



Patikirikoral, T., Wang, L., Colman, A., & Han, J. (2012). Hammerstein-Wiener nonlinear model based predictive control for relative QoS performance and resource management of software systems.

Originally published in *Control Engineering Practice*, 20(1), 49–61.  
Available from: <http://dx.doi.org/10.1016/j.conengprac.2011.09.003>

Copyright © 2011 Elsevier Ltd. All rights reserved.

This is the author's version of the work. It is posted here with the permission of the publisher for your personal use. No further distribution is permitted. If your library has a subscription to this journal, you may also be able to access the published version via the library catalogue.



# Hammerstein-Weiner nonlinear model based predictive control for QoS management in complex software systems

Tharindu Patikirikoral, Alan Colman, Jun Han, Liuping Wang

---

## Abstract

*Keywords:*

---

## 1. Introduction

The increasing complexity and scale of the software systems demand effective and efficient mechanisms to manage them in a systematic way. Multi-tire e-commerce systems, banking systems, cloud computing environments and many other businesses rely on software systems to deliver desired functional services, while maintaining the nonfunctional aspects at an acceptable level. The response time, throughput and security are some of these non-functional attributes such systems must maintain at all times to avoid customer dissatisfaction. Traditional methods such as manual tuning via human intervention or fixed/ ad-hoc policies depending on the peak demand has proven to be costly, error-prone, time consuming and increasingly difficult to design [1, 2].

For instance, emerging technologies such as cloud computing enables different consumers to deploy their software systems in shared resource pools, provided by the cloud environment provider. It is an important objective of the cloud consumers and providers to allocate limited amount of available resources to maximize revenue. In such environments, static resource allocation decisions depending on estimated peak resource demands either end up with dissatisfied customers when the demand is underestimated or wasting resources when it is over estimated. Typically, many software systems provide functional services to multiple client classes. Consequently, these systems may face varying workloads from different client classes over time.

In addition, these clients may have different service level agreements (SLA) and performance isolation requirements with the service provider. Hence, the resource provision decisions have to be made depending on the varying workloads and demands of multiple client classes, while guaranteeing the quality of service agreements (constraints). These requirements arise for management of software systems due to business domain. In addition, management of such systems is difficult because of the inherent behavior of the software system. There are many disturbances/behaviors in the software systems that could affect the runtime management process. For instance, thread context switching in multithread applications, different scheduling mechanisms of the operating systems, garbage collection of execution environments and memory/CPU competition among components are such disturbances/behaviors. Many of these behaviors cannot be effectively modeled to be used in performance management of complex software systems.

These performance management requirements and inherent system behavior and disturbances demand effective methodologies to integrate decision making capabilities in to the software systems, reducing human intervention. Feedback control is one such methodology that could be effectively used to achieve above requirements. The feedback control has been used to maintain quality of service attributes such as response time, throughput and CPU consumption in complex software systems with minimal human intervention ( eg: [4, 5, 6, 7]). However, application of feedback control is difficult due to lack of first principle models, inherent nonlinear nature of the system (including actuators and sensors) and discrete/discontinuous control inputs [8, 9].

The existing approaches model inherently nonlinear software systems using a linear black-box model. The linear models (typically a linear time invariant models) of the system captures the dynamics of the nominal operating region it was created. That is the model will be sufficiently accurate around that operating region. Inherent nonlinear behavior of the system may affect the accuracy of the model, when the system operates away from nominal operating region. Thus, such approaches may fail when the system is required to operate in the entire operating range. In addition, implemented actuators and sensors could integrate nonlinearities in to the system, which has to be modeled by this single linear model. Thus, representing a complex system with a single linear model is quite a challenging task. In contrast, the nonlinear modeling of the system may provide means to describe the global behavior of the system in the entire operating range, instead of limiting to a

certain operating region [10].

In this work, we investigate the performance improvement that can be achieved by modeling a software system using a nonlinear models and Model Predictive Control (MPC). In particular, we use a technique called Hammerstein-Wiener model based predictive control. The Hammerstein-Wiener model is a block structure model that has been useful to represent nonlinear dynamics of chemical [11], electrical [12] and biological [13] systems. Figure 1 shows the block diagram of Hammerstein-Wiener model.

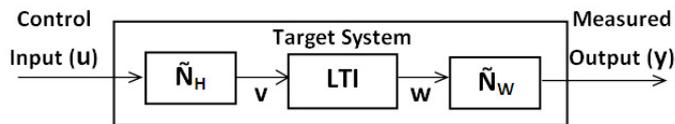


Figure 1: Hammerstein-Wiener model

In the Hammerstein-Wiener model, the linear block of the model is surrounded by two nonlinear blocks. The entire model can be divided in to two segments called Hammerstein and Wiener. The Hammerstein model has a nonlinear component followed by a linear component. In contrast, the Wiener model has a nonlinear component followed by a linear component. These nonlinear blocks assumed to model the static nonlinearities in the system. When these two schemes are combined together we can call it as Hammerstein-Wiener model.  $u$  and  $y$  denote the input and output of the Hammerstein-Wiener model respectively. The intermediate variables  $v$  and  $w$  are not measurable.

Typically, a model of the system (the relationship between the input and output) is constructed using a system identification experiment [14]. The generated input-output data pares are fit into a sufficiently accurate linear time invariant (LTI) model. The model structure selection and parameter derivation is involved in this model fitting process. Finally this model will be used to analyze the system behavior or controller design purposes. However, identification of Hammerstein-Wiener model becomes complex because of the addition of nonlinear blocks. The complexity induces because, instead of concentrating only on linear block of the model we have to be concerned about the model structure selection and the parameters derivation of the nonlinear blocks. Even though, we have additional parameters to decide we

can only use input-output ( $u, y$ ) data, since the intermediate ( $v, w$ ) variables are not measurable. The nonlinear components can be represented by many different nonlinear functions. The piecewise linear, polynomial with degree  $n$ , nonlinear-ARMA (auto regressive moving average) models and splines are some of these functions used in literature to approximate these static nonlinearities [11, 15, 16, 17, 18]. There are different types of system identification and estimation algorithms to derive parameters for different functions from input and output data. However, if some details about these static nonlinearities are known at the design time modeling process could be simplified. For instance, known nonlinearities of the control input could be modeled and integrated to the system as an input nonlinear component of the Hammerstein-Wiener model, instead of using aforementioned approximation methods.

The model predictive control is an effective controlling mechanism which utilizes the dynamic model of the system to predict and optimize the future behavior of the system [20, 21, 29]. The model predictive control has proven to be useful to control many physical plants due to the capability of constraints handling and integrating multiple variables. However, many applications of MPC are based on linear model of the system. Even the accuracy of the linear model prevails around the nominal operating region, linear models have proved to be useful to control nonlinear processes [26]. When linear models encounter highly nonlinear processes, the controller performance tends to degrade due to incapability of linear models to describe the global behavior of the system. Motivated by such cases, nonlinear-MPC techniques are developed to control inherently nonlinear systems. Several nonlinear models have been already applied with MPC and proved to be useful. The physical (first principle) models, neural networks, nonlinear-ARMA models and Volterra models [18, 26, 29] can be highlighted as such models. Many of these models impose difficulties in identification of the model due to nonlinear operators and number of parameters [30]. In addition, computational and time complexity of the constraint problem that has to be solved online increases [30, 31]. Solving a nonlinear quadratic programming problem online is difficult compared to standard quadratic programming problem [31]. The attractive feature of block structured nonlinear modeling is that it integrates static nonlinearity into the control system by preserving some of the numerical properties of original MPC constraint problem [26]. The integration of inverse nonlinear compensators into the control system enables the predictive controller to work with transformed variables so that we

can still use standard quadratic programming, avoiding unnecessary computational complexity of solving nonlinear constraint problem [26, 29, 30]. However, due to involvement of transformed dynamics optimization problem may settle for sub optimal solution compared to the original problem [30]. There are many work that has shown individual application of Hammerstein model [31, 12] and Wiener model [26, 29] can improve the control of physical plants. In contrast, successful application of combined Hammerstein-Wiener model based predictive control is rare. In this work we show how to formulate the constraint optimization problem with transformed variables for Hammerstein-Wiener model based predictive controller.

The relative guarantee is one of the design schemes that is looked at many times in literature [6, 19]. The design of these control schemes integrates actuator and sensor nonlinearities to the systems, so that even with a sufficiently accurate model of the system, the feedback control system settles down with large steady state errors and overshooting. Hence, the system becomes oscillatory and unstable when it operates away from its nominal operating point. In this work we illustrate the performance improvements that can be achieved by integrating the known static control input nonlinearities and estimating unknown output nonlinearities as Hammerstein-Wiener model. In addition, we use model predictive [20, 21] controller(MPC) and try to utilize the constructed Hammerstein-Wiener model in controlling the system. The sufficiently accurate model of the system is an important requirement to apply model in MPC. To this end, we show that Hammerstein-Wiener model based predictive controller achieve significantly better performance compared to single linear model of the system. In addition, the proposed design process preserves the numerical properties of linear MPC constraint problem, while incorporating nonlinear modeling of the system. Thus, it is much computationally inexpensive compared to solving nonlinear quadratic problem.

## 2. Related Work

The feedback control has been used to incorporate self-managing capabilities to software systems in last decade. Use of control theory to control web server systems [6, 22], cache and storage systems [5, 19] and data centers/server clusters [7, 23] are such attempts. However, they design a liner

system with linear control inputs to control an inherently nonlinear system. Feedback control was also applied in multi-client class environments in [5, 6, 7]. D. Kusic et al, proposed a feedback control mechanism [7] to allocate servers in the cluster to specific client classes. Their approach was designed considering detailed information about the behavior of the system, instead of black-box modeling (modeling the system using only input and output data). In many legacy software systems (including the class of systems described in section 3), may not have detailed behavioral and workload specific information so that such designs are hard to apply. In this work we use black-box modeling similar to [4]. Relative guarantee design scheme and feedback control is discussed in [6, 19, 24], however the performance degradation due to nonlinearities introduced by design is not addressed. In addition, they do not consider other common policies (minimum amount of allocation) in the design.

In this work, we model the system under consideration as a black-box and try to design a relative guarantee control scheme for a multi-client class environment to achieve the desired performance objectives of the software/service provider.

### 3. Case study

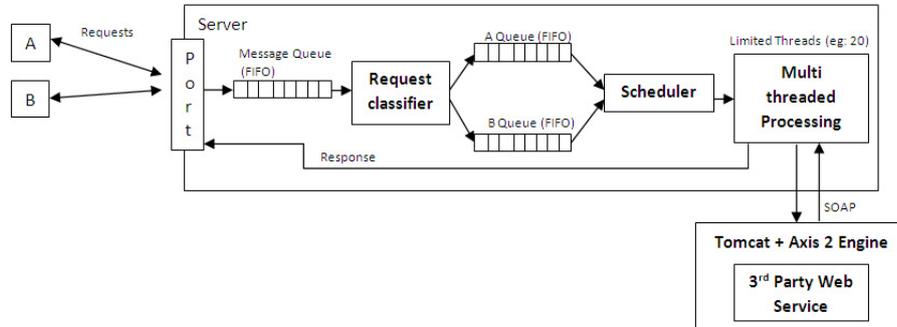


Figure 2: The target software system

The software system in figure 2 illustrates one of the common resource allocation problems in traveling (flight) reservation systems. The server has to communicate with a 3<sup>rd</sup> party supplier to respond to client requests (eg: check/book flight availabilities). However, the 3<sup>rd</sup> party supplier only provides limited number of sessions (20 in the scenario discussed in this paper,

typically range from 50-200 in a real systems) to communicate with them. Assume that, two clients (travel agents A and B) are interested in the services provided by this server (service provider). The objective of the server is to allocate the limited number of sessions (communication links) provided by the 3<sup>rd</sup> party supplier between agents A and B, and maintain the response time of the requests at an agreed level, under varying workloads. Fast responses to the requests are one of the main objectives of the travel reservation web applications to avoid customer dissatisfaction (e.g: flight search results).

A prototype of a similar reservation system was developed having the architecture shown in figure 2. The services of the sever can be accessed by connecting to the socket (IP and port number). After the connection is made the clients can send different messages invoking different service methods. When a message is received by the message queue, time stamp-1 is applied and then the request is classified according to the client class and put it to the specific client queue. Then on the availability of the sessions, the scheduler access the client queues in first come first serve (FIFO) fashion, and schedules the messages to be sent to the 3<sup>rd</sup> party supplier. When the response is received by the server it is sent back to the client through the socket. The time stamp-2 is applied before the response is sent back via the client socket.

In this work we allocate sessions depending on varying workloads so that the response time could be maintained within the bounds of response time mentioned in Service Level Agreement (SLA). The end-to-end response time consist of three main delay components. The communication delay, connection delay and the processing delay [6]. In this work we refer to processing delay as the response time because it is hard to guarantee the end-to-end repose time due to network specific issues and different scheduling designs of different operating systems. The response time for a single request is the time difference between time stamp-1 and 2 and it is measured by seconds. However, for control purposes the average response time of the workload is taken within in 2 second sampling window. Let us denote the average response time of workload A and B as  $R_a$  and  $R_b$  respectively. Similarly, the session allocation between workload A and B as  $S_a$  and  $S_b$ . Then, a relative guarantee feedback [6] control system is composed to achieve the control objectives of the server.

According to the relative guarantee scheme the manipulated variable is the ratio of session allocation, represented by  $\frac{S_a}{S_b}$  and the output variable is the

ratio of average response time of the workloads, represented by  $\frac{R_a}{R_b}$ . The control objective of this control system is to maintain a constant response time ratio between two workloads  $A$  and  $B$ . Such control objectives are useful in multi-client class environment, because it gives the designer flexibility to control the relative importance of the client class. For instance, if client  $A$  is important for the service provider more than client  $B$ , it can be incorporated as an objective in the decision making. The costs, penalties for violating SLAs and response time requirements can be used to decide this relative importance.

In addition to the main control objective, the scheduler is constrained with set of other control policies to avoid starvation of sudden burst of requests. The minimum number of sessions for a client class is declared to serve this purpose. We constrained  $S_a, S_b \geq 4$ , however whether to impose these policies or not depends on the requirements of the application. When the session ratio is decided to be changed, the ongoing communication of sessions is not halted immediately. We also take a non-preemptive design approach similar to [6], where we let the current session communications to terminate, before taking over them to implement new scheduling decisions. We assume that this actuation delay is negligible. In addition, relative guarantee problem to be meaningful we have to assume there is always at least one request arrives from each client class in each sampling period. Further, we assume that different message types that are sent are equal, when taking the average response time.

The prototype system was deployed in a machine with Intel Core(TM)2duo E8400 CPU@3.00 GHz, 2.99 GHz processor and 2 GB memory. To simulate two client workloads we use a machine with Core(TM)2duo E6550 CPU@2.33 GHz, 2.33 GHz processor and 3 GB memory. The server communicates with 3<sup>rd</sup> party supplier component using Simple Object Access protocol (SOAP), similar to most of the real world scenarios. The 3<sup>rd</sup> party supplier component was designed using a Java as a web service which was deployed in a Tomcat 5.5 with Axis 2 web service engine. The machines were connected via 1 Gps Ethernet. From the above requirements the control input  $\frac{S_a}{S_b}$ , can only take certain discrete values. Let  $S_a = 20 - S_b$  then  $\frac{S_a}{S_b} = \frac{20-S_b}{S_b}$ , where  $\frac{S_a}{S_b} = \frac{4}{16}, \frac{5}{15}, \frac{6}{14} \dots \frac{15}{5}, \frac{16}{4}$ . If we closely analyze these operating points, we can see that they are discontinuous with non-equal gaps between two consecutive points.

In this work we show how these known input nonlinearities and unknown

static output nonlinearities can affect the performance of a relative guarantee control system. We investigate the possibility of performance improvements by incorporating these nonlinearities in the model as Hammerstein-Wiener models. In addition, we provide a systematic process of designing and integrating Hammerstein-Wiener model for relative guarantee control problem. However, we believe that it is more useful in cases where input nonlinearities are known and static output nonlinearities can be estimated. Moreover, we show how we can design a model predictive controller using Hammerstein-Wiener model. Finally, we provide a performance analysis and performance comparison of fixed controller, Single linear model, Hammerstein model, Wiener model and Hammerstein-Wiener model based MPC.

#### 4. Dynamic modeling

To analyze the behavior of the system and design a feedback controller, construction of a relationship between control input and measured output is a necessity. Typically, system identification [14] is widely used technique to build this relationship or the model of the system. In system identification, an experiment is conducted to gather input-output data from the system. Specially designed (persistently exciting [14, 25]) control input signal is applied in to the system and the measured output is observed for certain period of time. Afterwards, the input and output data pairs are used in the selection of the structure and deriving the parameters of the model. There are different types of models that could be used to represent the system model. For instance, Autoregressive Moving Average (ARMA) models, Finite Impulse Response (FIR) models and State space models are used as common models. Depending on the structure of the model the number of parameters to be decided going to vary. In general, these models represent the liner systems or linear components of a nonlinear system such as the Hammerstein/Hammerstein-Wiener models.

To model the prototypical system described in section 4, we use two approaches. In section 4.1 single linear model of the system is constructed to represent the system. In this case, the entire system is assumed to be linear. Section 4.2,4.3,4.4, provides the systematic design process of the same system as different nonlinear block structured (Hammerstein, Wiener and Hammerstein-Wiener) models.

#### 4.1. Single linear model

A pseudo random signal consisting of possible values of  $u$  was injected into the system and the measured output was observed for 600 sample periods. Then the gathered data was divided in two sets called estimation set and test set. Data samples between 1st to 400<sup>th</sup> samples were included in the estimation set, which was used to construct the model of the system. The rest of the data samples were used as test set to validate and assess the quality of the model. For this particular data set we used Autoregressive exogenous input (ARX) model. The standard structure of ARX model is as follows:

$$\text{ARX}(n,m,d)$$
$$y(k) = \sum_{i=0}^n a_i y(k-i) + \sum_{j=0}^m b_j u(k-d-j)$$

where  $n, m$  - order of the model,  $\max(n, m)$  is normally considered as the order.

$a_i, b_j$ - parameters of the model

$d$  - delay (time intervals taken to observe a change of input in the output).

Given the estimation set, least square regression algorithm [14] was used to decide the order, delay and the parameters of the model. We constructed ARX(1,1,0), ARX(1,1,1) and ARX(2,2,0) models and used the test set to validate the quality of the model. From the validation we can say that this system fit better to a second order ARX model having high predictive accuracy (97%). However, due to simplicity of design we used ARX (1,1,0) model to represent the system which had close to 96% fit with the test set. The model and the parameters are as follows:

$$y(k) = 0.8331y(k-1) + 0.6542u(k) \quad (1)$$

#### 4.2. Hammerstein-Wiener model construction

As the figure 1 illustrates, Hammerstein-Wiener model introduces two additional nonlinear blocks compared to single linear model construction process covered in section 4.1. This leads to the complexity of deciding the model structure for input and output nonlinear blocks as well as deciding parameter for selected structure. Hammerstein-Wiener model is useful when the considered nonlinearities are static (not time varying). In addition, if some information about these static nonlinearities (such as structure) is known at the design time we can simplify the model construction process.

We investigated the possible operating points and output of the system using set of experiments to check whether we can observe any static nonlinearity. This investigation yield interesting results.

The figure 3 shows all operating points depending on the requirements in section 3. When  $A$  and  $B$  are equally important and when they are sending same amount of workloads, the session allocation around 1 can be taken as the nominal operating point. Compared to the nominal operating point when session allocation for  $A$  increases, the operating points are indicated as  $A$  operating region. Similarly, when session allocation for  $B$  increases, the operating region is indicated by  $B$  operating region. It is clear that these operating regions are not similar. The gaps between the consecutive operating points in region  $A$  increases when it reaches the boundary of the operating range. In contrast, the gaps decrease in the  $B$  region. Such discontinuous/nonlinear operating points could drastically affect the performance of a linear controller. The problem is how to remove this known input nonlinearity in the system so that performance of a linear controller can be improved. This motivates us to use Hammerstein model to in-cooperate the known static nonlinearity as input nonlinearity in to the control system.

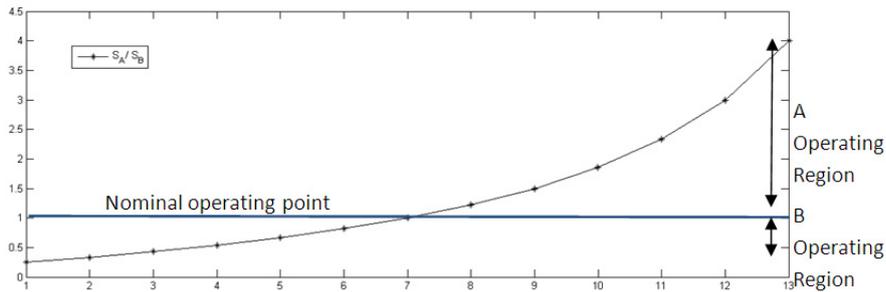


Figure 3: Possible operating points (control input)

The input nonlinear component of the model can be represented by different functions/model structures [18]. For instance, polynomial, piecewise linear, nonlinear-ARMA models, splines and neural-networks are such functions [16, 18, 26]. However, high emphasis has to be given to the selected function, because it is a necessary conditions that the selected function should have an inverse, to aid controller design process [26]. We used a polynomial function of order  $r$  to estimate the input nonlinearities. The specific advantage of using polynomial function is that, it is convenient to construct its

inverse function with sufficient fit [26]. Let us assume that un-measurable intermediate variable  $v$  takes the values of -6, -5, 4 ... -1, 0, 1 ... 5, 6. The main property in this selection is that  $v$  has equally spaced points in its range. Then using polynomial function  $f$  we try to approximate  $v$  using  $u$  (see equation 2). The curve fitting tool (cftool command in Matlab [27]) was used to construction of  $f$ . A polynomial of degree 4 was used to approximate  $v$  using  $u$ . It is always recommended in any model construction process to approximate a good fit while keeping the order of the polynomial as low as possible to avoid computational overhead [28]. Then the inverse function  $f^{-1}$  was approximated by interchanging the  $x$  and  $y$  parameters of the cftool command. That is range of  $v$  was taken as  $x$  data and  $u$  was taken as  $y$  data. The model  $f^{-1}$  was also approximated with sufficient fit by a polynomial of order 4 (See equation 3). The idea behind estimating inverse function of  $f$  is that it can be used to cancel off the existing (/imaginary) input nonlinearity. To check the quality of the function approximation we can use following step by step process.

- Calculate the output of  $f^{-1}$  using the range of  $v$  as the input. Let's call this ( $u'$ ).
- Then use  $u'$  in  $f$  to calculate the output. Let's call this ( $v'$ ).
- Compare the original input at step 1  $v$ , with  $v'$ . If their relationship looks approximately linear we can say that the approximation is good enough. Otherwise we have to redesign the function with different structures and parameters.

In the design process function  $f$  could be fit in to the model 2 with goodness of fit value of 0.9998. If we used a 5<sup>th</sup> order polynomial we would have achieved goodness of fit value of 1. However, due to additional computational burden we opted to used polynomial of degree 4. Similarly, the inverse function of  $f$  could be sufficiently fit in to a polynomial of degree 4 with a goodness of fit value of 0.9998. After estimating these functions the above step by step validation process was carried out to check the accuracy. The figure 4 shows the comparison of  $v$  and  $v'$ , which illustrates a sufficient linear characteristics.

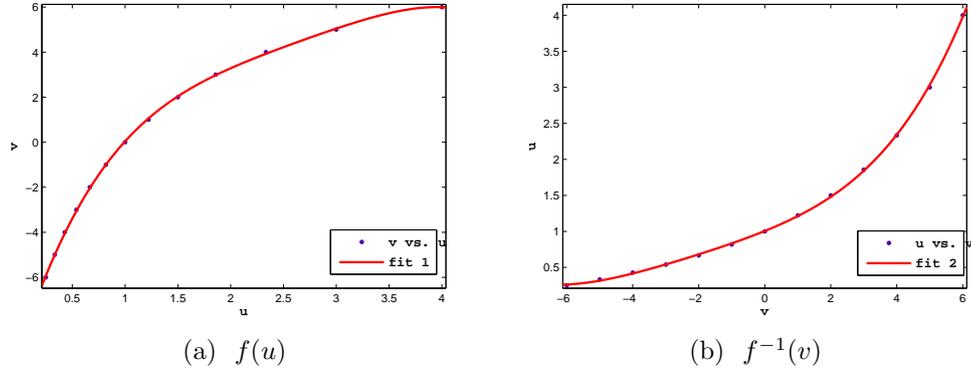


Figure 4: The data points and function approximation

$$\begin{aligned}
 v &= f(u) \\
 &= -0.1584 * u^4 + 1.696 * u^3 - 6.959 * u^2 + 14.63 * u - 9.17 \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 u &= f^{-1}(v) \\
 &= 0.0003828 * v^4 + 0.003445 * v^3 + 0.01722 * v^2 + 0.1857 * v + 1.006 \quad (3)
 \end{aligned}$$

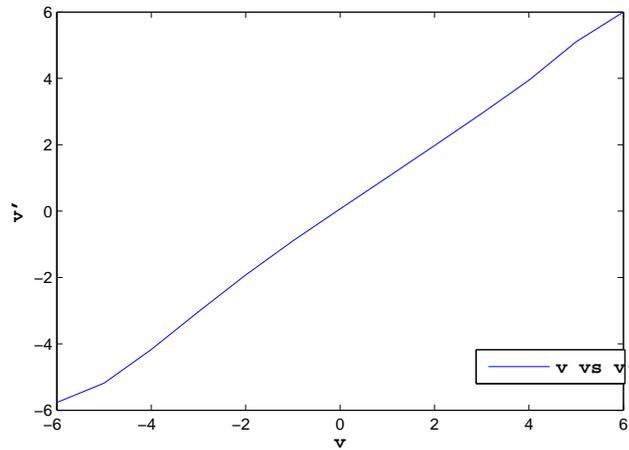


Figure 5: validation of estimated nonlinear functions ( $v$  vs  $v'$ )

Then assuming that this imaginary  $f$  function is a part of the system

model we integrated the inverse of it as a component in to the software system. It was integrated just before the actuator (the actuator accepts original control  $u$  as an input). After this step the transformed system operates with  $v$  instead of  $u$ .

The next step is to investigate whether there is static output nonlinearity in the system. We conducted several experiments with different control signals and observed the behavior of the output. Out of those experiments, the output for the sinusoidal control signal demonstrated the behavior in figure 6.

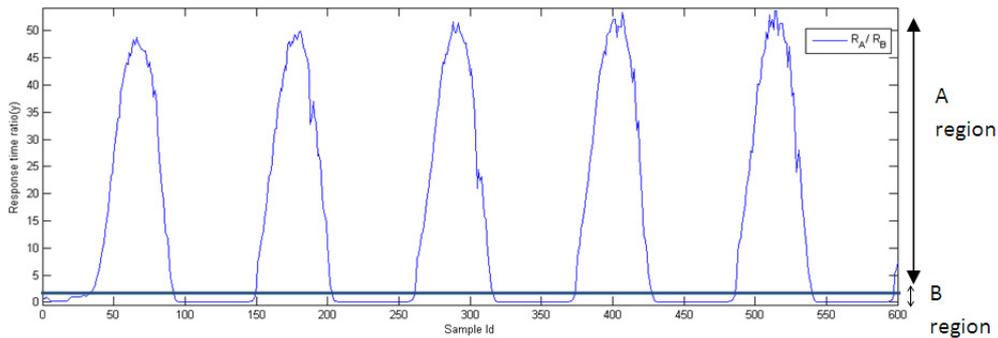


Figure 6: Output of the system for a sinusoidal input signal

Figure 6 shows that there is nonlinearity in the output as well. The range of values when system is operating in region A is drastically higher than B. That is the span of the measured output is wide in region A compared to region B. In addition, we noticed that such behavior is shown periodically in the data with very small noticeable deviations. This behavior indicated that the nonlinearity in the output is not drastically time varying. In other words it remains sufficiently static to be modeled as static output nonlinearity. The intuition behind this was to give the linear controller more equally spanned output to operate. With this idea in mind, we conducted several system identification experiments using different types of input signals. Basically, we used popular input signals that are used in control literature such as two types of Pseudo Binary Random signals (PBRs), sinusoidal signal and white noise [11]. The main concern in selecting an input signal for nonlinear system identification experiment is the signal should be rich enough to excite the static nonlinearities, and should not be over exciting to invoke other none static nonlinearities [16]. The PBRs signal generally considered as not

rich enough to excite nonlinearities of the system [16]. However, sinusoidal and white noise has been used extensively to approximate the nonlinearities due to their rich excitation (eg: [11]). Then the data gathered from these system identification experiments were divided in to estimate and test set in a similar fashion described earlier. To estimate the linear subsystem and nonlinear subsystem of Wiener model we used Matlab [27] command called *nlhw*, which provides flexibility to use deferent model structures for linear and nonlinear subsystems. In addition, it provides the option to select different types of nonlinear functions useful to represent the nonlinear subsystem of Hammerstein-Wiener model.

```
%ze - contains the data in the estimation set, zv- contain the data in test set
% 'unitgain' indicate we are not going to model Hammerstein model
% poly1d function describes a polynomial of order r
% eg : [n m d] indicates the order of the LTI component
M = nlhw(ze,[n m d], 'unitgain',poly1d('deg',r));
% evaluative the model respect to the test set data and provide the indication of accuracy
compare(zv,M);
```

Experiment settings are as follows:

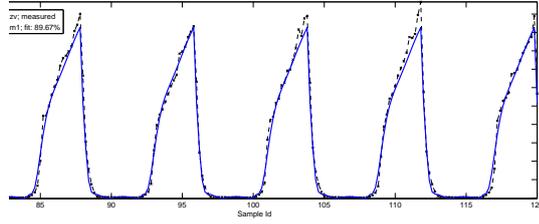
- For PBRS identification experiment we used the midpoint of operating range of A and B. The signal consists of  $v = 4$  and  $v = -4$  levels. We used two types of switching algorithms. The value was selected randomly and applied in the system every 5<sup>th</sup> sample (we call this as PBRS). Then in the other case we applied two values interchangeably every 20<sup>th</sup> sample time (we call this as PBS), which is a periodic square wave.
- For the sinusoidal experiment, input signal consists of all possible values of  $v$  changed in a sinusoidal manner with a period of 112 samples.
- For the white noise, input signal consists of randomly selected level from all possible values of  $v$  with a switching time of 4 time samples.

The model validation results for different model structures are shown in table 1. (n,m,d) represent the orders of linear complement and (r) represent the polynomial order of the nonlinear component of the Wiener model.

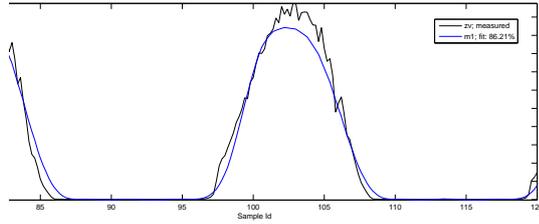
Table 1: Model validation results

(n,m,d,r)	PBRs	PBS	Sinusoidal	White noise
(1,1,0,3)	57.75	87.39	82.28	40.66
(1,1,1,3)	48.83	83.57	83.05	31.36
(2,2,0,3)	57.71	87.7	84.68	37.3
(1,1,0,4)	56.14	87.69	85.53	39.89
(1,1,1,4)	48.06	82.55	85.86	31.14
(2,2,0,4)	55.72	87.74	86.18	37.36
(1,1,0,5)	54.34	87.95	86.21*	39.93
(1,1,1,5)	47.8	82.52	86.51	30.18
(2,2,0,5)	53.7	87.74	86.29	37.46
(1,1,0,6)	53.29	89.07	86.07	39.96
(1,1,1,6)	47.42	82.59	86.43	28.07
(2,2,0,6)	52.99	87.29	86.1	37.48
(1,1,0,7)	52.63	89.67*	85.76	38.04
(1,1,1,7)	46.88	82.86	86.07	25.2
(2,2,0,7)	52.66	88.29	85.76	36.71

The model validation results indicate that, the data generated from PBS and sinusoidal signals fit to data better compared to the signals with random variations. The lower order ( $r = 3, 4$ ) polynomials provide less fit to data than the higher order polynomials ( $r = 5, 6$ ) in the case of PBS. In particular, the model with first order linear model and 7<sup>th</sup> order polynomial provided the highest fit with 89.67% accuracy. In many model settings first order model without delay fit with high accuracy compared to first order model with delay or the second order model. In sinusoidal signal based experiment, when the polynomial function with order 5 provides the highest accuracy compared to other model parameter settings. Overall, the lower order models have lower fit compared to higher order models. The experiments based on signals with random variations, fit to model data with drastically low fit. This could be because of control input signal is overly exciting so that the non-static nonlinearities are also excited. When this is the case fitting the data with static nonlinear function would amplify the other unmolded dynamics excited by white noise [11]. However, different types nonlinear functions (other than polynomials which we used in this case), may provide better fit than what we observed in the table 1. From the above results, it looks promising to



(a) PBS (1,1,0,7)



(b) SINE(1,1,0,5)

Figure 7: The model fit for a) PBS (1,1,0,7) b) SINE(1,1,0,5)

use models constructed with data from PBS (we refer to it PBS (1,1,0,7)) and Sinusoidal (we refer to it SINE(1,1,0,5)) order settings for the rest of the experiments. Figure 7 shows the validation results for the model fits.

After deciding on the model structure and parameters for the output non-linearity, next step is to estimate the inverse function. However, to estimate the inverse of the polynomial function, we need to calculate the corresponding data for intermediate variable  $w$ . To derive this data we used Matlab [27] command called `sim` to simulate the model. The figure 9 shows the code.

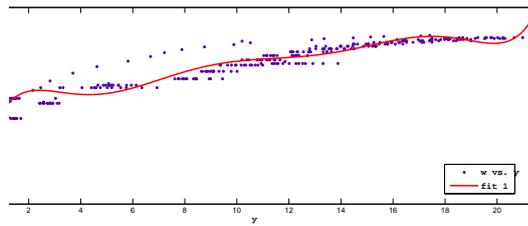
```
% M is the model object returned by nlhw function
% V contains the control signal data points
linModel = M.LinearModel;
w = sim(linModel, V);
```

After calculating data for intermediate variable  $w$ , `cftool` command was used to estimate the inverse function of  $y$  and  $w$ . The validation results for the model fits are shown in table 2.

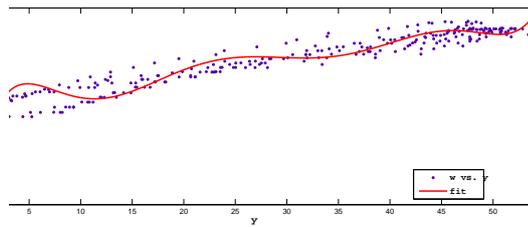
For both cases validation results did not improve drastically after the 7<sup>th</sup> order. To avoid over fitting and reduce computational burden we used poly-

Table 2: Validation results of the output inverse function

Order of Inverse polynomial	$g^{-1}(y)_{PBS}$	$g^{-1}(y)_{SINE}$
4	0.93	0.88
5	0.95	0.89
6	0.95	0.9
7	0.96	0.91



(a) PBS (1,1,0,7)



(b) SINE(1,1,0,5)

Figure 8: The model fit for inverse function - a) PBS (1,1,0,7) b) - SINE(1,1,0,5)

nomial with 7<sup>th</sup> order model in PBS and sinusoidal based inverse function. The respective functions are as follows:

$$\begin{aligned}
w &= g^{-1}(y)_{PBS} \\
&= 1.019e^{-005} * y^7 - 0.0007953 * y^6 + 0.02493 * y^5 - 0.4004 * y^4 + \\
&3.479 * y^3 - 15.78 * y^2 + 34.02 * y - 20.06
\end{aligned} \tag{4}$$

$$\begin{aligned}
w &= g^{-1}(y)_{SINE} \\
&= 6.054e^{-008} * y^7 - 1.212e^{-005} * y^6 + 0.00097 * y^5 - 0.03944 * y^4 + \\
&0.855 * y^3 - 9.437 * y^2 + 46.76 * y - 53.40
\end{aligned} \tag{5}$$

The model fit for these inverse function approximations are shown in figure 8.

Then for each case we integrated this estimated inverse output nonlinear function as a component in the system following the sensor. The sensor provides the original output of the system, which is the input to the output nonlinear component. Figure 9 shows the structure of the system after the integration of inverse input and output nonlinearity functions.

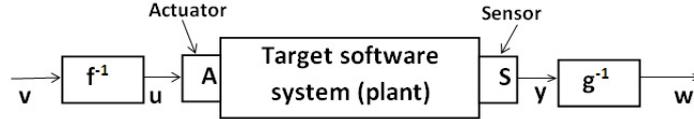


Figure 9: Structure of the software system after integrating the inverse function

Then the main question is what about the linear subsystem of the Hammerstein-Weiner model. This model is important for analyzing the behavior of the system and controller design purposes. We already decided on first order ARX model to capture the dynamics of this linear subsystem, when approximating the nonlinear function for Wiener model. The linear component model derived during Wiener model estimation is as follows:

$$PBS : w(t + 1) = 0.84w(t) + v(t) \tag{6}$$

$$SINE : w(t + 1) = 0.96w(t) + v(t) \tag{7}$$

The initial controller design and experiment attempts using these models showed significant performance issues. The reason could be the input signals

that were used to construct the Wiener model were not sufficiently exciting to capture the other dynamics of the system. These unmolded dynamics may have affected the performance of the controller in those experiments to affect the performance of the linear controller. To rectify this issue, we did another white noise input signal based system identification experiment with the system structure shown in figure 9. We did two separate identification experiments replacing  $g^{-1}$ , with  $g^{-1}(y)_{PBS}$  and  $g^{-1}(y)_{SINE}$ . The equations 8 and 9, shows the ARX models for the linear component of Hammerstein-Weiner model and the validation.

$$PBS : w(t + 1) = 0.82w(t) + 1.1v(t) \quad \text{fit : 14.1\%} \quad (8)$$

$$SINE : w(t + 1) = 0.81w(t) + 2.39v(t) \quad \text{fit : 92.1\%} \quad (9)$$

The fit of the PBRs case was quite low because there was a one large spike in the predicted values from the model in equation 8. It generated a large prediction error in model validation process. Due to this spike the model fit was quite low. From the observation of data we realize that this large spike is due to a value that has not captured in PBS experiment based Weiner modeling. Such unseen values provide drastically different values when the inverse nonlinear function is plugged in to the system. However, we believe that careful selection of switching period and levels of PBS could solve such large spikes.

If we compare the equations 6 and 7, we see that there is a drastic difference in the parameters of the model especially, in the case of sinusoidal. Such inaccuracies may have affected the performance of the controller as we discussed earlier.

To check the quality of the function estimation we can take the approach similar to the case of input nonlinearity.

- From the identification data simulate the model (eg: equation (8), (9)) and calculate the  $w'$
- Using observed measured output calculate  $w''$  using inverse of output nonlinearity.
- Compare  $w'$  and  $w''$ . If they are linearly distributed the quality of the estimation is fair enough. (see Figure 10)

Using the same data gathered from experiments described in this section, we constructed Hammerstein and Winner models separately for prototypical system.

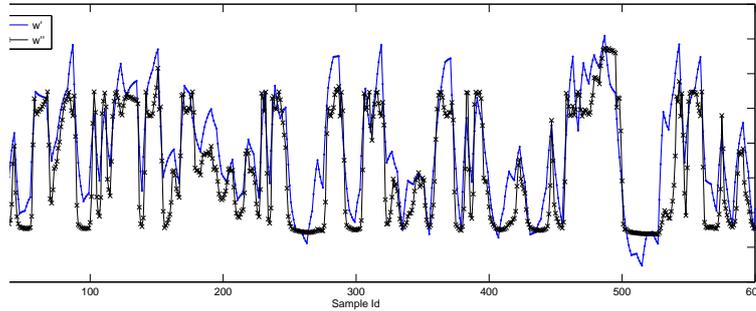


Figure 10: Validation of output nonlinearity case of SINE. ( $w'$  vs  $w''$ )

#### 4.3. Hammerstein model construction

The data gathered from the final system identification experiment was used for this purpose. We created estimation and test sets with  $v$  and  $y$  variables. Using the estimation set we used a first order ARX model to fit the data. The equation 10 shows the model, which was fit to data with 96% accuracy.

$$y(t + 1) = 0.91y(t) + 0.31v(t) \quad (10)$$

#### 4.4. Wiener model construction

To construct the Wiener model for the prototypical system, we utilized the data gathered with the sinusoidal signal of experimented discussed in section 4.2. Then  $u$  and  $y$  data values were fit to a 5<sup>th</sup> order polynomial and first order ARX model similar to the procedure illustrated in section 4.2. It was fit with 81% accuracy. However, the model fit was not that high compared to the Hammerstein-Weiner model constructions. This could be because of the discontinuous input  $u$ , used in the individual Wiener model. Then the inverse of the nonlinear function was derived using Matlab code given in section 4.2. The inverse function fit in to 5<sup>th</sup> order polynomial with 87% of fit. The inverse function and the data gathered from final white noise experiment in the Hammerstein-Weiner model construction process in section 4.2, was used to derive the linear model. First using the inverse nonlinear function, the intermediate variable  $w$  was calculated. Then the data points in white noise signal  $u$  and calculated  $w$  was used to construct

the linear component of the Weiner model. The model details are as follows:  
 Linear model:

$$\begin{aligned}
 w &= g^{-1}(y)_{Weiner} \\
 &= 8.33e^{-007}y^5 - 0.0001256y^4 + 0.006973y^3 - 0.1798y^2 \\
 &\quad + 2.879y + 24.37
 \end{aligned} \tag{11}$$

Linear model:

$$w(t + 1) = 0.92w(t) + 1.52u(t) \tag{12}$$

After these modeling procedures, next step is to understand the systematic process of designing a controller for this particular system so that self-managing objectives can be achieved. For this purpose we hope to use model predictive control. In the next section we provide the systematic process of designing a model predictive controller for Hammerstein-Weiner system.

## 5. Model predictive control design

### 5.1. Linear Model predictive control

The model predictive control is a technique which tries to optimize the future behavior of the system using the predictions from a dynamic model of a system [20, 21, 29]. The model predictive control has proven to be useful to control many physical plants due to the capability of constraints handling and integrating multiple variables. In MPC, the dynamic model of the system is used to predict the future behavior of the system. These predictions are compared with the reference signal (set point/control objective) to check whether there are any deviations (control errors). Depending on the error sequence, an optimal control input sequence is derived by optimizing the cost function  $J(k)$  (see equation 13). However, only first element of the calculated input sequence is applied in the system, rest of the sequence is discarded. This concept is called as receding horizon principle [21]. The

main variables governing the controller is as follows:

$$J(k) = (R_s - Y)Q(R_s - Y)^T + \Delta U R \Delta U^T \quad (13)$$

$R_s$  = the reference trajectory

$$Y = [y(k+1|k)y(k+2|k)\dots y(k+N_p|k)]^T$$

contains predictions at  $k^{th}$  time instance

$$\Delta U = [\Delta u(k)\Delta u(k+1)\dots \Delta u(k+N_c-1)]^T$$

$$u(k) = u(k-1) + \Delta u(k), \quad (14)$$

where  $u(k)$  is the actuation for the current time interval.

$R = r_w I$ , where  $r_w$  indicates the control effort

$Q$  = weight vector to indicate the trust on the future predictions.

To tune the performance and computational overhead of the controller, there are several tuning parameters integrated to the control design. The prediction horizon  $N_p$ , control horizon  $N_c$ ,  $r_w$  and  $Q$  can be used for this purpose. If the dynamic model of the system is trustworthy and system is not highly time varying  $N_p$  can be fixed to be large value. However, large prediction horizon imposes high computational overhead [26]. Similarly, large values for  $N_c$  increase the amount of computation.  $r_w$  parameter is useful to adjust the control effort by the controller. Use of large values reduces controller effort or aggressiveness, while low values increases the aggressiveness. In general all these parameters can be used to adjust the performance, but it is convenient to fix values of several of them and adjust desirable parameter to tune the controller. In addition, the model predictive control is useful because we can incorporate constraints in to the optimization process which is solved at runtime. The constraints always exist in the real physical systems due to limited range of input. For instance, in the prototypical system we only have 20 sessions to be allocated among different client classes, depending on the workload disturbances. The constraints fall under soft or hard constraints. The hard constraints are involved with control input because we cannot exceed the limits at any cost. However, other constraints on measured output and state space variable can be relaxed to improve the performance of the controller. Typical constraints are as follows:

$$\begin{aligned}
& \text{Minimize } J(k) \\
& \text{Subject to :} \\
& u_{min} \leq u(k) \leq u_{max} \\
& \Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max} \\
& y_{min} \leq y(k) \leq y_{max}
\end{aligned}$$

This constraint problem is solved online using standard quadratic programming [21]. We used the general predictive controller for the linear model constructed in section 4.1.

### 5.2. Hammerstein-Weiner Model predictive control

From the section 4, the linear component of the Hammerstein-Wiener model can be generally represented as :  $w(t + 1) = aw(t) + bv(t)$  , where the original variables of MPC,  $u(t)$  and  $y(t)$  are replaced by the intermediate variables  $v(t)$  and  $w(t)$  respectively. With this, the cost function can be represented with the transformed. The transformed controller is as follows:

$$J(k)_{(HW)} = (R_{s(HW)} - W)Q_{(HW)}(R_{s(HW)} - W)^T + \Delta V R_{(HW)} \Delta V^T \quad (15)$$

$$R_{s(HW)} = g^{-1}(R_s) \quad (16)$$

$$w(k) = g^{-1}(y(k)), y(k) \text{ is taken from the sensor.} \quad (17)$$

$$W = [w(k + 1|k)w(k + 2|k) \dots w(k + N_p|k)]^T$$

contains predictions at  $k^{th}$  time instance

$$\Delta V = [\Delta v(k)\Delta v(k + 1) \dots \Delta v(k + N_c - 1)]^T \quad (18)$$

$$v(k) = v(k - 1) + \Delta v(k), \quad (19)$$

$$u(k) = f^{-1}(v(k)), \quad (20)$$

where  $u(k)$  is the actuation for the current time interval. (21)

$$R_{(HW)} = r_{w(HW)}I \text{ where } r_{w(HW)} \text{ indicates the control effort}$$

$$Q_{(HW)} = \text{weight vector to indicate the trust on the future predictions}$$

Similar to original cost function of the MPC problem discussed in section 5.1, here the variables are transformed. The controller operates with the intermediate variables  $v(t)$  and  $w(t)$ . The measured output for the controller in the current time sample is calculated by equation 17, using the sensor

information of the original system. Similarly, equation 20 is used to calculate the original control output of the system before actuation. Other important alteration is reference signal or the set point ( $R_s$  to  $R_{s(HW)}$ ). The reference signal has to be filtered through the inverse output nonlinear function to be transformed in to intermediate variable. The tuning parameter selection process does not change, however they have to be selected analyzing the model of linear component of Hammerstein-Wiener model. Let us try to transform the constraints as well. Now, the limits of the control input and the measured output has to be transformed in an effective manner so that the constraints of the original problem can be preserved indirectly. This transformation is as follows:

$$\begin{aligned} & \text{Minimize } J(k)_{(HW)} \\ & \text{Subject to :} \\ & f(u_{min}) \leq v(k) \leq f(u_{max}) \\ & \Delta v_{min} \leq \Delta v(k) \leq \Delta v_{max} \\ & g^{-1}(y_{min}) \leq w(k) \leq g^{-1}(y_{max}) \end{aligned}$$

The original constraint problem is now transformed in to intermediate variables based constraint problem. We used inverse nonlinear input and output functions. The upper limit and the lower limit of rate of change of control input, has to be decided depending on the selection of range of  $v$ . For the prototypical system we used variable  $v$  to range from -6 to 6 with equal gaps of 1. So if its desirable we can use  $-1 \leq \Delta v(k) \leq 1$  as a constraint. Form the above transformation process we can see that the MPC algorithm and the constraint problem have not changed drastically. The final control system takes the structure shown in figure 11.

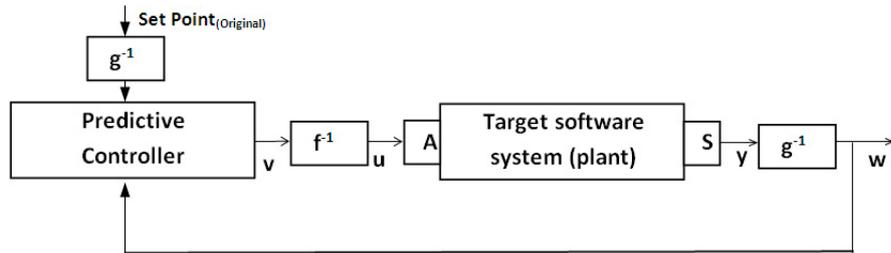


Figure 11: Block diagram of the software system after integrating the inverse function

Using the model predictive controller described in this section we provide comparative study on the performance of the linear and nonlinear model predictive controllers.

## 6. Experiment results

In this section we utilize the models and the controllers designed in previous sections to investigate their performance in a relative guarantee control system. In particular, linear model, Hammerstein model, Wiener model and Hammerstein-Wiener model based predictive controllers will be evaluated. Further we show the performance of the fixed controller to motivate the requirements of self-managing capabilities in complex software systems. For all these experiments we use analytical (synthetic) workload settings showed in figure 12. We opted to use analytic workload instead of trace base workloads (workloads recorded from a real system) because it is easy to understand the behavior of the control system when it faces the workloads with interested characteristics. At 30<sup>th</sup> sample *A* workload increases from 10 requests/sec to 30. This could be a case where travel agent *A* has advertised special travel plan/fares for a limited amount of time, so that there is a sudden increase of workloads. Then at 80<sup>th</sup> sample the *A* workload reduce to nominal request rate. Then at 100<sup>th</sup> sample *B* workload increases to 30 requests/sec from 10 requests/sec while *A* remain at 10 requests/sec. In addition, we set the reference value (set point) to 1, indicating that both client classes are equally important for the initial experiments.

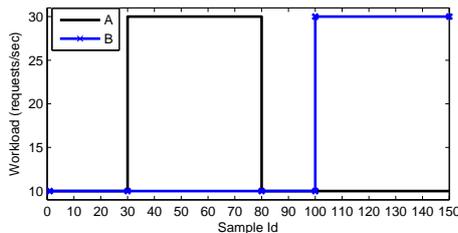


Figure 12: Workload settings

Firstly, we evaluate the performance of the fixed controller, which does not take the feedback into account to make decisions. It does not make any session allocation decisions at runtime. At design time, depending on the previous knowledge about the workloads and the system behavior, resource

demand is forecasted and it is hardcoded in the system as the allocation for A and B. Since the set point for the experiments is 1, we implemented equal allocations for two classes ( $S_a = 10$  and  $S_b = 10$ ). Figure 13 shows the performance of a fixed control.

The performance is satisfactory till the 30<sup>th</sup> sample, however it fails to achieve the set point afterwards when either A or B workloads increased over the nominal operating region. From 30<sup>th</sup> sample period to 80<sup>th</sup> we can see that B starves for resources while A having lot more than required. Similar behavior occurs after 100<sup>th</sup> sample time. This justify that fixed allocation is really inefficient for the systems that face unpredictable and time varying workloads. These reasons lead to the importance of having adaptive decision making mechanism at runtime depending on the workload fluctuations.

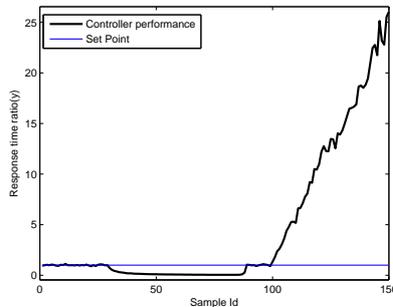


Figure 13: Performance of the fixed controller

Then we show the performance of the model predictive controllers based on different types of models discussed so far. In section 5, we mentioned about the different tuning parameters of the model predictive controller. For our convenience we set many parameters to fixed values and used  $r_w$  parameter as the main tuning variable in the controller. The change of the values of this parameter changes the pole locations of the close loop system, so that the performance specifications such as stability, overshooting, and settling time can be analyzed effectively. The table 3 shows the values of these variables that were fixed for all experiments discussed in this section. In addition, we formulated the constraint problem to avoid control input deviating out of range (i.e.  $u_{max} = 4, u_{min} = 0.25$ ). That is the cost function was optimized with hard constraints. We did not use any soft constraints for these experiments.

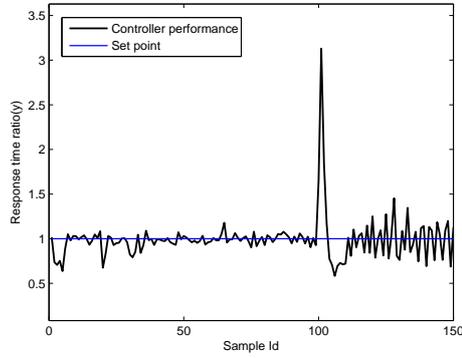
Table 3: Parameters of MPC for the experiment

Parameter	Value
$N_p$	5
$N_c$	20
Q	Identity matrix
$R_s$	1

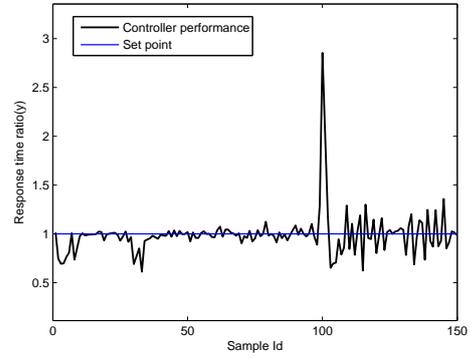
Firstly, we investigate the performance of the single linear model based predictive controller performance. The model construction and details were discussed in section 4.1. We use the model in equation 1 for this purpose. Figure 14 shows the performance comparison for different values of tuning parameter  $r_w$ .

The general observation from figure 14 is that, when a disturbance occurs (i.e at  $30^{th}$  and  $100^{th}$  samples), there is a overshooting in the measured output but it eventually settles down illustrating the disturbance rejection capabilities of the predictive controller. However, what is interesting to note here is the steady state behavior of the control system after a disturbance. When the workload of A increases at  $30^{th}$  sample, performance deviates from the objective value and settles down with low steady state error. In contrast, when workload of B increases at  $100^{th}$  sample steady state error is comparatively high after settling down. This behavior is evident in most of the cases shown in figure 14. The issue of steady state error is less evident when  $r_w = 5$  and 10, but the settling time and the overshooting at the  $30^{th}$  sample has drastically increased. In addition, the startup performance has degraded as well. Theoretically, when the  $r_w$  is small the poles of the close loop system get close to the origin of the complex plane (for more details refer [21]). This means gain of the predictive controller increases hence, the controller becomes more aggressive towards disturbances. Thus, the increase of settling time at large values of  $r_w$  ( $= 5$  and 10) can be explained. The steady state behavior after  $100^{th}$  sample is satisfactory when  $r_w = 5$  and 10 cases indicates that controller is less vulnerable for noisy workload conditions due to low aggressiveness.

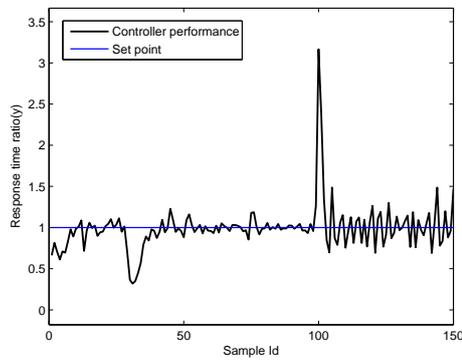
From the analysis of these results it is difficult to select the control tuning parameter (or controller gains) to achieve satisfactory performance in either A or B regions. If we use high gains controller performance in B region degrades due to large steady state errors and oscillatory behavior. This issue



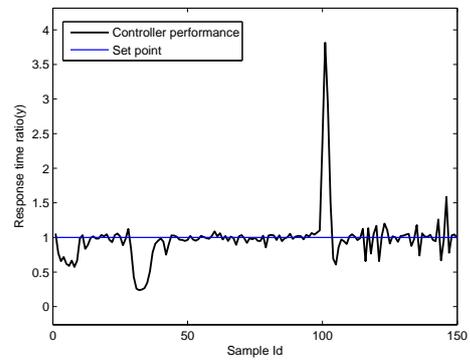
(a)  $r_w = 0$



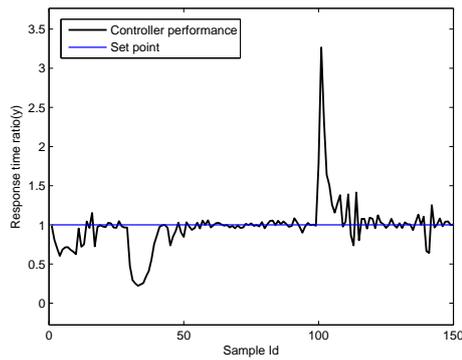
(b)  $r_w = 0.1$



(c)  $r_w = 1$



(d)  $r_w = 5$



(e)  $r_w = 10$

Figure 14: Performance of the controller with the linear MPC

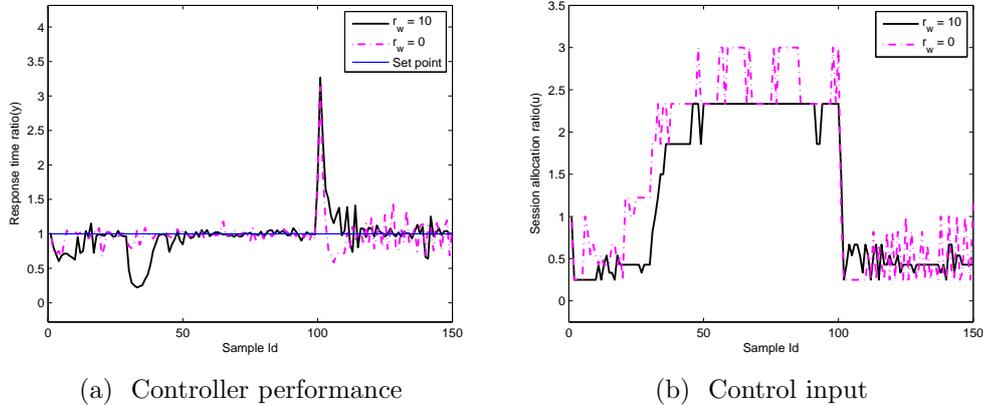


Figure 15: Controller performance and input variations for  $r_w = 0$  and 10

occurs because of the discontinuous, unequal spacing of the control input in the region of B. Due to this reason an aggressive controller cannot settle to a desirable operating point. Even if it settles to a point for a short while, the noisy workloads tend to affect the performance of the controller. In contrast, if we use low gains, performance of region B can be improved however settling time and overshooting increases for A region showing the lack of aggressiveness of the controller. Figure 15 shows a comparison of the performance and control input variations for a highly aggressive controller ( $r_w = 0$ ) and less aggressive controller ( $r_w = 10$ ). Figure 15b) indicates that control input becomes really oscillatory in the case of  $r_w = 0$ , compared to  $r_w = 10$  after  $100^{th}$  sample, which lead to high steady state error in  $r_w = 0$  case. At the same time, Figure 15b) shows that, it takes comparatively large amount of time (13 samples more) for the  $r_w = 10$  based controller to settle to the desired operating point when the system encounter high A workload at  $30^{th}$  sample.

This analysis shows that control system designer will face difficulties in selecting desirable values for the tuning parameters for this relative guarantee control system. In addition, to general performance specific matrices of the control system, discriminative behavior for each client class has to be considered carefully, before selecting a appropriate controller parameters (gains).

Now, we investigate performance of the control system when the Hammerstein model constructed in section 4.3, is used in predictive controllers. The

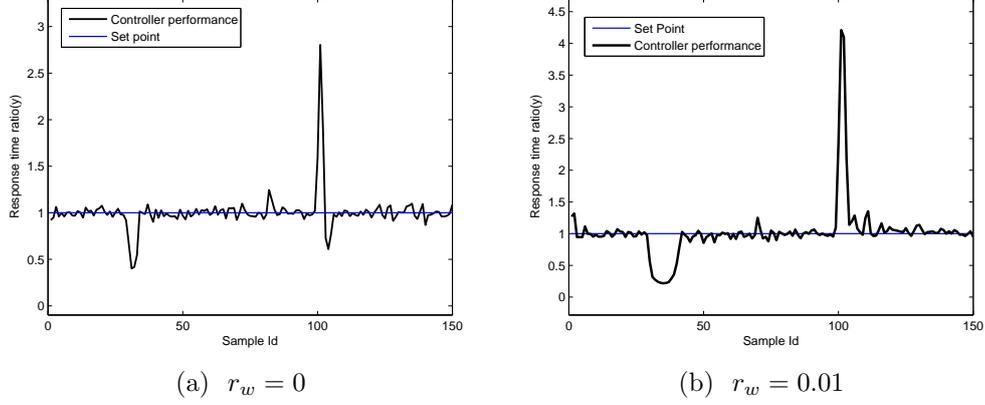


Figure 16: Hammerstein model based controller performance of for a)  $r_w = 0$  and b)  $r_w = 0.01$

control system structure was not illustrated specifically for Hammerstein model based control. However, the structure is similar to figure 11, only difference is that the  $g^{-1}$  component is removed from the control system. Figure 16 shows the performance of the controller for two cases of tuning parameters.

The main noticeable improvement is that oscillatory behavior when the system is operating in region B has vanished. The steady state behavior after disturbances at  $30^{th}$  and  $100^{th}$  sample periods is similar for both client classes. This is already a drastic improvement from the linear model based controller. It is because the predictive controller is now operating with the transformed input variable  $v$ , instead of original variable  $u$ . This transformed variable gives the controller an input with equal spacing, which gives the predictive controller more flexibility to settle down to a desired operating point even with noisy workloads. So that the variation of the transformed control input ( $v$ ) shows similar behavior in both A and B regions. From figure 17, we can see that the fluctuations of the original input variable after the  $30^{th}$  sample (when system operates in region A), looks the same for linear model and Hammerstein model cases. In contrast, we can see that in the case of Hammerstein model the original control input does not show oscillatory behavior compared to linear model case after the  $100^{th}$  sample. Hence, we see this performance improvement of the controller. Another noticeable property is, the overshooting and settling time has improved a lot when we use more

aggressive controller ( $r_w = 0$ ). In addition, when selecting the gains we have selected very low values for the tuning parameters otherwise settling time and overshooting increases drastically. It is evident in figure 16b), even if we increase  $r_w$  from a small value like 0.01, the overshooting increases close to 1.5 times compared to  $r_w = 0$  case. We can see similar increments in settling time.

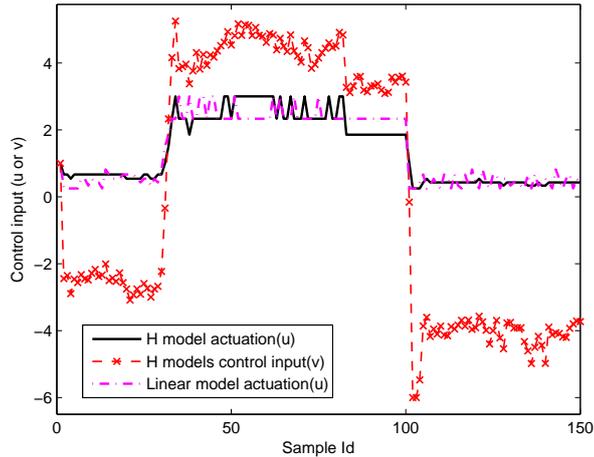


Figure 17: Comparison of original control input ( $u$ ) and transformed input ( $v$ ) for linear model (case of  $r_w = 1$ ) and Hammerstein model (case of  $r_w = 0$ )

Then we used the Wiener model, which was constructed in section 4.4 as the model of the predictive controller. For these experiments the  $f^{-1}$  component of the control system was removed and  $g^{-1}$  is replaced by equation 11. The performance of the model is shown in figure 18.

From figure 18a), the performance of the control system is satisfactory till the 40<sup>th</sup> sample period and then it becomes really oscillatory when the controller is operating in both regions. This gives us the indication that the Wiener model performance satisfactorily when it operating with nominal workloads settings. It is also noticeable that the steady state error in the region of A is less than the region of B. If we look at figure 18b) the reason behind such oscillatory behavior is saturation of the control input. However, the most interesting thing is the overshooting and the settling time. Compared to figure 16, we can see that overshooting and settling time is drastically low after a disturbance. This behavior may be because of the

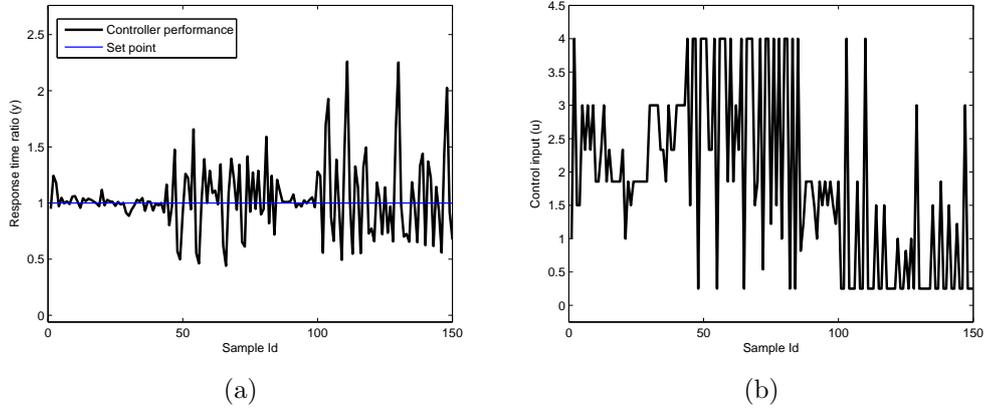


Figure 18: a) Performance of the Wiener model (case of  $r_w = 100$ ) b) control input ( $u$ )

existing input nonlinearity amplifies when the output is linearized. In addition, we believe that the model fit is not sufficiently accurate compared to the case of Hammerstein model and Hammerstein-Weiner model, so that relatively inaccurate static nonlinearities and the model could affect the performance of the control system.

Next, we investigated the performance of Hammerstein-Wiener model, which incorporates both input and output nonlinearities in the system model.

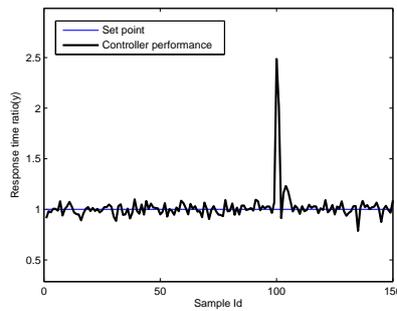


Figure 19: Hammerstein-Wiener model based predictive controller performance ( $r_w = 50$ )

Figure 19 shows the best performance we saw thus far in this discussion. The high steady state error issue when the system is operating in region of B

is not seen in the controller performance. In addition, the overshooting and settling time has drastically reduced compared to Hammerstein model based controller performance (figure 16). That is 21% reduction of overshooting at 100<sup>th</sup> sample, compared to Hammerstein model based controller performance. The setting for tuning parameter had to be set to very high value to achieve this performance. The reason is zeros of the models (see equation  $\text{refLmsinefinal}$ ) lie well outside the unit circle. To adjust the gains of the controller and place the close loop poles in desired locations,  $r_w$  had to be set to high values. If we set  $r_w$  at low values, still we can achieve considerable performance but controller becomes more vulnerable for noisy workload conditions and disturbances.

Figure 20 shows the performance and control input signal comparison of linear model based predictive controller and two nonlinear model based controllers, we discussed so far. We did not include the performance of Wiener model, because the performance was not that satisfactory compared to other models. The Hammerstein-Wiener model clearly outperforms linear and Hammerstein model based controller performance. From figure 20b), we can see that at the start the different models settle down into different operating points. When the workloads are low at 10 requests/second by each client class, there are wide ranges of operating points the controller can settle to achieve the same control objective. Then at 30<sup>th</sup> sample with A workload increases all controllers try to allocate more resources to A. However, Hammerstein-Wiener model reaches the desired range of operating points quickly than other models. All controllers show similar performance characteristics in that region, till the B workload increase at 100<sup>th</sup> sample. Hammerstein-Wiener model reaches the desired range of operating points 2 sample intervals before other models (100<sup>th</sup> vs 102<sup>nd</sup>), showing rapid response to disturbances. In addition, it shows the performance characteristics of the Hammerstein model embedded, because it avoids the oscillatory behavior when the system is operating in region of B, compared to the linear model. In addition, it shows the qualities of embedded Wiener model with low settling times and overshooting.

Figure 21 illustrates the output signal (intermediate variable -  $w$ ) of linear component of the Hammerstein-Wiener model. It shows similar behavior as the original measured output ( $y$ ), because  $w$  is calculated by sending the original measured output through the static inverse nonlinear function filter (see equation 11). The set point is  $w = -15.2605$  (calculated using equation

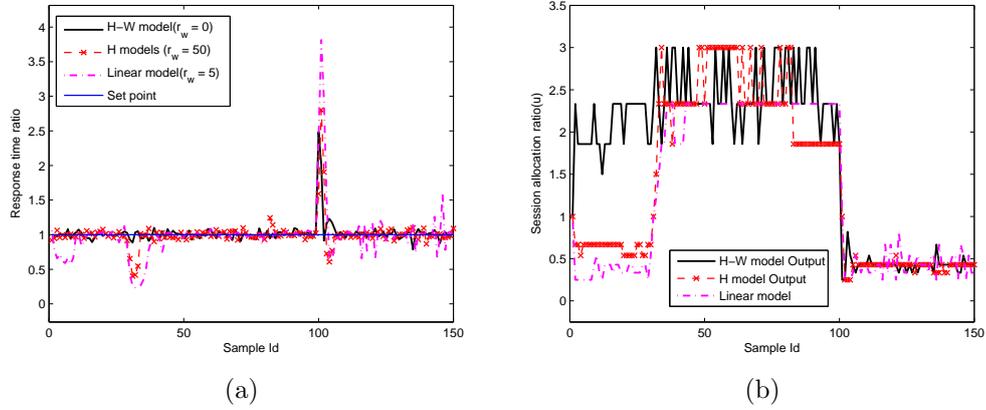


Figure 20: a )Performance comparison of Linear model, Hammerstein model (H), Hammerstein-Wiener model (H-W) , b) controller output comparison

16). The model predictive controller tries to achieve this set point, which indirectly leads to achieve the performance objectives of the original system. Incorporating known static input nonlinearities and the estimated output nonlinearities in the system model and using it for control purposes, consequently achieves superior performance compared to original/conventional control system design.

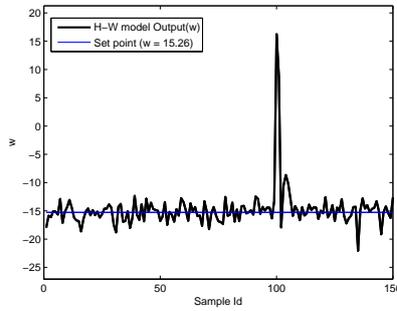


Figure 21: The intermediate variable  $w$  of the Hammerstein-Wiener model (H-W)

We conducted set of other experiment to make this study more comprehensive. The tuning parameters that provided the best performance was used in the predictive controller (Linear model -  $r_w = 1$ , Hammerstein-Wiener

model  $r_w = 50$ ). Firstly, we evaluated how these models perform in the nominal operating region. We changed the workload settings for these set of experiments, where A and B start off with sending 10 requests/second each, till the 50<sup>th</sup> sample and then both classes increase their workloads to 20 requests/second afterwards. The performance in the first 100 samples is shown in the figure 22. This shows that the linear model provides better performance even with large workload changes in the nominal operating region. Similar performance can be achieved with Hammerstein-Wiener model based predictive control. However, starting performance is better in Hammerstein-Wiener model based predictive control. This shows that nonlinear models are capable to represent the nominal operating region behavior equally well as the linear model. However, nonlinear models are superior in the sense of describing the global behavior of the system.

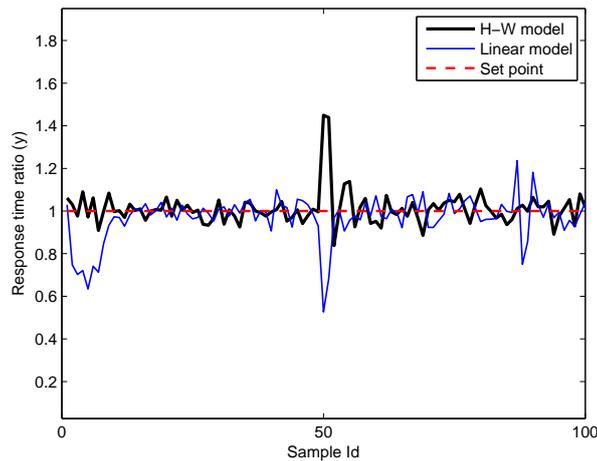


Figure 22: Performance of Linear model and Hammerstein-Wiener model in nominal operating region

Secondly, we investigated the performance comparisons when the system is overloaded. On average server software system can handle 40-45 requests/second without overloading. When the system is overloaded the performance of the system can degrade because the arrival rate and the service rate of the system may have large mismatch. If this is the case desired performance is hard to achieve whichever the control mechanism we use. The normal practice is to implement some kind of admission control policy in the

server system so that when system reaches its capacity the requests can be rejected [6, 7]. However, if the system is marginally overloaded a control system can still achieve the performance objectives. In these cases the estimated nonlinearities and the linear models may have drastic mismatches with the system. The main concern is the stability of the model when system encounter unbounded disturbance. In addition, actuator saturation happens more often, making the system to oscillate around the set point. However, assuming the estimated linear models and nonlinear functions are sufficiently accurate, we compare the performance of the controllers we designed in the marginally overloaded circumstances. The workload setting was similar to the workload settings in figure 12. The main difference is to overload we increased the workload of A to be 35 requests/sec at 30<sup>th</sup> sample and B to be 35 requests/sec at 100<sup>th</sup> sample. Clearly the overshooting and settling time performance specifications are better in Hammerstein-Wiener model compared to the linear model (see figure 23). The oscillatory behavior is shown in both models however, Hammerstein-Wiener model shows high oscillations in region A because of its aggressiveness. The gain of the linear model based predictive controller is less aggressive in the region of A, so it shows fewer oscillations in that regions. Similar, behavior can be seen after 100<sup>th</sup> sample, both models showing equal oscillations. It is hard to differentiate which controller performs better in this marginally overloaded case, because of uneven performance in the linear controller. However, we can see that Hammerstein-Wiener model provide similar type of performance for both operating regions and low overshooting and settling time performance specifications compared to the linear model.

Further, we conducted experiments to check the performance of the linear model and the nonlinear models with different set points. For the experiments described above, we treated client class A and B as equally important. From the experimental results, it indicates that resource allocation can be effectively done to achieve the set point equal 1. However, relative guarantee control scheme can be used when the performance requirements of different client classes are different. For instance, if B client class needs higher response time than A. We can set the set point to greater than 1 ( eg: 2, 3). If set point is 2, the controller will try to maintain the response time of B twice as much as of A. Similarly, we can give priority to B by having set point less than 1(e.g. : 0.5 ,0,25). To calculate the set point, response time requirements requested in the SLA can be used, but it could depend on other concerns as well. The decision of the set point must be taken by

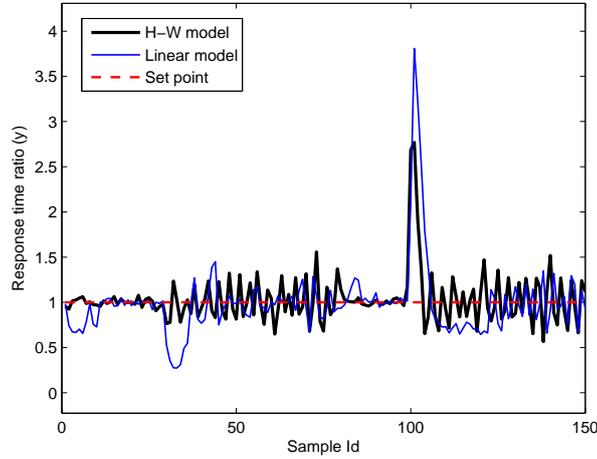


Figure 23: Performance of Linear model and Hammerstein-Wiener model when system is marginally overloaded

the control system designer considering the application settings and requirements of the software system. To show the performance of the models we conducted two experiments setting reference to 3 and 0.5. For the first case we applied 10 and 30 requests/second for A and B respectively, where as in the second experiment we set the workloads 30 and 10 requests/second for A and B respectively. Figure 24 shows the experiment results.

From figure 24a) it is clear that similar behavior we saw earlier experiments is evident when set points are set to different values. When the system is given an objective to maintain response time of B thrice as of A, performance of the linear model becomes highly oscillatory. The Hammerstein-Wiener model provides much better control by achieving on average, the desired reference value. Then in figure 24b) we can see that the settling time issue at the beginning till 30<sup>th</sup> sample, but it settles down after wards showing similar performance to Hammerstein-Wiener model. These set of experiments indicates that the Hammerstein-Wiener model can also perform well in different set points.

## 7. Discussion

In summary, the startup performance of the linear model is comparatively deviates from the control objectives compared to both nonlinear mod-

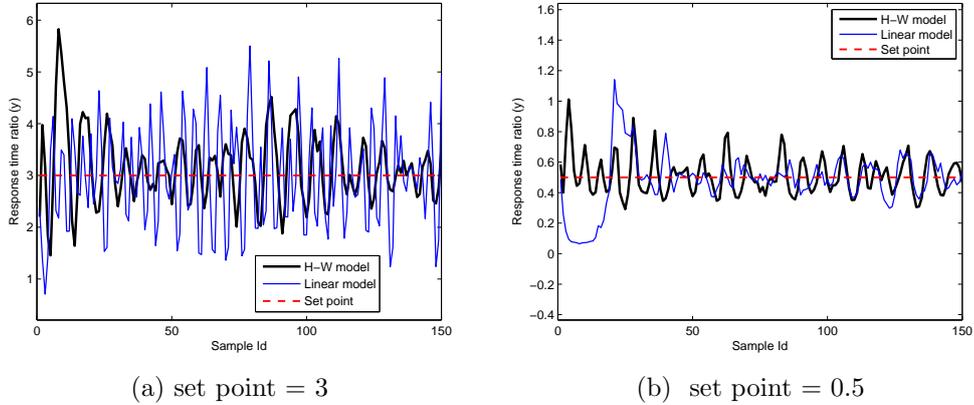


Figure 24: Performance of Linear model and Hammerstein-Wiener model with set point at a) 3 and b) 0.5

Table 4: Statistical comparison of performance

Statistical Parameter	Linear ( $r_w = 5$ )	H ( $r_w = 0$ )	H-W ( $r_w = 50$ )
SSE	19.8	6.05	3.71
Max	3.81	2.8	2.49
Min	0.24	0.4	0.79

els. Similarly, when the system face a disturbance linear model takes more time to settle down with high overshooting compared to nonlinear models. In addition, when linear model is operating in region of B (away from the nominal operating region) the performance becomes oscillatory with high steady state error. This issue poses difficulties in selecting the tuning parameters for the predictive controller. The control system designer has to trade-off the performance between client classes to achieve the desired performance. The performance of the nonlinear models looks drastically better in above concerns. However, Hammerstein-Wiener model provides high disturbance rejection characteristics compared to the Hammerstein model due to the integration of output nonlinearities in the model.

Table 4 gives a performance comparison with statistical operators. We used sum of squares, Maximum and minimum values of the measured output as statistical operators in this comparison. It is clear from this comparison that Hammerstein-Wiener model outperform other two models with fair distance.

There are many reasons to see such improvements in performance. The linear model tries to represent an inherently nonlinear system with a single linear model. The linear model is sufficiently accurate within the nominal operating region it was designed for. When a disturbance makes the system to operate away from this nominal operating region, the nonlinear behavior in input and output affects the model accuracy, leading the model predictive controller to deviate from the control objectives. In addition, when a linear model encounter discontinues/nonlinear control inputs it is hard to select the gains and tuning parameters to achieve the optimal performance. The Hammerstein model tries to capture the static input discontinuities/nonlinearities providing the linear controller more room and flexibility to control. Hence, the tuning parameter and gain selection becomes convenient. Moreover, integrating Wiener model in to the system enable the model to capture nonlinearities in the output, providing more linear model of the system. The main thing to remember here is Hammerstein-Wiener models are most suitable to captures the static nonlinearities in the system, which indicates that there are other nonlinear dynamics not captured by the Hammerstein-Wiener model. These dynamics makes the system to deviates from the objective value with some fluctuations. Overall, nonlinear block-oriented models captures sufficient amount of nonlinear dynamics improving the controller performance of the prototypical system we implemented.

*Limitations:* From the experiment results we can see that the nonlinear block oriented models described an inherently nonlinear system more accurately compared to a linear model of the system. However, limitations must be mentioned as well. The nonlinear block oriented Hammerstein-Wiener models are most useful in cases where nonlinearities are non-static (non time varying). If this is not the case, integrated nonlinear blocks may amplify the non-static nonlinearities making the system unstable [11]. On the other hand, if the information about static nonlinearities is know at the design time design process and runtime control becomes convenient. Further, one of the main concerns in the self-managed software systems is the computational demand of the decision making module [32](in our case the model based predictive controller). If we have fixed allocation there is no additional computational overhead, but we showed that it cannot achieve the desired performance even when the system is operating marginally away from the designed range. The linear controller can achieve required performance up to some extent by imposing some additional computational overhead. Similarly, the nonlinear model predictive controller performance much better in

wide range of situations, but imposes even more computational overhead compared to the linear model. Addition of two nonlinear blocks introduces many Multiplications and Addition (#MAD) operations. The computational efficiency also depends on the nonlinear function type selected (eg: polynomial Vs nonlinear-ARMA). For the controller we designed with two nonlinear blocks introduced 49 additional MAD operations, but for current software system platforms such number of primitive operation should not pose any difficulties. Yet another limitation is different original output ( $y$ ) values may map to relatively similar values of the transformed variables ( $w$ ). This was an issue in our implementation for instance when  $y = 3$  and  $15$  the transformed variable tend to have similar values ( $w = 22.06$  and  $22.53$  respectively). Such mappings make the transformed system to realize that both values are similar in transformed control system. So that, when the set point is at  $3$ , if a disturbance drag the system to have value of  $15$ , the transformed control system may settle the original system to  $15$  in instead of  $3$ , because of this similar in transformed control system. If such cases are possible, other types of functions could be utilized (such as splines).

*Applications:* The nonlinear block oriented approach we discussed in this work is applicable when some of the static nonlinearities in the inputs and outputs are known or can be analyzed at the design time. These cases would exist if relative generate feedback control design is applied in many systems for on-demand resource allocation, similar to the case study we presented in this paper. For instance, three tire e-commerce systems with limited storage /database connection threads, web servers with limited number of process threads and data centers with server clusters and cloud computing environments. The Hammerstein model is applicable in optimizing power consumption of processors (CPUs) by adjusting the operating voltage/ frequency through dynamic voltage scaling technology. For example, the possible voltage transitions [33] has nonlinear operating points, which could utilize the Hammerstein modeling approach described in this paper to achieve/improve power optimization objectives via feedback control. In addition, one of the issues in many resource allocation problems of software system is response time (or measured output) is *inversely* (nonlinearly) related to the amount of resources allocated. If we allocate more resources the response time going down, where as if we reduce the resources response time is going to increase. This gives us the indication that resource allocation is inversely proportional to the response time. Many publications have argued that due to this rea-

son to improve the model fit they have inverted the response time from the sensor and integrated in to the system [23, 23, 23]. Such nonlinearities could be effectively modeled as output nonlinearity using Weiner model. However, there could be cases where time varying behavior of nonlinearities. In such situations approach presented in [11], may be useful.

## 8. Conclusions

In this work we explore the performance improvements that can be achieved using nonlinear modeling of the software system. In particular, we develop a prototype of a real world software system and designed a relative guarantee control system to achieve the business objectives of the stake holders. We investigated the performance issues when the system is controlled using a linear model based predictive control. The investigation showed that the issues are mainly due to discontinuous control inputs and nonlinearities in the system. Consequently, we modeled the relative guarantee control scheme as a nonlinear block oriented model. We provide a step by step design procedure to model the software system as a Hammerstein-Wiener model, and how it can be used to implement a constraint based model predictive controller with transformed variables. Finally, we experimentally evaluate the performance of the linear, Hammerstein, Wiener, Hammerstein-Wiener models. Experiment results indicate that Hammerstein-Wiener model based predictive controller shows superior performance in many operating conditions compared to other models.

- [1] Y. Diao, J. L. Hellerstein, S. Parekh, J. P. Bigus, Managing web server performance with autotune agents, Vol. 42, 2003, pp. 136–149, 1014768.
- [2] H. Liu, S. Wee, Web server farm in the cloud: Performance evaluation and dynamic architecture, 2009, pp. 369–380.
- [3] M. Shaw, Beyond objects: a software design paradigm based on process control, Vol. 20, 1995, pp. 27–38, 225911.
- [4] M. Karlsson, X. Zhu, C. Karamanolis, An adaptive optimal controller for non-intrusive performance differentiation in computing services, Tech. rep., Hewlett Packard Laboratories (Februar1 8 2005).

- [5] M. Karlsson, C. Karamanolis, X. Zhu, Triage: Performance differentiation for storage systems using adaptive control, Vol. 1, 2005, pp. 457–480, 1111612.
- [6] C. Lu, Y. Lu, T. F. Abdelzaher, J. A. Stankovic, S. H. Son, Feedback control architecture and design methodology for service delay guarantees in web servers, *IEEE Trans. Parallel Distrib. Syst.* 17 (9) (2006) 1014–1027, 1159253.
- [7] D. Kusic, N. Kandasamy, Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems, in: *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on, 2006*, pp. 74–83.
- [8] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, K. Shin, What does control theory bring to systems research?, *SIGOPS Oper. Syst. Rev.* 43 (1) (2009) 62–69, 1496922.
- [9] J. L. Hellerstein, *Challenges in control engineering of computing systems*, 2004.
- [10] J. C. GOMEZ, E. BAEYENS, *Hammerstein and wiener model identification. using rational orthonormal bases*, 2003.
- [11] A. D. KALAFATIS, L. WANG, W. R. CLUETT, *Identification of wiener-type nonlinear systems in a noisy environment*, Vol. 66, 1997, pp. 923 – 941.
- [12] F. Jurado, *Hammerstein-model-based predictive control of micro-turbines*, 2006, pp. 511 – 521.
- [13] S. W. Sung, C. H. Je, J. Lee, D. H. Lee, Improved system identification method for hammerstein-wiener processes, *Korean Journal of Chemical Engineering* 25 (2008) 631–636.
- [14] L. Ljung, *System identification: theory for the user*, Prentice-Hall, Inc., 1997, 21413.
- [15] J. Voros, An iterative method for hammersteinwiener systems parameter identification, *Journal of ELECTRICAL ENGINEERING* (2004) 328–331.

- [16] E.-W. Bai, Decoupling the linear and nonlinear parts in hammerstein model identification, Vol. 40, 2004, pp. 671–676, doi: DOI: 10.1016/j.automatica.2003.11.007.
- [17] F. Giri, F. Chaoui, M. Haloua, Y. Rochdi, A. Naitali, Hammerstein model identification.
- [18] P. R.K, P. M, Gray-box identification of block-oriented nonlinear models, Vol. 10, 2000, pp. 301–315.
- [19] Y. Lu, T. Abdelzaher, C. Lu, G. Tao, An adaptive control framework for qos guarantees and its application to differentiated caching services, 2002.
- [20] A. Bemporad, M. Morari, Robust model predictive control: A survey, Vol. 245/1999, 1999, pp. 207–226.
- [21] L. Wang, Model Predictive Control System Design and Implementation Using MATLAB, Springer Publishing Company, Incorporated, 2009, 1592901.
- [22] S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, J. Bigus, Using control theory to achieve service level objectives in performance management, Vol. 23, 2002, pp. 127–141, 607902.
- [23] X. Zhu, Z. Wang, S. Singhal, Utility-driven workload management using nested control design, Tech. rep., Hewlett Packard Laboratories (April 06 2006).
- [24] R. Zhang, C. Lu, T. F. Abdelzaher, J. A. Stankovic, Controlware: A middleware architecture for feedback control of software performance851887 301.
- [25] K. J. strom, B. Wittenmark., Adaptive Control, 2nd ed. Electrical Engineering: Control Engineering, Addison-Wesley Publishing Company, 1995.
- [26] S. J. Norquay, A. Palazoglu, J. Romagnoli, Model predictive control based on wiener models, Vol. 53, 1998, pp. 75–84, doi: DOI: 10.1016/S0009-2509(97)00195-4.

- [27] Matlab, Matlab system identification toolbox.
- [28] W. C. L. Wang, From Plant Data to Process Control: Ideas for Process Identification and PID Design, Taylor and Francis, 2000.
- [29] J. C. Gomez, A. Jutan, E. Baeyens, Wiener model identification and predictive control of a ph neutralisation process, Vol. 151, 2004, pp. 329–338.
- [30] B.-G. Jeong, K.-Y. Yoo, H.-K. Rhee, Nonlinear model predictive control using a wiener model of a continuous methyl methacrylate polymerization reactor, Vol. 40, 2001, pp. 5968–5977, doi: 10.1021/ie990887b.
- [31] R. B. Abdennour, M. Ksouri, F. M’Sahli, Nonlinear model-based predictive control using a generalised hammerstein model and its application to a semi-batch reactor, The International Journal of Advanced Manufacturing Technology 20 (11) (2002) 844–852, 10.1007/s001700200225.
- [32] M. Salehie, L. Tahvildari, Self-adaptive software: Landscape and research challenges, ACM Trans. Auton. Adapt. Syst. 4 (2) (2009) 1–42, 1516538.
- [33] Intel, Enhanced intel speedstep technology for the intel pentium m processor.