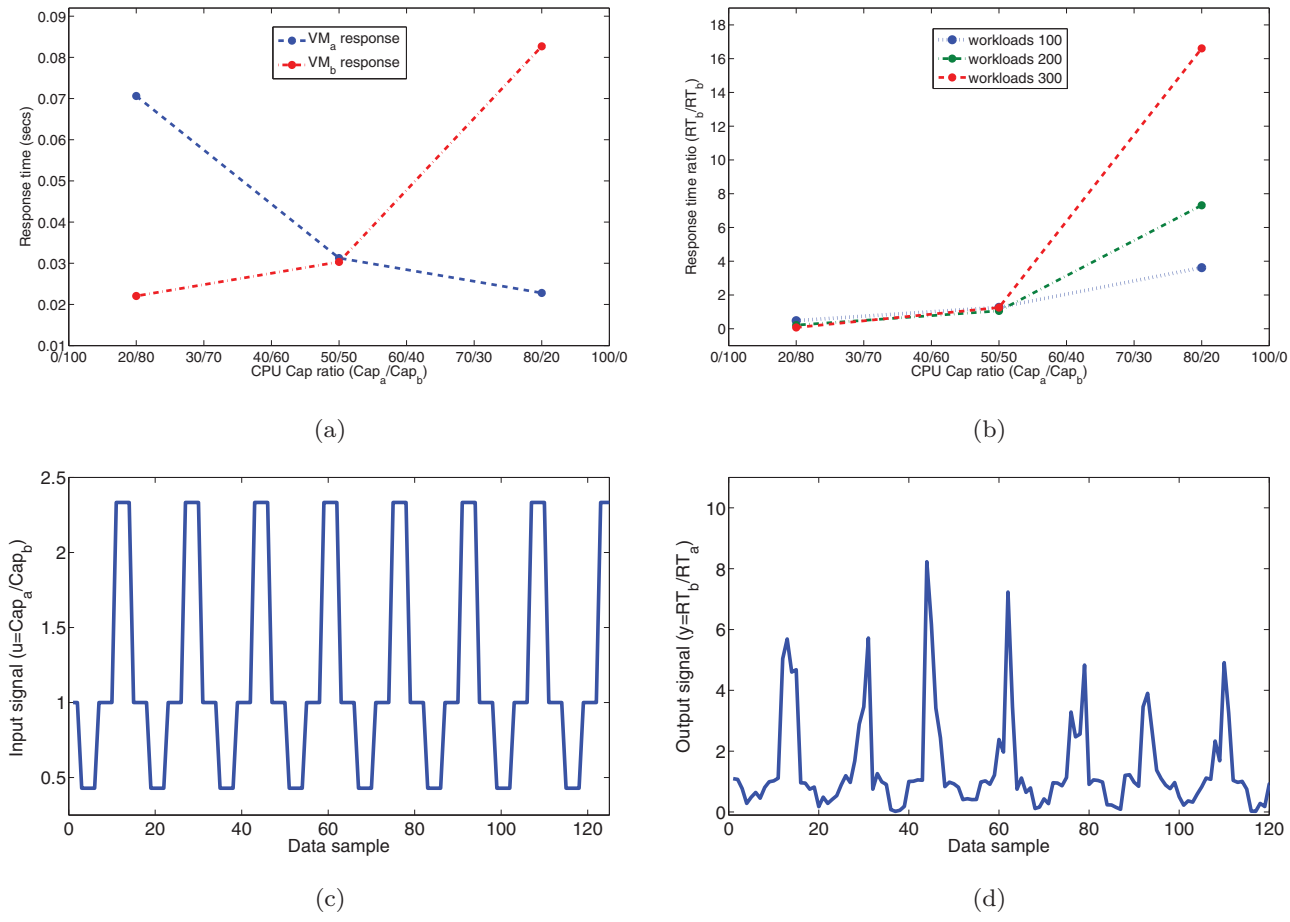


Figure 4. Characteristic of virtualised software systems testbed, (a) Average response time of each *VM* for workload constant 100, (b) Average response time ratio of the *VM*s for different workloads, (c) and (d) Input and Output signal for constant workloads 100.

one request to a VM arrives from client in each sampling time. $Cap_a(k)$, $Cap_b(k)$ are denoted as the CPU allocations VM_a and VM_b respectively that make the total CPU capacity $Cap_{\text{total}} = Cap_a + Cap_b$. It yields the input equation to

$$u(k) = \frac{Cap_a(k)}{Cap_b(k)} \tag{1}$$

The portion of resource sharing is in the percentage of total CPU capacity where full CPU capacity equals to 100%. Therefore, to prevent a shortage of resources when workload requests suddenly increase in unpredictable conditions, CPU share is constrained to a minimum capacity. In this experiment, $Cap_{a,\,\min}(k)$, $Cap_{b,\,\min}(k) = 20$. So, it will ensure that a certain share of CPU capacity is assigned for each VM during runtime. On output side, measured response time is the output of this virtualised system which is expressed in $y(k) = \frac{\mathrm{RT}_a(k)}{\mathrm{RT}_b(k)}$. $\mathrm{RT}_a(k)$, $\mathrm{RT}_b(k)$ represent response time to the workloads from VM_a and VM_b, respectively. When the amount of CPU allocation for a workload increases, the average response time decreases due to more resources been provisioned to deal with the client's request. Consequently, variables $\frac{Cap_a(k)}{Cap_b(k)}$ and $\frac{\mathrm{RT}_a(k)}{\mathrm{RT}_b(k)}$ are inversely proportional to each other which lead to the configuration of a compatible input and output structure where the output is modified to

$$y(k) = \frac{\mathrm{RT}_b(k)}{\mathrm{RT}_a(k)} \tag{2}$$

3.2. Dynamic nonlinearities

Nonlinearities in virtualised software systems have been characterised by Patikirikorala et al. (2011a) which is the reference of this study to formulate dynamic nonlinearities in the testbed. Nonlinear behaviours exist on input and output variables. Nonlinearities in input variable are caused by the relationship between variable $Cap_a((k))$ and $Cap_b((k))$ in the form of their ratio as in Equation (1) and the minimum bound for the input signal $Cap_{\min}$. The input signal is formulated as

$$\frac{Cap_a}{Cap_b} = \frac{Cap_{\text{total}} - Cap_b}{Cap_b} = \frac{Cap_{\text{total}}}{Cap_b} - 1 \tag{3}$$

This equation leads to static restriction of the operating points in input variable. For instance, if $Cap_{\text{total}} = 100$ and $Cap_{\min} = 20$, then the possible operating points configuration will be $n = 61$ points $(u = u_1, u_2, u_3, \ldots, u_{n-1}, u_n)$ with the following sequence:

$$u = \frac{20}{80}, \frac{21}{79}, \ldots, \frac{50}{50}, \ldots, \frac{79}{21}, \frac{80}{20}$$

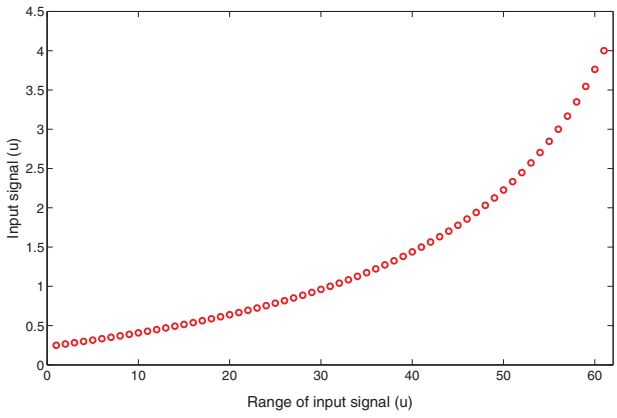


Figure 5. Description of input nonlinearity.

The order of these points is nonsymmetrical because the deviations between these points are unequal. Therefore, if these points are plotted, it is clearly shown that the range of the u points is nonlinear, depicted in Figure 5.

Likewise in output dynamics, the nonlinearity issue exists in the same way. Yet, the points could not be predefined because the variable is response time which has to be measured during runtime. The nonlinear characteristic in output led by the output signal function in Equation (2) where the response time value of VM is used in a division operation. If workload variations for VM_a and VM_b have large difference, the $RT_a(k)$ and $RT_b(k)$ will also have large a range of time difference. Therefore, $y(k) = \frac{RT_b(k)}{RT_a(k)}$ forms disproportionate values. When response time VM_a is bigger than response time VM_b ($RT_a(k) > RT_b(k)$), the output increases to high value. But when ($RT_a(k) < RT_b(k)$), the $y(k)$ value decreases in higher rate as well. For example, a certain ratio of input applied to the system and the measured response time value for $RT_a(k) = 0.01$(sec) and $RT_b(k) = 0.2$(sec), then by using Equation (2), returns $y(k) = 20$. However, if the input ratio is in reverse, the response time also changes to $RT_a(k) = 0.2$(sec) and $RT_b(k) = 0.1$(sec), the $y(k)$ will be 0.05.

4. Identification algorithm

Hammerstein–Wiener structure is a general scheme of nonlinear models which has been used comprehensively in nonlinear system identification. Parameter estimation of dynamic systems in Hammerstein–Wiener model is comprised of a form of block-oriented model that consists of a linear model and nonlinear blocks on input and output elements, as illustrated in Figure 1. The static nonlinearities are modelled in nonlinear blocks and the rest of system dynamics are captured in the linear block. Wide range of applications has chosen Hammerstein–Wiener approach to approximate dynamic models because the method profoundly presents good performance in system identification for process with significant nonlinearity issues (e.g. Hong, Gong, & Chen, 2011; Jurado, 2006; Kalafatis, Arifin, Wang, & Cluett, 1995; Zhu, 1999). This section addresses the system identification algorithm of virtualised software system in Hammerstein–Wiener structure with a straightforward estimation for the Wiener model.

For identification purposes, two basic assumptions are made to analyse the behaviour of virtualised software system. First, the dynamic linear element is stable and second the static nonlinearity is continuous and single-valued in the range of input–output data. According to Figure 1, the measured variables are input $u(k)$ and output $y(k)$, while $w(k)$ and $x(k)$ are denoted as the intermediate variables representing unmeasurable input and output of linear element. In Hammerstein block, the nonlinear dynamic of input signal is captured in the form of its inversion. In Wiener blocks, the linear element is formulated in FSF function and the nonlinear element is in inverse static nonlinearity of the output signal which is modelled in B-spline curve terms. This approach will be advantageous for control system design because by integrating the inverse static nonlinearities in input and output elements, the presence of nonlinear dynamics is sufficiently eliminated and the control system can be designed as a linear system (Patikirikorala et al., 2011a).

4.1. Hammerstein block

One of the earliest applications of Hammerstein model identification was introduced by Billings and Fakhouri (1979) using cross-correlation techniques to separate the linear element from the nonlinear dynamics. To date, researcher used extended approaches to approximate the nonlinear models, i.e. asymptotic method (Zhu, 2000), blind approach (Bai & Fu, 2002) and B-spline neural network (Hong et al., 2012). Nonlinear model in Hammerstein block is assigned to get the relationship between input signal u and intermediate input w in the form of inverse static nonlinearity function. The estimated model will be employed as compensator for the nonlinear characteristic in input element. As discussed in Section 3.3, nonlinear issue occurs in input element because of uneven deviation of the operating points set. Patikirikorala et al. (2011a) have proposed a simple technique to transform the operating points to equally spaced operating points. This approach is adopted to estimate the inverse static input nonlinearity in this paper. This technique was formulated by defining an intermediate variable w where $w_{\min} \le w \le w_{\max}$. To have an equally spaced operating points, a fixed deviation is determined by $\delta w = \frac{w_{\max} - w_{\min}}{n-1}$. Therefore, the range of operating points for intermediate variable w is selected as a sequence of $w = w_1, w_2, w_3, \cdots, w_n$, where $w_1 = w_{\min}$, $w_2 = w_1 + \delta w$, $w_3 = w_2 + \delta w$ and $w_n = w_{\max}$. Then, the respective operating points of input u are mapped to the w points. From system identification point of view, the intermediate variable w becomes the input signal to the Wiener system in the next stage of system identification. However, for the future control applications, it is advantageous to find the relationship between input signal u and intermediate variable w. The relationship between data pairs from the mapping is approximated in inverse static input nonlinear function ($u = f^{-1}(w)$). This inverse function is expressed in polynomial form

$$u(k) = \beta_0 + \beta_1 w(k) + \beta_2 w(k)^2 + \cdots + \beta_m w(k)^m = \phi_{ni}(k)\theta_{ni} \tag{4}$$

where the coefficient parameter vector $\theta_{ni} = [\,\beta_0\ \beta_1\ \cdots\ \beta_m\,]^T$ and data vector $\phi_{ni}(k) = [\,1\ w(k)\ \cdots\ w(k)^m\,]$. The model coefficients

are estimated by least squares method:

$$\hat{\theta}_{ni} = \left(\sum_{k=1}^{n} \phi_{ni}(k)^T \phi_{ni}(k)\right)^{-1} \sum_{k=1}^{n} \phi_{ni}(k)^T u(k) \tag{5}$$

4.2. Wiener block

Different submodel structures have been broadly studied in the approximation of Wiener system. Numerous schemes have been implemented in both parametric model (Bai, 1998; Kalafatis et al., 1995, 1997) and non-parametric model (Greblicki, 1992). Pajunen (1992) suggested to use transfer function for the linear system and B-spline for the nonlinearity. Further study in Voros (1995) used transfer function for linear system and estimated the nonlinearity in polynomial form. Another approach by Kalafatis et al. (1997) used FSF terms for linear model and power series for the nonlinear element. In Hughes and Westwick (2005), Hong et al. (2012), Hong, Mitchell, and Chen (2013) and Zhu (1999), B-spline curve function has been utilised to identify the nonlinearity of a system dynamic in Wiener modelling.

Considering the block structure shown in Figure 1, Wiener block is composed of linear model followed by nonlinear model. The linear model is represented in FSF model. Moreover, the identification of nonlinear model will be delivered in term of inverse static nonlinearities by assigning B-spline curve function as the predicted nonlinear model.

4.2.1. Frequency sampling filter

FSF model is a frequency domain model which is established from the transformation of a time domain finite impulse response (FIR) model. It is first introduced by Bitmead and Anderson (1981) in the area of filter design. Afterwards, the extensive implementations of FSF model for system identification are successfully developed by Wang and Cluett (1994, 1997). These existing works confirmed that an FSF model can represent the model in fewer parameters than an FIR model. In addition, the estimation is unbiased and reliable to reduce noise effect, for an input only model. FSF model estimation only requires prior knowledge about settling time of the process with an assumption that the system is stable, linear and time-invariant process. The basic idea of FSF model is summarised below for the completion of this paper and details are referred to Wang and Cluett (2000). Transfer function of FIR is given by

$$G(z) = \sum_{i=0}^{M-1} h_i z^{-i} \tag{6}$$

where M is filter order and the impulse response h_i is defined for $0 \le i \le M-1$. Assuming M as an odd integer number, the discrete frequency response of the process can be corresponded to its impulse response coefficient as

$$h_i = \frac{1}{M} \sum_{l=-\frac{M-1}{2}}^{\frac{M-1}{2}} G(e^{j\frac{2\pi l}{M}}) e^{j\frac{2\pi li}{M}} \tag{7}$$

Substituting Equation (7) to Equation (6) produces

$$G(z) = \sum_{i=0}^{M-1} \frac{1}{M} \sum_{l=-\frac{M-1}{2}}^{\frac{M-1}{2}} G(e^{j\frac{2\pi l}{M}}) e^{j\frac{2\pi li}{M}} z^{-i} \tag{8}$$

By rearranging positions of elements in Equation (8), another formulation can be extracted as

$$\sum_{i=0}^{M-1} e^{j\frac{2\pi li}{M}} z^{-i} = \frac{1 - z^{-M}}{1 - e^{j\frac{2\pi l}{M}} z^{-1}} \tag{9}$$

which leads to the transfer function of FSF model:

$$G(z) = \sum_{l=-\frac{M-1}{2}}^{\frac{M-1}{2}} G(e^{j\frac{2\pi l}{M}}) \frac{1}{M} \frac{1 - z^{-M}}{1 - e^{j\frac{2\pi l}{M}} z^{-1}} \tag{10}$$

Equation (10) can be simply written as

$$G(z) = \sum_{l=-\frac{M-1}{2}}^{\frac{M-1}{2}} G(e^{j\omega_l}) H_l(z) \tag{11}$$

$$H_l(z) = \frac{1}{M} \frac{1 - z^{-M}}{1 - e^{j\omega_l} z^{-1}} \tag{12}$$

Following the finding of Wang and Cluett (1997), the FSF expression can be formulated in a reduced mth-order FSF model which is written as follows:

$$G(z) = \sum_{l=-\frac{m-1}{2}}^{\frac{m-1}{2}} G(e^{j\omega_l}) H_l(z) \tag{13}$$

Equation (13) is the reduced structure with $\omega_l = \frac{2\pi l}{M}$ as the centre frequency in $l = 0, \pm 1, \pm 2, \cdots, \pm\frac{m-1}{2}$. m is the effective order which indicates the significant parameters of FSF model and M represents the order of individual FSF filters corresponding to the process settling time $M = T_s/\Delta t$ and Δt is a sampling interval. m is odd number and much smaller than M. In FSF model estimation, the coefficients of regressor vector are generated by passing the input signal of the system through a set of parallel structured narrow band-limited FSFs. Then, it is weighted by the discrete frequency response corresponding to the centre frequency. The estimated output is the noise-free process output which is yielded from the summation of weighted filter outputs.

In relation to the system identification of linear model for Hammerstein–Wiener structure in Figure 1, the input and output variables for linear model are $w(k)$ and $x(k)$, respectively. The structure of FSF model estimation is described in Figure 6 and the model formulation is given below:

$$x(k) = G(z)w(k) \tag{14}$$

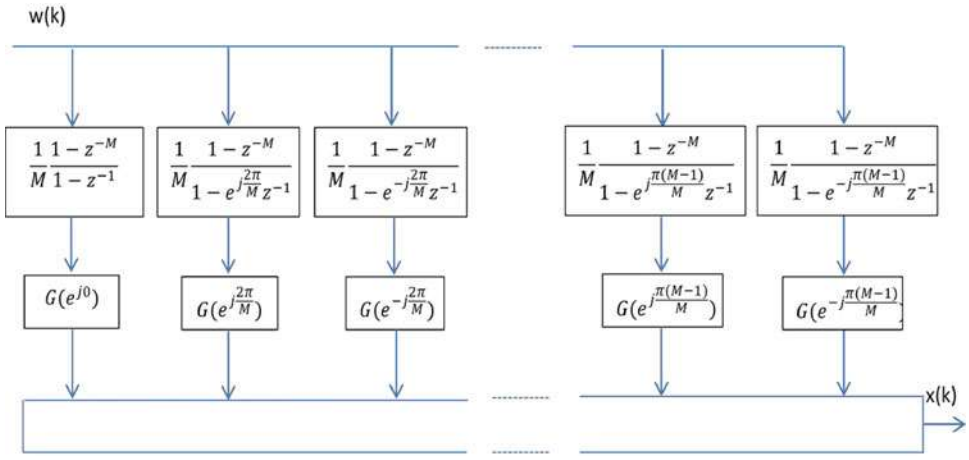


Figure 6. Diagram of frequency sampling filters (Wang & Cluett, 2000).

Referring to Equation (13), the coefficients of FSF model can be approximated as

$$x(k) = \sum_{l=-\frac{m-1}{2}}^{\frac{m-1}{2}} G(e^{j\omega_l}) f_l(k) \tag{15}$$

where $f_l(k) = H_l(z)w(k)$.

4.2.2. B-spline

The estimation of B-spline function is motivated by the initial finding of DeBoor (1978) which is widely known as 'de-Boor' algorithm. B-spline curve estimation has been very well established in computer aided geometry design (Farin, 1994; Piegl & Tiller, 1987). It is a mathematical model which is commonly applied to create and illustrate curves and surfaces for computer graphics. B-spline is the most famous type of spline functions to represent curve. Its signature is a B-spline function with order d constructed by joining several piecewise polynomials of $d-1$ degree (p) and parameterised by a knot vector which contains a set of points selected from the curve to break it into segments (Piegl & Tiller, 1995). The first stage in a B-spline approximation is selecting a set of parameters along data points range. These points will be used to calculate the knot vector intervals (knot spans). The intervals will direct the control points to form the curve. Therefore, this step determines the shape of the curve because inappropriate selection of the parameters could lead to an unpredictable curve estimation (Hong et al., 2013). There are two basic ways to select these parameters, uniform and nonuniform parameters. It is uniform if they are equally spaced, otherwise, it is called nonuniform. For nonlinear identification in this study, nonuniform parameters are implemented because it provides flexibility for mapping parameters onto curve by changing the knot spans length to accommodate the curve space. In conventional practice, the number and position of knots are predetermined to produce a model as accurate as possible with less control points. However, the accuracy of model prediction is sensitive to the location of knots especially when severe local nonlinearities exist.

In B-spline form, a curve is represented by combining control points and the basis functions. A parametric B-spline curve $B(q)$ is defined as

$$B(q) = \sum_{i=0}^{s} N_{i,p}(q) P_i \tag{16}$$

where $N_{i,p}(q)$ are the normalised pth-degree basis function of B-spline defined on a knot vector $Q = \{Q_0, Q_1, \ldots, Q_{s+p}, Q_{s+p+1}\}$ and P_i $(i = 0, \ldots, s)$ are the control points.

Knot vector can be constructed in two forms, periodic and nonperiodic (Piegl & Tiller, 1995). In this study, nonperiodic knot vector is used which is characterised by the multiplicity at the first and the last knots. Suppose $s+1$ parameters are selected from the observational data set ($T = \{t_0, t_1, \ldots, t_s\}$) and the B-spline degree is p, there will be $r+1$ knots to be used in the curve modelling, where $r = s+p+1$. The first $p+1$ knots are equal to the minimum parameter while the last $p+1$ knots are similar with the maximum parameter. In brief, to define a B-spline curve in degree p, knot vector $Q = \{Q_0, Q_1, \ldots, Q_{s+p}, Q_{s+p+1}\}$ is computed using the formula:

$$Q_0 = Q_1 = \ldots = Q_p = t_{\min} \tag{17a}$$

$$Q_{j+p} = \frac{1}{p} \sum_{i=j}^{j+p-1} t_i \quad \text{for } j = 1, 2, \ldots, s-p \tag{17b}$$

$$Q_{r-p} = Q_{r-p+1} = \ldots = Q_r = t_{\max} \tag{17c}$$

Basis function values for B-spline curve with degree p are calculated using the following equations:

$$N_{i,0}(q) = \begin{cases} 1 & \text{if } Q_i \le q < Q_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$N_{i,p}(q) = \frac{q - Q_i}{Q_{i+p} - Q_i} N_{i,p-1}(q) + \frac{Q_{i+p+1} - q}{Q_{i+p+1} - Q_{i+1}} N_{i+1,p-1}(q) \tag{19}$$

$N_{i,p}$ is a composite curve of degree p polynomial with joining points in knot span $[Q_i, Q_{i+p+1}]$. Therefore, before nonzero basis functions for q value are calculated based on Equations (18) and (19), the knot spans where q value lies should be known.

In association with nonlinear system identification for Wiener model as structured in Figure 1, the nonlinear characteristic will be estimated in the form of inverse static nonlinear model. Variable input of the model is $y(k)$ and the output is $x(k)$ which is the intermediate output variable. Thus, the representative formulation of B-spline curve $B(y(k))$ is in $x(k)$ which makes Equation (16) become

$$x(k) = \sum_{i=0}^{s} N_{i,p}(y(k))P_i \tag{20}$$

Subsequently, to compute a point of B-spline curve at fixed input values, the nonzero basis functions are multiplied with the corresponding control points.

$$x(k) = N_{0,p}(y(k))P_0 + N_{1,p}(y(k))P_1 + \ldots + N_{s,p}(y(k))P_s \tag{21}$$

4.2.3. Data clustering

Clustering is a process to divide a set of data points into non-overlapping groups using information from the data which reflect the relationship between all points (Anderberg, 1973; Kaufman & Rousseeuw, 1990). Data clustering is also known as cluster analysis which is very popular in scientific applications that involve analysis for structures with multivariate data. The aim of data clustering is to determine the natural classification of a group of points or objects such that each data point is exactly within one group. In practical explanation, clustering is commenced from a given representation of n objects and based on quantitative comparison in similarities or relationship, the objects are partitioned into c smaller groups (clusters). Therefore, objects in the same group have high similarities or comparable to one another while objects classified in different groups are dissimilar and unrelated to the objects in other groups.

Clustering algorithm is divided into two main classes: hierarchical and partitional (Kaufman & Rousseeuw, 1990). Hierarchical clustering algorithms can be executed in two ways, agglomerative and divisive. The agglomerative method is starting with each data point in its own cluster and consecutively joining the most similar groups of clusters to move up the cluster hierarchy. Divisive clustering starting by bringing together all data points as one cluster and recursively dividing the cluster into smaller clusters. Single-link and complete-link are the most common algorithms in hierarchical approach. Differing from hierarchical clustering algorithms, partitional clustering algorithms discover all the clusters concurrently as partition of the data. The most considerable algorithm in partitional clustering is k-means. k-means is widely applied in the field of image processing and pattern recognition. The ability to partitioning data based on the density is very valuable to classify and sort out scatter data. k-means has an extensive history since the method introduced by MacQueen (1967). Even though k-means was first proposed decades ago, it is still one of the most widely used algorithms for clustering. Ease of implementation, simplicity, efficiency and empirical success are the main reasons for its popularity. Recent studies by Abraham, Cornillon, Matzner-Lber, and Molinari (2003) and Tarpey (2007) implemented data clustering to find representative curve shapes over distinctive shapes in functional dataset. Their works focus on clustering curves based on the functional structures of the data where the clustering method is employed for curve segmentation. In this study, k-means clustering is functioning as a tool to automatically select the dominant points of the data which are the centre point of clustered data. These points are utilised as the set of parameters $T = \{t_0, t_1, \ldots, t_s\}$ for knot vector calculation in B-spline model estimation. Therefore, the positions of these points determine the shape of the B-spline curve. To our best knowledge, this is the first study to use k-means clustering to estimate the proper set of parameters for knot vector distribution in B-spline curve model identification. The partitioning procedure in k-means implements an uncomplicated approach to classify the given data set through a certain number of c clusters. The basic idea of k-means is to find the clusters that minimise the distance from data values in a cluster to the related cluster centre. k-means commonly uses Euclidean distance as the distance metric for partitioning criterion. This metric calculates the distance between a point and the cluster centre. The Euclidean distance between two sets of data $\{a_1, a_2, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_n\}$ is defined as

$$d_{a_i,b_i} = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{22}$$

Suppose that $V = \{v_1, v_2, \ldots, v_n\}$ as the set of data points to be analysed. This data set will be partitioned into c sets $R = \{R_1, R_2, \ldots, R_c\}$ with respect to the squared Euclidean distance which is defined as $d(v_i, \mu_j) = ||v_i - \mu_j||^2$. Therefore, the goal is to find c cluster centres $\mu = \{\mu_1, \mu_2, \ldots, \mu_c\}$ which minimise the objective in the following form,

$$J_{k-\text{means}} = \text{argmin}_R \sum_{j=1}^{c} \sum_{V \in [R_j]} ||V - \mu_j||^2 \tag{23}$$

where μ_j is calculated as the mean of points in S_j set.

The algorithm requires two user-specified parameters: number of clusters c and initial setting for centre point of each cluster (centroid). The clustering result is highly dependent on the initialisation of centroids. Thus, the calculation is often performed several times, with different initialisations of the centroids. The basic algorithm for k-means clustering is as follows:

(1) Select c values within the range of data set as initial centroids of the clusters.
(2) For each values, calculate the Euclidean distance to all centroids and assign each data point to the closest centroid to form c clusters.
(3) Recompute the centroid of new clusters by calculate the mean value of all points within each clusters.
(4) Repeat (2) and (3) until the result converges.

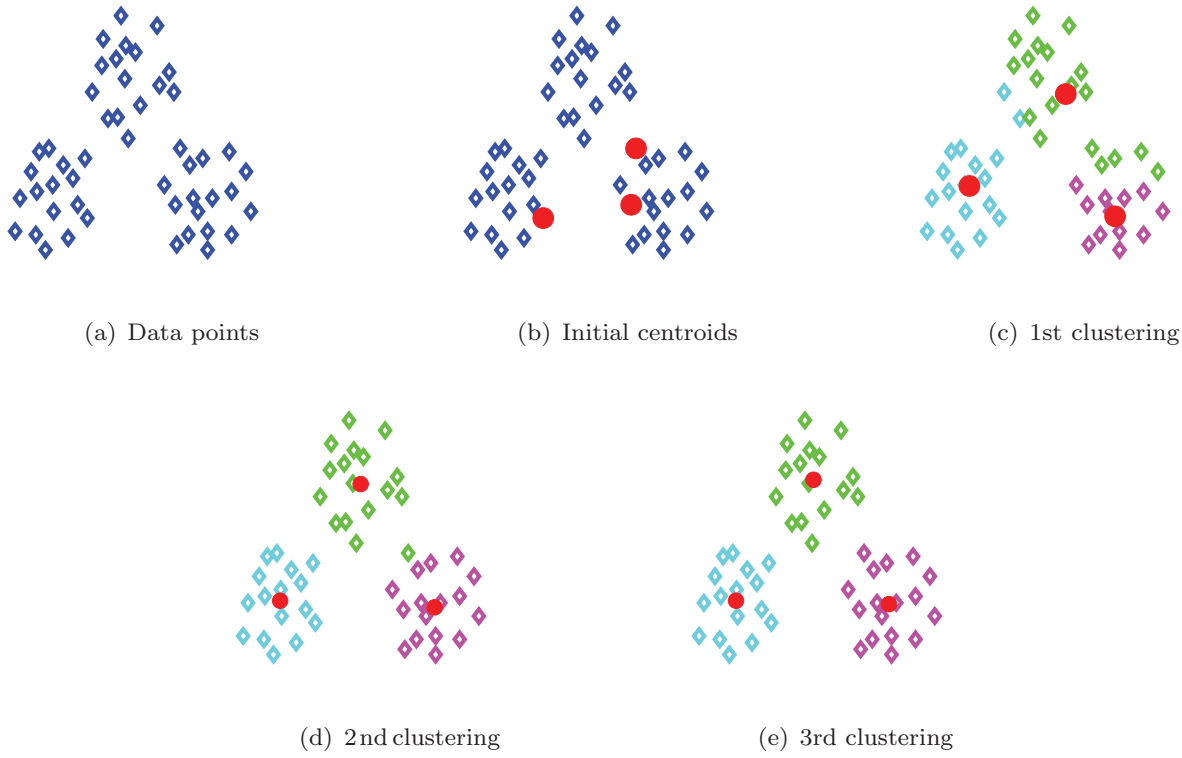


Figure 7. Example of k-means data clustering.

Figure 7 describes the clear picture of k-means algorithm. It shows a set of data points to be clustered into three groups (see Figure 7(a)). Initially, three centroids are chosen randomly (see Figure 7(b)). The clustering process is started by computing the distance between each point to all centroids. Next, the points are grouped based on their closest distance to the nearest centroid (see Figure 7(c)). Subsequently, the mean values of points in each cluster are calculated and used as the new centroids. From this stage, the clustering process is repeated until the position of centroids does not change(see Figure 7(e)).

To identify inverse static output nonlinearity in this study case, the procedure begins with a crude estimation where the models are estimated using random parameters. This step is important to have rough description of model characteristics. Then, k-means performs data clustering and selects the centre point based on the preference number of clusters. Since inverse static output nonlinearity represents the relationship of $y(k)$ as input and $x(k)$ as the output ($x(k) = f(y(k))$), data clustering and centroids calculation are based on data pairs of these variables. In the estimation, parameter c which is the number of required clusters is set equal to the number of required parameters $T = \{t_0, t_1, \ldots, t_s\}$ minus 2 points $(s-2)$ since t_0 is equal to the minimum value of data range and t_s is equal to the maximum point of data range. The results are c centroid points in the form of data pairs of $\{y_i, x_i\}$. Therefore, the y value is taken to represent the centroid value because it will be used as parameter to estimate the B-spline in respect to the output variable $y(k)$.

The implementation of k-means data clustering causes more centroids occur on the dense data curve and less centroids on the flat curve. This objective will support the properties of B-spline model estimation that requires more parameters for higher density region in B-spline curve. The centroid points resulted from this clustering process are used as a set of parameters for knot vector generation. As explained in B-spline section, knot vector of B-spline curve estimation is calculated using parameter $T = \{t_0, t_1, \ldots, t_s\}$ which is replaced by the centroid points $\mu = \{\mu_1, \mu_2, \ldots, \mu_c\}$. This approach eliminates the pre-analysis effort that has to be done to define the appropriate knots.

4.2.4. Parameter estimation

Parameter estimation for linear model and inverse static nonlinearity in Wiener block is carried out in a straightforward manner, by means that both of the models are approximated simultaneously in one process output ($\hat{y}(k)$) equation. Process output can be formulated by equating the linear model (15) and inverse static nonlinear model (20). Using the assumption that the inverse static function is a single-valued smooth function, it is also assumed that $P_0 = 1$ without lose of generality. The output function is composed as

$$N_{0,p}(\hat{y}(k)) = \sum_{l=-\frac{m-1}{2}}^{\frac{m-1}{2}} G(e^{j\omega_l}) f_l(k) - N_{1,p}(y(k))P_1 - N_{2,p}(y(k))P_2 - \ldots - N_{i,p}(y(k))P_i \quad (24)$$

where $(i = 0, \ldots, s)$. To solve the prediction of model parameters, Equation (24) is transformed to matrix notations where the model is composed of parameter and regressor vector. Model

parameter vector is

$$\Theta = \begin{bmatrix} G(e^{j0}) & G(e^{j\omega_1}) & G(e^{-j\omega_1}) & \cdots & G(e^{j\omega\frac{m-1}{2}}) \\ & G(e^{-j\omega\frac{m-1}{2}}) & P_1 & P_2 & \cdots & P_i \end{bmatrix}^T \tag{25}$$

and the corresponding regression vector as

$$\phi(k) = \begin{bmatrix} f_0(k) & f_1(k) & f_{-1}(k) & \ldots & f_{\frac{m-1}{2}}(k) f_{-\frac{m-1}{2}}(k) & -N_{1,p}(y(k)) & -N_{2,p}(y(k)) & \ldots & -N_{i,p}(y(k)) \end{bmatrix} \tag{26}$$

for data samples $k = 0, 1, \ldots, n$, the vector yields

$$\Phi = \begin{bmatrix} f_0(0) & f_1(0) & f_{-1}(0) & \cdots & f_{\frac{m-1}{2}}(0) & f_{-\frac{m-1}{2}}(0) & -N_{1,p}(y(0)) & -N_{2,p}(y(0)) & \cdots & -N_{i,p}(y(0)) \\ f_0(1) & f_1(1) & f_{-1}(1) & \cdots & f_{\frac{m-1}{2}}(1) & f_{-\frac{m-1}{2}}(1) & -N_{1,p}(y(1)) & -N_{2,p}(y(1)) & \cdots & -N_{i,p}(y(1)) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & & \\ f_0(n) & f_1(n) & f_{-1}(n) & \cdots & f_{\frac{m-1}{2}}(n) & f_{-\frac{m-1}{2}}(n) & -N_{1,p}(y(n)) & -N_{2,p}(y(n)) & \cdots & -N_{i,p}(y(n)) \end{bmatrix} \tag{27}$$

The output vector is

$$Y^T = \begin{bmatrix} N_{0,p}y(0) & N_{0,p}y(1) & \cdots & N_{0,p}y(n) \end{bmatrix} \tag{28}$$

Parameters of Θ can be solved as a least squares solution

$$\hat{\Theta} = (\Phi^T\Phi)^{-1}\Phi^T Y \tag{29}$$

4.3. A summary of the system identification algorithm

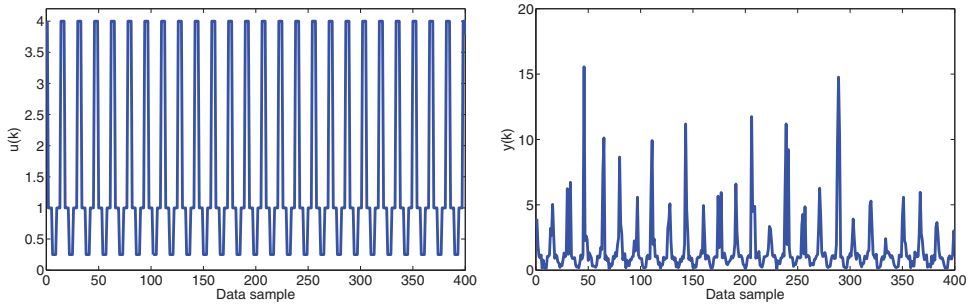
The system identification algorithm can be summarised as follows:

Hammerstein model estimation

(1) Calculate operating points of u using Equation (3)
(2) Predetermine intermediate input variable $w = w_1, w_2, w_3, \ldots, w_n$ with a fixed deviation $\delta w = \frac{w_{\max} - w_{\min}}{n-1}$, where n is the number of operating points of u. $w_1 = w_{\min}$, $w_2 = w_1 + \delta w$, $w_3 = w_2 + \delta w$ and $w_n = w_{\max}$
(3) Map the respective operating points of input u to w to create data pairs
(4) Predetermine the polynomial order
(5) Form the ϕ_{ni} matrix and U vector (4) from the operating points of the input signal u
(6) Obtain the model coefficients of inverse static input nonlinearity using Equation (5)

Wiener model estimation

(1) Predetermine the process settling time M and the model order m for the linear model
(2) Predetermine the B-spline curve degree (p) for the nonlinear model
(3) Choose random points from output data range (y) as a set of parameters for knots calculation
(4) Apply Equation (17) to calculate the knot vector
(5) Compute the basis functions of the curve using Equations (18) and (19)
(6) Form Φ matrix (27) and Y vector (28)
(7) Execute the crude estimation for the model parameters of linear and inverse static output nonlinearity using Equation (29)
(8) Calculate the intermediate output ($x(k)$) using Equation (20) and construct data pairs from the value of output ($y(k)$) and intermediate output ($x(k)$)
(9) Predetermine the number of clusters (c) for the curve
(10) Apply the k-means algorithms in Section 4.2.3 to find c centroid points (y_i, x_i), where $i = 1, 2, \cdots c$
(11) Take the y value of all the centroids as a set of parameter for knot calculation
(12) Repeat steps 4–6
(13) Estimate the model coefficients (Θ) for the linear and inverse static output nonlinearity models using Equation (29)



(a) Data Input (b) Data Output

Figure 8. Data set for system identification.

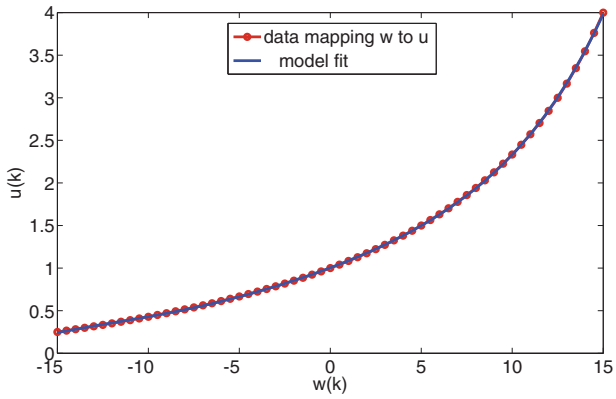


Figure 9. Inverse static input nonlinearity ($u(k) = f^{-1}w(k)$).

5. Identification results

This section covers identification results and predicted model analysis. The results are obtained from the implementation of proposed system identification procedures which have been explained in Section 4. The algorithms are employed to estimate the linear model and the inverse static nonlinearity of input and output model for virtualised software system environment. Data set for identification is a set of observed input and output data from the experimental test-bed which is built based on the process scenario described in Section 2. Four hundred samples of data pairs are generated using multi-level sinusoidal input with minimum 100 requests/s workloads for each client class (see Figure 8). Input variable for the system identification is calculated based on Equation (3) for total available CPU capacity $Cap_{\text{total}} = 100$ and the minimum capacity allocation for each VM is 20 CPU caps.

Validation of the estimated model is based on the value of Mean Squared Error (MSE) which calculates the difference between the prediction result over the true values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{30}$$

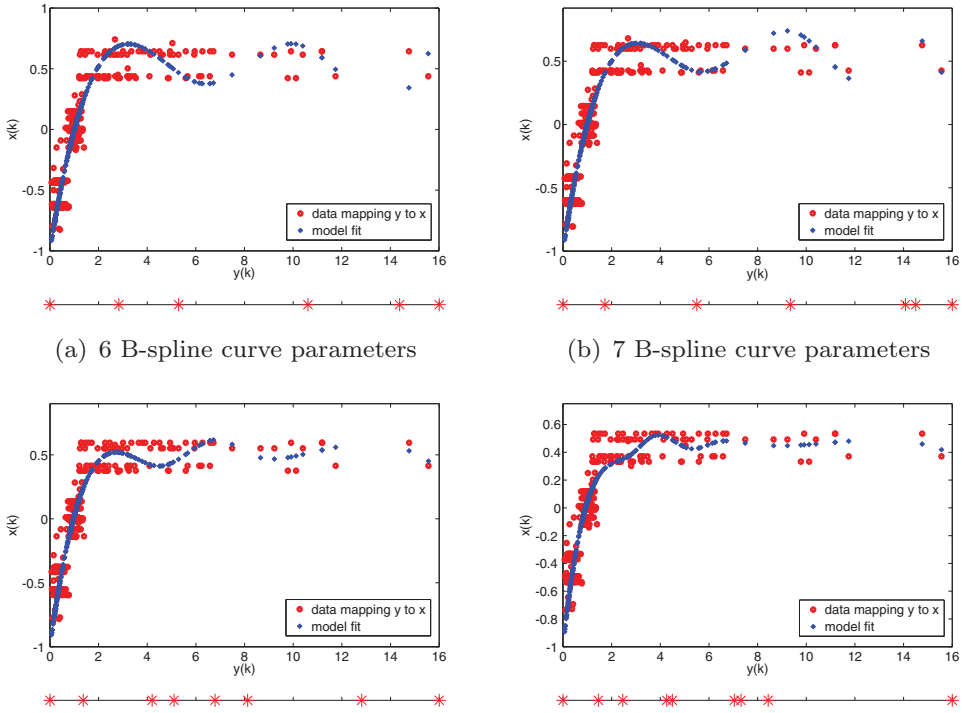
where $\hat{y}$ is the response of predicted model and y is the observed response.

5.1. Hammerstein model

Based on the modelling approach on Section 4.3, the inverse static input nonlinearity models $u(k) = f^{-1}(w(k))$ are approximated in polynomial function. The intermediate variable $w(k)$ values are set arbitrarily by $w_{\min} = -15$, $w_{\max} = 15$ where $\delta w = 0.5$. These settings were used in Patikirikorala et al. (2011a). The data pairs of input u and intermediate variable w are

$$u = \left\{\frac{20}{80}, \frac{21}{79}, \ldots, \frac{50}{50}, \ldots, \frac{79}{21}, \frac{80}{20}\right\}$$
$$w = \{-15, -14.5, \ldots, 1, \ldots, 14.5, 15\} \tag{31}$$

Using least squares method, the inverse static input nonlinearity model is represented in the function below with MSE value 0.000068, and the model fitting can be seen in Figure 9. After identifying input nonlinearity, the rest of the system will be



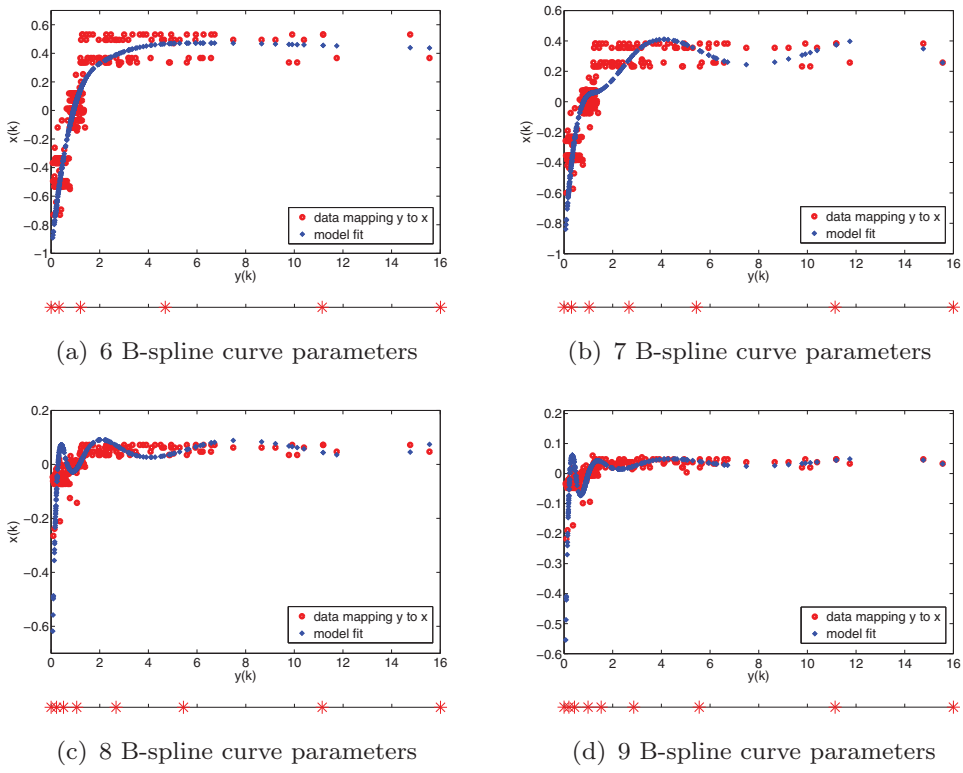
(a) 6 B-spline curve parameters

(b) 7 B-spline curve parameters

(c) 8 B-spline curve parameters

(d) 9 B-spline curve parameters

Figure 10. Estimated B-spline model of inverse static output nonlinearity and the position of non-uniform parameters from random selection.



(a) 6 B-spline curve parameters (b) 7 B-spline curve parameters

(c) 8 B-spline curve parameters (d) 9 B-spline curve parameters

Figure 11. Estimated B-spline model of inverse static output nonlinearity and the position of nonuniform parameters from centroids of clustered data.

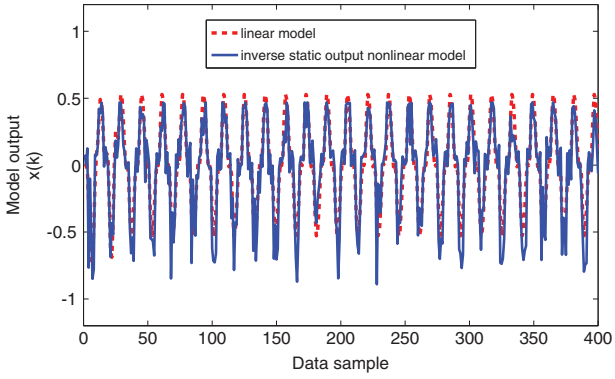


Figure 12. The output of linear and inverse static output nonlinear models.

estimated as a Wiener system.

$$u(k) = 4.17e^{-7}w(k)^5 + 9.34e^{-6}w(k)^4 + 1.018e^{-4}w(k)^3 + 0.0028w(k)^2 + 0.08w(k) + 1.0045 \quad (32)$$

5.2. Wiener model

Referring to the block structure of Figure 1, the intermediate output variable ($x(k)$) represents the output of the linear model and the inverse static output nonlinearity model. FSF structures represent the linear model and B-spline functions capture the inverse static output nonlinearity. Both models are estimated in a straightforward formulation using the algorithm from Section 4.2.4. To apply this approach, the regressor and output vector are composed of the observational data values using Equations (27) and (28). In the subsections below, the model parameters are evaluated in accordance with the model prediction fitness measurement in MSE value. For nonlinear model evaluation, three different estimations are implemented: polynomial, B-spline curve with nonuniform random selection parameters and B-spline curve with nonuniform parameters using the clustering method as proposed in this study.

5.2.1. Linear model in FSF function

Referring to Figure 1, the input of linear model is the intermediate input variable $w(k)$ and the output is the intermediate output variable $x(k)$ and FSF function is employed to capture the linear characteristic. Two parameters should be predetermined: the process settling time in terms of samples (M) and the number of frequency (m). For model analysis purpose, these parameters are changed to have a clear understanding about their influence on the modelling accuracy. Curve parameters for knots distribution in nonlinear model are set to 12 points $T = \{0, 0.51, 1.35, 2.24, 3.87, 6.93, 10.37, 10.41, 12.03, 13.72,$

Table 1. Mean Squared Error (MSE) of model estimation in different FSF parameter.

Model parameters (M/m)	30/9	30/13	30/17	70/9	70/13	70/17
MSE	0.0203	0.0183	0.0170	0.0234	0.0216	0.0208

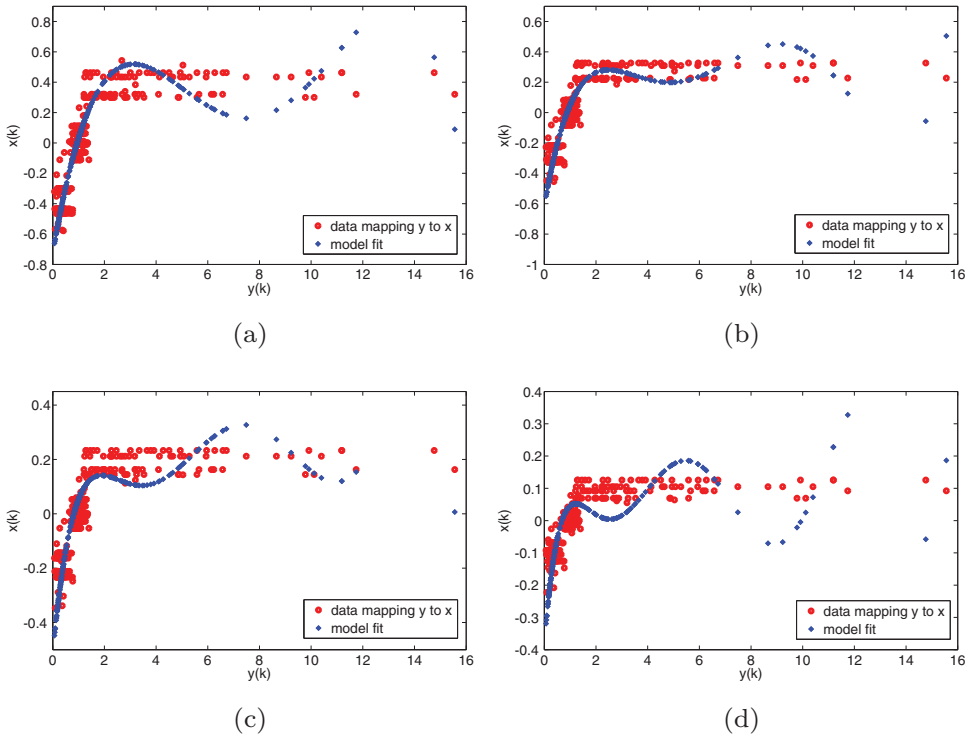


Figure 13. Estimated polynomial model of inverse static output nonlinearity.

Table 2. Mean Squared Error (MSE) of model estimation with non-uniform parameters from random selection.

Model parameters	6	7	8	9
MSE	0.0301	0.0281	0.0257	0.0239

Table 3. Mean Squared Error (MSE) of model estimation with non-uniform parameters from the centroids of clustered data.

Model parameters	6	7	8	9
MSE	0.0236	0.0198	0.0061	0.0039

15.55, 16}. Then, the knot vector is calculated using Equation (17) which yields the nonperiodic knot vector $Q = \{0, 0, 0, 0, 1.36, 2.48, 4.35, 7.06, 9.24, 10.94, 12.05, 13.77, 16, 16, 16, 16\}$. The prediction results are presented in Table 1. It can be seen from the table that increasing parameter M leads to less accuracy of prediction. On the other hand, the variation of the number of frequency (m) shows that the higher m, the better prediction will be gained. Based on the MSE values, the best parameters are $M = 30$ and $m = 13$. These parameters are used in the next system identification analysis presented in this paper.

5.2.2. *Inverse static output nonlinearity in B-spline function*

The input for B-spline curve estimation is the degree of spline and a sequence of parameter that will be used in knot vector generation. In this study, the degree of spline is set to 3 for cubic spline model and knot vector is non-uniform. The parameter selections for knot points calculation are examined in two different approaches, randomly selected parameters and parameters from centroids of clustered data.

Case A. Parameters from random selection

The sequence of the parameters are selected randomly along data values of system output (y). The number of parameters

Table 4. Linear model parameters in Case A.

	Number of B-spline model parameter in estimation			
Linear model coefficient	6	7	8	9
$G(e^{j0})$	0.11756	0.10888	0.1089	0.08858
$G(e^{j\omega_1})$	− 0.04186	− 0.04062	− 0.03959	− 0.02997
$G(e^{j\omega_2})$	− 0.01331	− 0.00942	− 0.00091	− 0.00010
$G(e^{j\omega_3})$	− 0.05172	− 0.04757	− 0.02279	− 0.01466
$G(e^{j\omega_4})$	− 0.14067	− 0.09927	− 0.00796	− 0.01568
$G(e^{j\omega_5})$	0.00613	0.01204	0.00076	0.00578
$G(e^{j\omega_6})$	− 0.01261	− 0.00357	0.00117	0.00198

Table 5. Linear model parameters in Case B.

	Number of B-spline model parameter in estimation			
Linear model coefficient	6	7	8	9
$G(e^{j0})$	0.05424	0.06946	0.09589	0.11295
$G(e^{j\omega_1})$	− 0.02438	− 0.02824	− 0.035	− 0.0412
$G(e^{j\omega_2})$	− 0.01711	− 0.01674	− 0.01579	− 0.01402
$G(e^{j\omega_3})$	− 0.08668	− 0.07922	− 0.06705	− 0.0539
$G(e^{j\omega_4})$	− 0.12962	− 0.13488	− 0.14019	− 0.14526
$G(e^{j\omega_5})$	− 0.0085	− 0.00468	− 0.0059	0.00813
$G(e^{j\omega_6})$	− 0.02152	− 0.02012	− 0.01993	− 0.01254

are varied (6,7,8,9) to express the effect of the length of parameter set when it is chosen randomly. It is inevitable that the predicted model is varying since any change in random values leads to different curve result. These estimation results are designated to demonstrate the accuracy of model identification by directly selecting non-uniform points as the curve parameters for knot vector generation. For each estimation, the identification procedures are repeated five times and selected the best model with the lowest MSE value. It can be seen from the model fitting in Figure 10 and the MSE value in Table 2 that more parameters give better prediction for the B-spline curve. However, this practice raises the issue about parameter redundancy of the predicted model. The best system identification result is the one that could approximate the system with least parameters.

Case B. Parameters from the centroids of clustered data

The estimation procedure and model setting are similar with Case A. The estimation results of inverse static output nonlinearity using data clustering approach can be seen in Figure 11. It shows that more points occupied the complex and dense area and few points on flat area. This is the superiority of the proposed method over basic parameter selection method. Model fittings of B-spline curve on Figures 10 and 11 highlight the importance of proper parameter selection for B-spline curve knot vector. Clustering the data values before choosing the parameters is a reasonable way to set the suitable breakpoints for knot vector, specially for systems such as virtualised software system where its nonlinearities lead to a scatter data relation between the input and output of the system. The improved model parameters selection method brings interesting result in terms of prediction accuracy. More accurate models with clustering approach can be achieved with less parameters (see Table 3). With only 6 parameters from clustering method, the error prediction is less than the estimation with 9 parameters in random selection method. The fitness of model prediction on intermediate variable $x(k)$ with 6 parameters from clustering method point is represented in Figure 12. Tables 4 and 5 represent the linear model parameters for *CaseA* and *CaseB* respectively where the real values of $G(e^{-j\omega_l})$ are similar with $G(e^{j\omega_l})$.

In spite of the fact that increasing the number of cluster gives smaller MSE value of the prediction, the estimated model will be overfitting which is unfavourable for control design. Therefore, the curve parameters should be located suitably to have an accurate estimation, yet with less parameters to avoid multiplicity and overfitting issue. In short, the proposed approach to apply clustering method for proper selection curve parameters is very effective to address these related objectives.

Table 6. Mean Squared Error (MSE) of model estimation in polynomial function.

Model order	4	5	6	7
MSE	0.02	0.01	0.007	0.005

5.2.3. Comparative study using polynomial function

Identification of nonlinear model in virtualised software system can also be conducted in polynomial function (Aryani, Wang, & Patikirikorala, 2014). The Wiener models are formulated using FSF terms and polynomial function to estimate $x(k) = f^{-1}(y(k))$. The polynomial model fitting can be seen in Figure 13 and the MSE values in Table 6. The results show that the higher polynomial order gives better model prediction. However, the issue of multiplicity, which has been a concern in nonlinear dynamics modelling, can be seen from the curve fitting of the estimated models.

6. Conclusion

This paper proposes a system identification method for estimating virtualised software system dynamics within the framework of a Hammerstein–Wiener model. It is a straightforward approach with significant improvement for the Wiener models where the linear dynamic is represented in FSF function and the nonlinear dynamic in B-spline function. Furthermore, data clustering method is utilised in curve parameter selection to manage the knot position of B-spline curve. The proposed approach has been implemented using experimental data from the virtualised software system test-bed and the system identification results demonstrate significant improvement for parameter selection of nonlinear model.

Acknowledgments

The authors gratefully acknowledged the Australia Award Scholarship provided to Dharma Aryani.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Dharma Aryani is a Ph.D. candidate in the School of Electrical and Computer Engineering, RMIT University, VIC, Australia. She received her Master of Engineering degree from the Department of Electrical Engineering, University of Indonesia, Indonesia. Since 2013, she has been a lecturer in the School of Electrical Engineering, The State Polytechnic of Ujung Pandang, Indonesia. Her Research interests are system identification, adaptive control, model predictive control, and process control.

Liuping Wang received her Ph.D. degree in 1989 from the Department of Automatic Control and Systems Engineering, University of Sheffield, UK. Upon completion of her Ph.D. degree, she worked in the Department of Chemical Engineering at the University of Toronto, Canada for eight years in the field of process control. From 1998 to 2002, she worked in the Center for Integrated Dynamics and Control, University of Newcastle, Australia. In February 2002, she joined the School of Electrical and Computer Engineering, RMIT University, Australia where she is a Professor of Control Engineering

Tharindu Patikirikorala receieved his Ph.D. degree in 2014 from the Faculty of Information and Communication Technologies, Swinbure University of Technology. He is an experienced software engineer and research fellow from Swinburne University of Technology. His research interests are cloud computing, feedback control and self-adaptive software systems.

References

Abraham, C., Cornillon, P.A., Matzner-Lber, E., & Molinari, N. (2003). Unsupervised curve clustering using B-splines. *Scandinavian Journal of Statistics, 30*(3), 581–595.

Anderberg, M.R. (1973). *Cluster analysis for applications*. New York, NY: Academic Press.

Andrzejak, A., Arlitt, M., & Rolia, J. (2002). *Bounding the resource savings of utility computing models*. (Technical Report No. HPL-2002-339). Retrieved from: http://www.hpl.hp.com/techreports/2002/HPL-2002-339.html

Armbrust, M., Fox, A., Rean, G., Joseph, A.D., Katz, R., Konwinski, A., … Zaharia, M. (2009). *Above the clouds: A berkeley view of cloud computing*. (Technical Report No. UCB/EECS-2009-28). Retrieved from:http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

Astrom, K.J., & Wittenmark, B. (2013). *Adaptive control* (2nd ed.). Mineola, NY: Dover.

Aryani, D., Wang, L., & Patikirikorala, T. (2014). Control oriented system identification for virtualized software systems. *IFAC Proceedings, 47*(3), 4122–4127. doi:10.3182/20140824-6-ZA-1003.01100

Azeez, A., Perera, S., Gamage, D., Linton, R., Siriwardana, P., Leelaratne, D., … Fremantle, P. (2010). Multi-tenant SOA middleware for cloud computing. In *2010 IEEE 3rd International Conference on Cloud Computing* (pp. 458–465). Miami, FL: IEEE. doi: 10.1109/CLOUD.2010.50

Bai, E.W. (1998). An optimal two-stage identification algorithm for Hammerstein–Wiener nonlinear systems. *Automatica, 34*, 333–338.

Bai, E.W., & Fu, M. (2002). A blind approach to Hammerstein model identification. *IEEE Transaction on Signal Processing, 50*(7), 1610–1619.

Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., … Warfield, A. (2003). Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles* (pp. 164–177). New York, NY: ACM. doi:10.1145/945445.945462

Billings, S.A., & Fakhouri, S.Y. (1979). Nonlinear system identification using the Hammerstein model. *International Journal of Systems Science, 10*, 567–578.

Billings, S.A., & Fakhouri, S.Y. (1982). Identification of systems containing linear dynamic and static nonlinear elements. *Automatica, 18*(1), 15–26.

Bitmead, R.R., & Anderson, B.D.O. (1981). Adaptive frequency sampling filters. *IEEE Transactions on Circuits and Systems, 28*, 524–534.

Bloemen, H.H., Van den Boom, T.J., & Verbruggen, H.B. (2000). Model based predictive control for Hammerstein systems. *Proceedings of the 39th IEEE Conference on Decision and Control,* 74(5), 4963–4968). doi:10.1109/CDC.2001.914719

Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems, 25*(6), 599–616.

Chenyang, L., Abdelzaber, T.F., Stankovic, J.A., & Son, S.H. (2001). A feedback control approach for guaranteeing relative delays in web servers. In *Proceedings of Real-Time Technology and Applications Symposium, 2001, Seventh IEEE* (pp. 51–62). IEEE. doi:10.1109/RTTAS.2001.929865

DeBoor, C. (1978). *A practical guide to splines*. Applied Mathematical Sciences. New York, NY: Springer-Verlag.

Farin, G. (1994). *Curves and surfaces for computer-aided geometric design: A practical guide*. Boston: Academic Press.

Fruzzetti, K.P., Palazoglu, A., & McDonald, K.A. (1997). Nonlinear model predictive control using Hammerstein models. *Journal of Process Control, 7*, 31–41.

Greblicki, W. (1992). Nonparametric identification of wiener systems. *IEEE Transactions on Information Theory, 38*, 1487–1493.

Hellerstein, J.L. (2004). Challenges in control engineering of computing systems. In *Proceedings of 2004 American Control Conference (ACC)*.

Hong, X., Gong, Y., & Chen, S. (2011). A wiener model for high power amplifiers using B-spline function approximation. In *Proceedings of the 17th International Conference on Digital Signal Processing* (pp. 1–5). IEEE. doi:10.1109/ICDSP.2011.6004878

Hong, X., & Mitchell, R.J. (2006). A pole assignment controller for Bezier–Bernstein polynomial based Hammerstein model. In *Proceedings of International Control Conference (ICC) 2006*. Glasgow, UK.

Hong, X., Mitchell, R.J., & Chen, S. (2012). Modelling and control of Hammerstein system using B-spline approximation and the inverse of De Boor algorithm. *International Journal of Systems Science, 43*(10), 1976–1984.

Hong, X., Mitchell, R.J., & Chen, S. (2013). System identification of Wiener systems with B-spline functions using De Boor recursion. *International Journal of Systems Science, 44*(9), 1666–1674.

Hughes, M.C., & Westwick, D.T. (2005). Identification of IIR Wiener system with spline nonlinearities that have variable knots. *IEEE Transactions on Automatic Control, 50*, 1617–1622.

Ioannou, P., & Sun, J. (1996). *Robust adaptive control*. Upper Saddle River, NJ: Prentice-Hall.

Jurado, F. (2006). Hammerstein-model-based predictive control of microturbines. *International Journal of Energy Research, 7*, 511–521.

Kalafatis, A.D., Arifin, N., Wang, L., & Cluett, W.R. (1995). A new approach to identification of pH processes based on Wiener model. *Chemical Engineering Science, 50*, 3693–3701.

Kalafatis, A.D., Wang, L., & Cluett, W.R. (1997). Identification of Wiener-type nonlinear systems in a noisy environment. *International Journal of Control, 66*, 923–941.

Karamanolis, C., Karlsson, M., & Zhu, X. (2005). Designing controllable computer systems. In *Proceedings of the 10th Conference on Hot Topics in Operating Systems,* pp. 49–54. Santa Fe, NM.

Karlsson, M., Zhu, X., & Karamanolis, C. (2005). An adaptive optimal controller for non-intrusive performance differentiation in computing services. In *International conference on control and automation* (Vol. 2, pp. 709–714). IEEE. doi:10.1109/ICCA.2005.1528215

Kaufman, L., & Rousseeuw, P.J. (1990). *Finding groups in data-an introduction to cluster analysis*. New York, NY: John Wiley Sons.

Kusic, D., & Kandasamy, N. (2006). Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing system. *Cluster Computing, 10*(4), 395–408.

Ljung, L. (1999). *System identification: Theory for the users* (2nd ed.). Upper Saddle River, NY: Prentice Hall information and system sciences series.

Lu, C., Abdelzaher, T.F., Stankovic, J.A., & Son, S.H. (2001). A feedback control approach for guaranteeing relative delays in web servers. In *Proceedings of the Seventh IEEE Real-Time Technology and Applications Symposium* (pp. 51–56). IEEE. doi:10.1109/RTTAS.2001.929865

Lu, Y., Abdelzaher, T., Lu, C., & Tao, G. (2002). An adaptive control framework for QoS guarantees and its application to differentiated caching services. In *Proceedings of the 10th IEEE International Workshop Quality of Service* (pp. 23–32). Miami, FL: IEEE. doi:10.1109/IWQoS.2002.1006571

Lu, Y., Abdelzaher, T., Lu, C., Sha, L., & Liu, X. (2003). Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers. In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium* (pp. 208–217). IEEE. doi:10.1109/RTTAS.2003.1203053

Lu, Y., Saxena, A., & Abdelzaher, T. (2001). Differentiated caching services; a control-theoretical approach. In *Proceedings of the 21st International Conference on Distributed Computing Systems* (pp. 615–612). IEEE. doi:10.1109/ICDSC.2001.918992

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 281–297). California, CA: University of California Press (Statistics).

Padala, P., Hou, K.Y., Shin, K.G., & Zhu, X. (2009). Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European Conference on Computer Systems* (pp. 13–26). New York, NY: ACM. doi:10.1145/1519065.1519068

Padala, P. (2010). *Automated management of virtualized data center* (PhD Thesis). University of Michigan.

Padala, P., Shin, K.G., Zhu, X., Uysal, M., Wang, S., Singhal, S., ... Salem, K. (2007). Adaptive control of virtualized resources in utility computing environments. In *Proceedings of SIGOPS/EuroSys European Conference on Computer Systems (EuroSys07)* (pp. 289–302). New York, NY: ACM. doi:10.1145/1272996.1273026

Pajunen, G.A. (1992). Adaptive control of Wiener type nonlinear systems. *Automatica, 28*(4), 781–785.

Pan, W., Mu, D., Wu, H., & Yao, L. (2008). Feedback control-based QoS guarantees in web application servers. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications* (pp. 328–334). IEEE. doi:10.1109/HPCC.2008.106

Patikirikorala, T. Colman, A., Han, J., & Wang, L. (2012). A systematic survey on the design of self-adaptive software systems using control engineering approaches. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)* (pp. 33–42). Piscataway, NJ: IEEE Press.

Patikirikorala, T., Wang, L., & Colman, A. (2011). Towards optimal performance and resource management in web systems via model predictive control. In *Proceedings of the 2011 Australian Control Conference* (pp. 469–474). IEEE.

Patikirikorala, T., Wang, L., Colman, A., & Han, J. (2011a). Hammerstein-Wiener nonlinear model based predictive control for relative QoS performance and resource management of software systems. *Control Engineering Practice, 20* (1), 49–61.

Patikirikorala, T., Wang, L., Colman, A., & Han, J. (2011b). A multi-model framework to implement self-managing control systems for QoS management. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 218–227). New York, NY: ACM. doi:10.1145/1988008.1988040

Patikirikorala, T., Wang, L., Colman, A., & Han, J. (2012). An evaluation of multi-model self-managing control schemes for adaptive performance management of software systems. *Journal of Systems and Software, 85*, 2678–2696.

Patikirikorala, T., Wang, L., Colman, A., & Han, J. (2014). Differentiated performance management in virtualized environments using nonlinear control. Retrieved March 2, 2015, from http://www.ict.swin.edu.au/personal/tpatikirikorala/docs/tharindu_pds_2013.pdf.

Piegl, L., & Tiller, W. (1987). Curve and surface constructions using rational B-splines. *Computer-Aided Design, 19*(9), 485–498.

Piegl, L., & Tiller, W. (1995). *The NURBS book*. New York, NY: Springer-Verlag.

Tarpey, T. (2007). Linear transformations and the k-means clustering algorithm: Applications to clustering curves. *The American Statistician, 61*(1), 34–40.

Voros, J. (1995). Identification of nonlinear dynamic systems using extended Hammerstein and Wiener models. *Control-Theory and Advanced Technology, 10*(4, Part 2), 1203–1212.

Wang, L., & Cluett, W.R. (1994). A more efficient way to estimate step response coefficients using the FSF model. In *Proceedings of 1994 American Control Conference* (Vol. 1, pp. 532–535). IEEE.

Wang, L., & Cluett, W.R. (1997). Frequency sampling filters: An improved model structure for step response identification. *Automatica, 33*, 939–944.

Wang, L., & Cluett, W.R. (2000). *From plant data to process control: Ideas for process identification and PID design*. London: Francis Taylor.

Weiss, A. (2007). *Computing in the clouds*. netWorker, 11(4), 16-25. New York, NY: ACM Press. doi:10.1145/1327512.1327513

Wu, K., Lilja, D., & Bai, H. (2005). The applicability of adaptive control theory to QoS design: Limitations and solutions. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium* (pp. 4–8). IEEE. doi:10.1109/IPDPS.2005.424

Zhu, X., Wang, Z., & Singhal, S. (2006). Utility-driven workload management using nested control design. In *American Control Conference* (pp. 709–714). IEEE. doi:10.1109/ACC.2006.1657688

Zhu, Y. (1999). Distillation column identification for control using Wiener model. In *Proceedings of the American Control Conference* (pp. 3462–3466). San Diego, CA: IEEE. doi:10.1109/ACC.1999.782408

Zhu, Y. (2000). Hammerstein model identification for control using ASYM. *International Journal of Control, 73*(18), 1692–1702.