

TECHNICAL RESEARCH REPORT

Run-to-Run Control Methods Based on the DHOBE Algorithm

by Hao Deng, Chang Zhang and John S. Baras

**CSHCN T.R. 99-33
(ISR T.R. 99-65)**



The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site <http://www.isr.umd.edu/CSHCN/>

Run-to-Run Control Methods based on the DHOBE Algorithm

Hao Deng, Chang Zhang and John S. Baras*

November 18, 1999

Abstract

Many run-to-run (RtR) control methods have been developed in recent years. Two particular set-valued RtR control schemes based on the Dasgupta-Huang Optimal Bounded Ellipsoid (DHOBE) algorithm are introduced. Compared to other RtR control schemes, the methods in this paper only need to know the bound of the noises, and are easy to implement. The DHOBE algorithm, for each recursion, returns an outer bounding ellipsoid of the estimated parameters. If the center of the ellipsoid each time is taken as the model coefficients, the explicit model update is implemented which leads to a model-reference method. If we choose the worst-case point which maximizes the cost function in the set, then we can apply the set-valued worst case approach. These two methods were compared with two other main RtR control schemes: the Exponentially Weighted Moving Average (EWMA) method and the Optimizing Adaptive Quality Controller (OAQC) method. Simulation results showed the superior performance of the RtR controllers based on the DHOBE algorithm. Furthermore this paper showed that it is necessary to apply nonlinear models to compensate for severe nonlinear processes.

1 Introduction

In industries like semiconductor manufacturing, specifications or changing conditions impose a need for adjusting process parameters on a run-to-run (RtR) basis. This need has originated a collection of techniques called RtR control [1]-[8]. RtR control is a form of discrete process and equipment control in which the product recipes with respect to a particular equipment

*This work was supported by the Center for Satellite and Hybrid Communication Networks, under NASA cooperative agreement NCC3-528

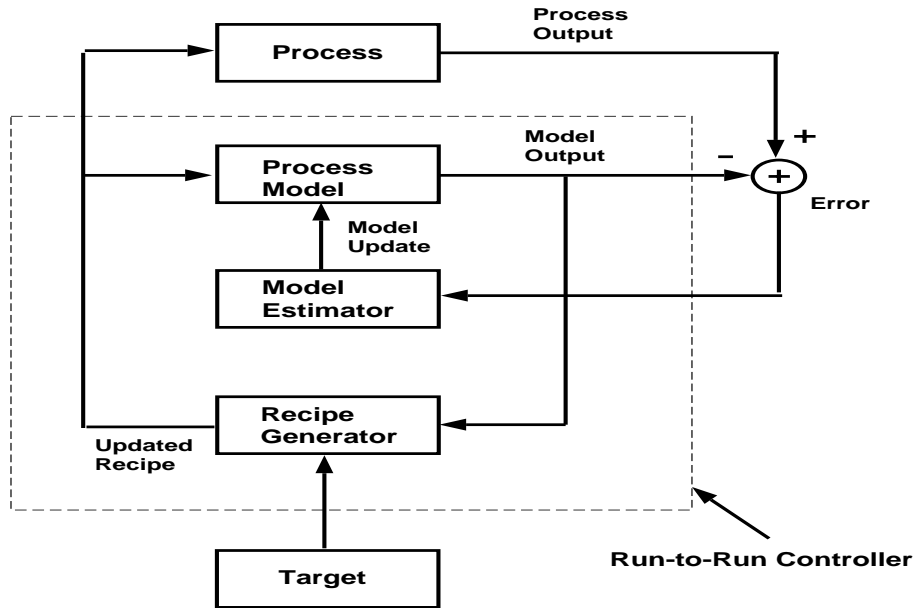


Figure 1: Structure of a RtR Controller

process are modified ex-situ, i.e. between equipment “runs”, so as to minimize the effects of process drifts, large shifts, and other variabilities to keep the outputs at the prescribed target values.

Generally the RtR controller is designed in the following way: First, it performs process control based only on post-process measurements. Then it responds to the measurements by updating models of the process between runs. Finally it provides a new recipe for use in the next run of the process. It does not modify the recipe during a run. The reason why it generates new recipes from the post-process measurements on a run-to-run basis is, on one hand, lack of online sensors for the process. On the other hand, frequent changes of inputs to the process may increase the variability of the process’s output. Sometimes dead-band is utilized in order to make the change less frequent.

The process models of a RtR controller need to take into account of the process variations throughout the whole process, providing enough accuracy for control without adjusting too many parameters. The initial models are derived from former off-line experiments such as using the response surface models. When the controller is online, the model within the controller is updated using the new measurements of the process from run to run. Different model updating methods usually lead to different kinds of controllers.

A typical block diagram of a RtR controller is illustrated in Figure 1. Normally the goal of the RtR controller is to reduce the variability of the process’s output as measured by the mean squared deviation (MSD) from the target.

Currently there are three main RtR control schemes available: The Exponentially Weighted

Moving Average (EWMA) RtR controller, the Optimizing Adaptive Quality Controller (OAQC), and the set-valued RtR controller.

1. The EWMA RtR controller was first developed by E. Sachs et al. from MIT [1]. It mainly compensates for slowly drifting processes that can be represented by linear(affine) models. The controller only updates the offset term in the process. The EWMA controller gives less and less weight to data as they get older and older . To control a process, it is convenient to forecast where the process will be in the next instance of time. Then if the forecast shows that a future deviation from target is too large, some remedial control actions from the controller or the process operator will compel the forecast to equal the target. The EWMA controller has been shown to improve the performance of semiconductor manufacturing processes. But a key problem with this method is that it is unable to adequately control processes which are poorly represented by linear models.

2. The OAQC controller is a form of model-referenced RtR controller [2], [5]. It can act both as an optimizer and a controller. In its optimizing mode, it updates the model at every run and in the controller mode, it uses a quadratic cost function to maintain the response of the process at the desired target with regards to the variation of the tunable parameters. It integrates the multivariate control chart as a dead-band to the controller in order to erase outliers, which are harmful to the control and optimizing actions. This method can be applied to light nonlinear models as well. Simulations have been done for second order Hammerstein models of the CMP process.

3. The set-valued RtR control method was proposed by J.S.Baras and N.S.Patel [4]. It is also a model-based method. However, the set-valued controller considers the uncertainty of the model identification. This uncertainty exists because normally the updated models can not be accurate due to model errors, measurement noises and other restrictions such as choice of the cost functions. What can be identified more reliably is the set of the model parameters in which the real process model resides. We could be quite certain that the model is somewhere in this set, but due to the randomness of the process, the exact position is unknown. In [4], they used the optimal volume ellipsoid (OVE) algorithm [9], [10] to obtain the parameter sets. Then they selected an estimate of model parameters from the set which makes the cost function the largest (worst case). Using these model parameters, the recipe that will optimize the cost function in the worst case will be found. However this recipe is somewhat conservative.

In this paper we are going to apply a more general ellipsoid algorithm, the Dasgupta-Huang Optimal Bounded Ellipsoid (DHOBE) algorithm in the realization of the RtR controller [8]. The DHOBE algorithm was first developed in 1987 by Dasgupta and Huang [12]. It modified the OBE algorithm of Fogel and Huang [11] by introducing a forgetting factor. It was further developed by Rao and Huang in 1993 [13] by introducing the rescue procedure. This algorithm belongs to a class of bounded-error estimation algorithms termed set-membership parameter estimation algorithms, which is different from the OVE algorithm.

The largest benefit of this algorithm is that unlike other recursive algorithms, the DHOBE algorithm discerns if the new measurement contains any fresh information. This reduces significantly the computation load for estimation. The only knowledge required from the real process is the strict bound of the noise instead of the detailed property (e.g. distribution) of the noise. Another improvement is the introduction of the rescue procedure. The semiconductor manufacturing process usually undergoes abrupt shifts (due to equipment maintenance) and modeling errors (considered as big shift at the startup). This normally causes other model identification methods to return an empty set for the parameters. The rescue procedure greatly improves the performance of the algorithm under this circumstance and accordingly that of the controller. Therefore the DHOBE algorithm based controller works well for large step disturbances and model errors, which are hard for other methods to compensate for.

The DHOBE algorithm also gives us the freedom of applying the model-reference method or the worst-case method. Because for each recursion, the returned result is not a single model (point) but an outer bounding ellipsoid of the estimated parameters. If, according to the algorithm, the center of the ellipsoid each time is taken as the model coefficients, the explicit model update is implemented which leads to a model-reference method. If we choose the worst-case point which maximizes the cost function in the set, then we can apply the set-valued worst case approach. We will call them DHOBE-MR and DHOBE-SV respectively in the later part of this paper.

2 The RtR Controller Based on the DHOBE algorithm

2.1 The DHOBE Algorithm

The main idea of the DHOBE algorithm is to recursively obtain the ellipsoidal outer bounds to the membership set. It can be applied to linear-in-the-parameters models like the Hammerstein model which ensures that the algorithm can be used to estimate the parameters of quadratic models.

Assume that the scalar process model is of the following form:

$$y_t = \theta^{*T} u_t + v_t \quad (1)$$

where y_t is the scalar output of the process, $\theta^* \in \mathfrak{R}^N$ is the true parameter vector and u_t is the input vector. v_t is the noise term which is bounded by γ , i.e.,

$$|v_t| \leq \gamma. \quad (2)$$

Suppose that at time instance t-1, the membership set of the parameters of the model is

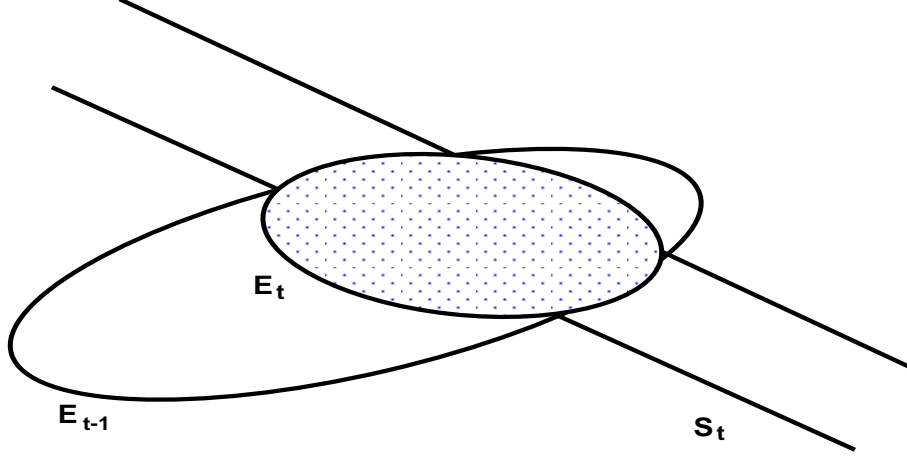


Figure 2: Recursive Formation of the Bounding Ellipsoid [12]

bounded by the ellipsoid E_{t-1} . It can be defined by its center θ_{t-1} , its orientation from the positive definite matrix P_{t-1}^{-1} and its size from the uncertainty parameter σ_{t-1}^2 :

$$E_{t-1} = \{\theta \in \mathfrak{R}^N : [\theta - \theta_{t-1}]^T P_{t-1}^{-1} [\theta - \theta_{t-1}] \leq \sigma_{t-1}^2\} \quad (3)$$

On the next time instance t , we have an observation of the process y_t . Then we can utilize it to obtain the set S_t that the process will reside under the constraints of the noises as follows:

$$S_t = \{\theta \in \mathfrak{R}^N : [y_t - \theta^T u_t]^2 \leq \gamma^2\}. \quad (4)$$

S_t will intersect with E_{t-1} which will enable us to recursively enclose the intersection of the two sets by a new ellipsoid E_t as follows:

$$\begin{aligned} E_t &= \{\theta \in \mathfrak{R}^N : (1 - \lambda_t)[\theta - \theta_{t-1}]^T P_{t-1}^{-1} [\theta - \theta_{t-1}] + \lambda_t [y_t - \theta^T u_t]^2 \\ &\leq (1 - \lambda_t)\sigma_{t-1}^2 + \lambda_t \gamma^2\}, \end{aligned} \quad (5)$$

where the updating gain λ_t is introduced. The gain λ_t is positive and time varying. We can also consider $(1 - \lambda_t)$ as the forgetting factor. λ_t is chosen to minimize σ_t^2 at each time instance in order to decrease the size of the ellipsoid from run to run. The recursive formation of the ellipsoid is illustrated in a 2-dimensional example in Figure 2.

The DHOBE algorithm can be generalized as the following steps:

step 1. Compute an error residual

$$\delta_t = y_t - u_t^T \theta_{t-1} \quad (6)$$

step 2. If

$$\sigma_{t-1}^2 + \delta_t^2 \leq \gamma^2 \quad (7)$$

then it is thought as a small noise disturbance and there is no update of the ellipsoid, otherwise go to step 3.

step 3. Compute two intermediate scalar variables:

step 3a.

$$G_t = u_t^T P_{t-1} u_t \quad (8)$$

step 3b.

$$\beta_t = (\gamma^2 - \sigma_{t-1}^2) / \delta_t^2 \quad (9)$$

step 4. Compute the update factor λ_t . Here $1 - \lambda_t$ is the forgetting factor. It is confined to the range $0 \leq \lambda_t < 1$.

$$\lambda_t = \min(\lambda_{max}, v_t) \quad (10)$$

where

$$v_t = \begin{cases} \lambda_{max} & \text{if } \delta_t^2 = 0 \\ (1 - \beta_t)/2 & \text{if } G_t = 1 \\ \frac{1 - \sqrt{\frac{G_t}{1 + \beta_k(G_t - 1)}}}{1 - G_t} & \text{if } \beta_k(G_t - 1) > -1 \\ \lambda_{max} & \text{if } \beta_k(G_t - 1) \leq -1 \end{cases} \quad (11)$$

where λ_{max} is a user-determined ‘‘design factor’’ which controls the rate of convergence of the algorithm. It is confined into the range of (0,1). In practice it was found that in the range from 0.05 to 0.10, λ_{max} usually results in rapid and well-behaved convergence.

step 5. Update the parameter uncertainty factor

$$\sigma_t^2 = (1 - \lambda_t) \sigma_{t-1}^2 + \lambda_t \gamma^2 - \frac{\lambda_t (1 - \lambda_t) \delta_t^2}{1 - \lambda_t + \lambda_t G_t} \quad (12)$$

step 6. This is the rescue procedure. It was developed by Rao and Huang in [13]. It is particularly necessary when there is a large jump in the parameters of the model. In such cases, the algorithm might return an empty set and thus there is no bounding ellipsoid generated. This is because when the parameters change abruptly in a step, the intersection of

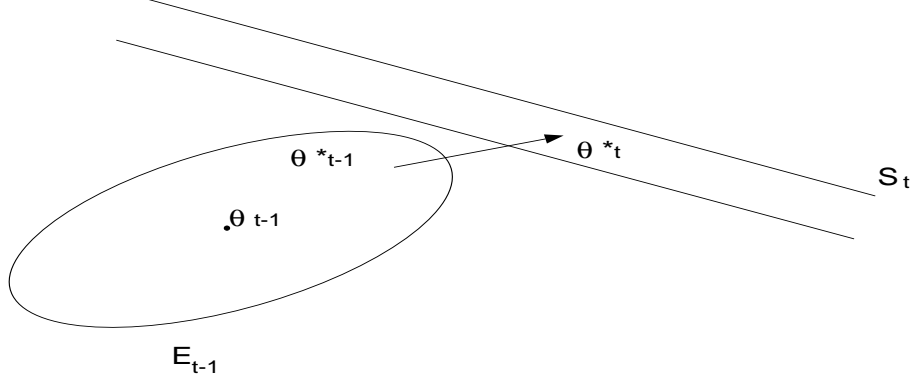


Figure 3: The DHOBE Algorithm Rescue Procedure Illustration

E_{t-1} and S_t will be void as illustrated in Figure 3. At this time the calculation of the σ_t^2 will become negative which is an indication that there will be no bounding ellipsoid. Then the rescue procedure is called at this time to enlarge the size of E_{t-1} so that the intersection of S_t and the enlarged ellipsoid E_{t-1} will no longer be void. This rescue procedure will migrate the center of the ellipsoid to the real parameter vector θ^* which will reduce the parameter estimation error:

When $\sigma_t^2 > 0$, proceed to step 7; otherwise, compute

$$\kappa = \begin{cases} \delta_t^2 + \gamma^2 - 2\gamma|\delta| & \text{if } \lambda_t \neq \lambda_{max} \\ \lambda_{max} \left[\frac{\delta_t^2}{1 - \lambda_{max} + \lambda_{max} G_t} - \frac{\gamma^2}{1 - \lambda_{max}} \right] & \text{if } \lambda_t = \lambda_{max} \end{cases} \quad (13)$$

Reset the uncertainty parameter for time t-1

$$\sigma_{t-1}^2 = \kappa + \zeta \quad (14)$$

then return to step 3b. Here ζ is called ‘‘inflation parameter’’. It is user-specified and usually set as 1 [14]. In simulation it was found that the rescue procedure is not often taken.

step 7. Update the ellipsoid parameters.

$$P_t^{-1} = (1 - \lambda_t)P_{t-1}^{-1} + \lambda_t u_t u_t^T \quad (15)$$

$$P_k = \frac{1}{1 - \lambda_t} \left[P_{t-1} - \frac{\lambda_t P_{t-1} u_t u_t^T P_{t-1}}{1 - \lambda_t + \lambda_t G_t} \right] \quad (16)$$

$$\theta_t = \theta_{t-1} + \lambda_t P_t u_t \delta_t \quad (17)$$

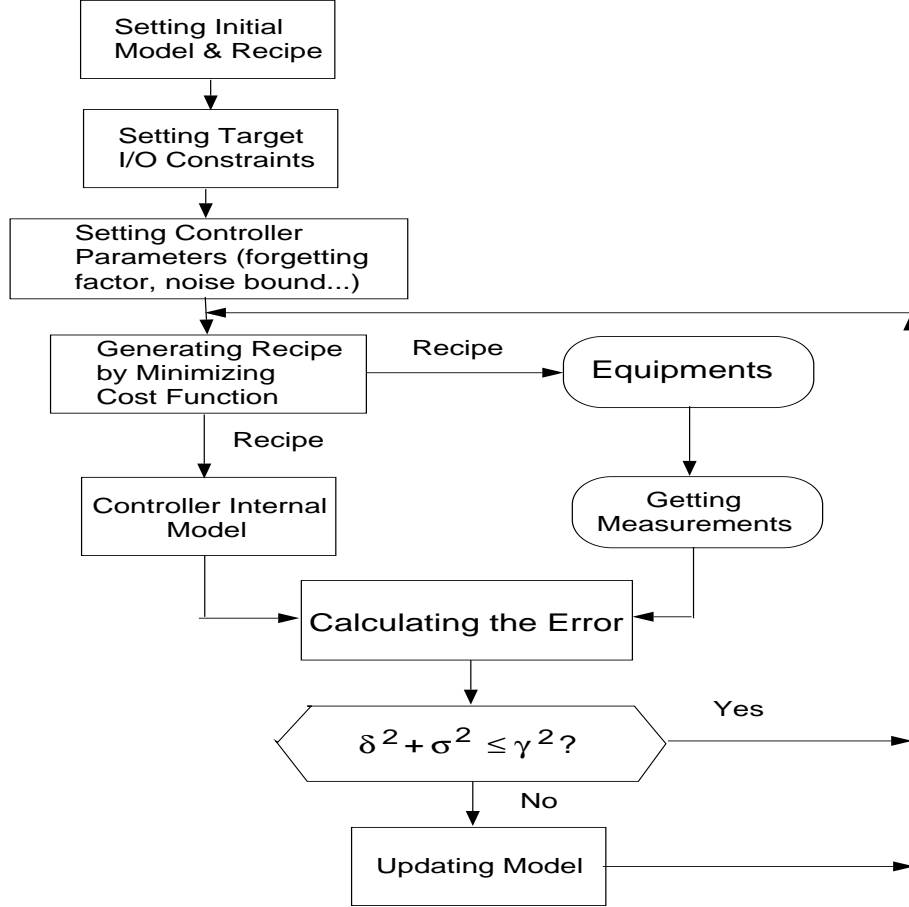


Figure 4: Block Diagram for the DHOBE algorithm Based Model-Reference RtR Controller

The initial conditions of the DHOBE algorithm can usually be set as $\sigma_0^2 = 100, P_0^{-1} = I$ to include the optimal estimation point.

In this paper the DHOBE algorithm is applied to both the model-reference RtR controller and the set-valued controller with the worst case approach.

2.2 The Model-Reference RtR Controller Based on the DHOBE Algorithm

In model-reference RtR controller, the DHOBE algorithm is used to implement RtR model parameter estimation. The recipes are generated using the minimization of the squared error between the model's predictive output and the target value. Figure 4 illustrates the basic flow chart of the controller.

The first step is to set the parameters related to the internal process model. This includes

the following:

- Building the initial model which might be obtained from off-line experiments.
- Setting the target value of the process outputs.
- Setting the constraints of the inputs.
- Setting the initial recipes of the equipment which might be the nominal value of the inputs.

The second step is to set parameters related to the controllers. This parameter setting is not easy as it is equivalent to the tuning of the controllers. But after some off-line simulations are completed, the recommended values can be available. These parameters include:

- Setting the strict noise bound γ .
- Setting the updating factor's (λ) upperbound λ_{max} .
- Setting the rescue step length ζ which is normally set to 1.
- Setting the ellipsoid's orientation matrix P which is normally set to I at the start.

The choice of the noise bound is the most important. When γ is chosen to be larger than the actual bound, the tracking ability of the algorithm will be increased. But from step 2 of the DHOBE algorithm, it is easy to see that when the noise bound is too large, the update of the model will not happen until the model's output significantly deviates from the target. This is not expected especially when the process is undergoing small drifts. The sensitivity of the controller will be decreased because of this larger noise bound. In our simulations, the process noise and measurement noise are combined to a white noise and the normal distribution has zero mean and variance σ_w . We took $3\sigma_w$ as the noise bound and it is shown that this bound is appropriate enough for the simulated circumstances.

The setting of the maximum value of the updating factor (λ_{max}) is also important. It controls the rate of convergence of the algorithm. If it is very small, convergence of the algorithm is slow. If it is too large, the size of the ellipsoid (σ^2) may change so rapidly that the ellipsoid's boundary excludes the real process parameter (θ^*). If θ^* ends up too far outside the ellipsoid, the result is usually an ellipsoid which collapses to the empty set and invokes the rescue procedure. The re-inflated ellipsoid can then miss θ^* again at the next update cycle and so on resulting in a non-converging oscillation of the ellipsoid. In our simulations, we use 0.4 as the maximum value and by selecting the proper re-inflation step ζ , satisfactory control results can be achieved.

After having a model estimation for each run, the recipe can be calculated for the next run by minimizing a predefined cost function. The cost function normally takes the form of

squared error between target value and the model's output using the current model estimation. When there is only one response the form might be

$$\min_u (T - \theta^T u) \quad (18)$$

where T is the target value.

If there are multiple responses, the cost function might take the form

$$\min_u (T - \theta^T u)^T W (T - \theta^T u), \quad (19)$$

where W is the weight diagonal matrix assigned for each response. The value of each element represents the priority of each response. The most important output is assigned the largest weight, which is used in order to keep the output value as close to its target value as possible. T is the target vector.

If the change of recipes incurs large costs, then this change should also be taken into consideration by adding an additional term into the cost function as in [2]:

$$\min_{u_t} \{w_1 (T - \theta^T u_t) + w_2 (u_t - u_{t-1})^T \Gamma (u_t - u_{t-1})\} \quad (20)$$

where Γ is the weight diagonal matrix assigned to each input of the recipe. If the cost for changing a certain tunable input is higher, then we can assign larger weight to that particular input so that by optimizing the cost function, it is kept as close to the previous run's value as possible. w_i are the weight terms also.

The minimization is achieved by finding the optimal recipe during each run through line searching. The calculated inputs should also satisfy the constraints that are essential for equipment and process requirements. These constraints are normally in the form of ranges which are quite natural and easy to get. They guarantee that the optimal recipe is feasible physically and reasonable to the equipment.

After the recipe is calculated, the equipment tunable inputs are adjusted accordingly, which might cause the output of the real process to change. Measurements are then fed back to the controller as new information. Then according to the DHOBE algorithm, the residual error $\delta(t)$ between the current measurement and the model's output will be calculated and used to estimate the next run's model.

At this time, the controller will decide if the updating is necessary by the following criterion:

$$\delta^2 + \sigma^2 \leq \gamma^2$$

If the inequality is satisfied, the controller uses the previous model for the next run and the cycle restarts; otherwise, the update of the process model is implemented.

2.3 The Set-Valued Worst-Case RtR Controller Based on the DHOBE algorithm

The set-valued worst-case RtR controller based on the DHOBE algorithm is similar to the model-referenced one except for the part of optimization of the cost function and generation of recipes. The model updating still uses the centers of the ellipsoids as the model parameter estimates, recursively. But after the outer bounds of the ellipsoids are generated, the optimization selects the model from these ellipsoids (sets) and tries to find the min-max of the cost function [4].

Suppose at run t , the ellipsoid is generated by the DHOBE algorithm with the parameters: center of ellipsoid θ_t^* , orientation P_t and size σ_t^2 . The optimization of the cost function is implemented as

$$\min_{u \in U} \max_{\underline{y}_t \leq y \leq \bar{y}_t} l(y) \quad (21)$$

where l is the cost function and U is the feasible set of inputs; U can be achieved by imposing constraints according to the requirements from the process, and

$$\underline{y}_t = \theta_t^{*T} u - \sqrt{u^T \frac{P_t}{\sigma_t^2} u} - \sqrt{\gamma} \quad (22)$$

$$\bar{y}_t = \theta_t^{*T} u + \sqrt{u^T \frac{P_t}{\sigma_t^2} u} + \sqrt{\gamma} \quad (23)$$

\underline{y}_t and \bar{y}_t define the inner maximization scope. In order to simplify the calculation, we impose the restriction that the cost function is convex. Then the maximization will only appear on either of the two points.

For multiple responses process, just like the model-reference controller, we used the Pareto-optimal method so that different weights are selected for different responses and then sum them together. The worst case treatment is to select the largest one of the cost functions between the upper and lower bounds of the response among all the responses and sum the weighted sub-functions together.

3 Simulations

The process models were derived from experiments or practical semiconductor processes. The models are in polynomial forms and can be linear, quadratic or of any higher order, which are subject to various disturbances that can happen. The measurements are also simulated as corrupted by noise. For simpleness it was assumed that the noises are normally distributed with definite variance and zero mean and uncorrelated (Other forms of noises are also simulated and the results are similar to the normally distributed one).

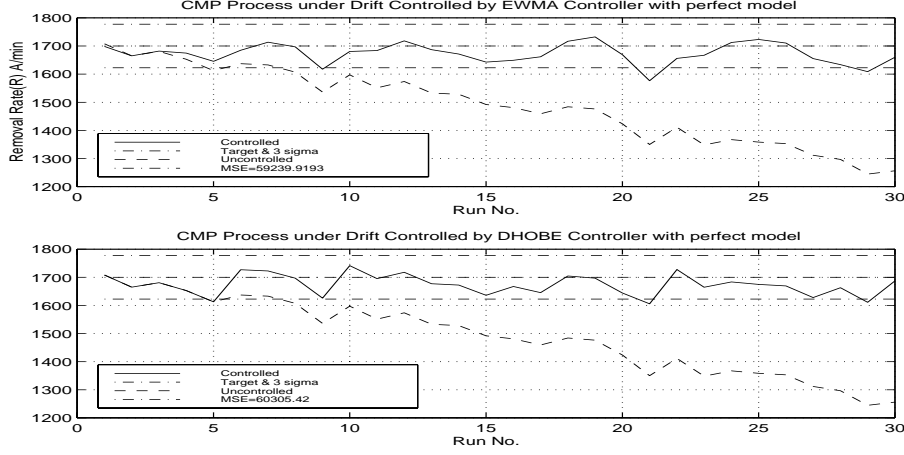


Figure 5: Comparison of the DHOBE-MR Controller with the EWMA Controller: Linear Perfect Model under Drifts

In the comparisons of the simulation results between the DHOBE algorithm based RtR controllers and other RtR controllers, the following statistical measures are used to evaluate the performance of the control action:

- \bar{y}_i is the mean of the sampling values from the real process i th output.
- S_{y_i} is the standard deviation of the process i th output.
- $MSD(y_i - T_i)$ is the square root mean square deviation of the process i th output from its target value.

3.1 Comparison of the DHOBE-MR and DHOBE-SV Controller with the EWMA controller

The process model we used is the chemical mechanical planarization (CMP) process [5], where the units are dropped for simpleness:

$$y[n] = -1382.60 + [50.18, -6.65, 163.4, 8.45] \times u^T(n) + w[n] + \delta n \quad (24)$$

where n is the run number, $w[n]$ is normally distributed white noise with variance 665.64, δ is the drift noise added to the process. Here $\delta = -17$. First let us assume that there is no model error exist. The weight of the EWMA controller is selected as 0.6 in this case. The simulation results are illustrated in Figure 5 and Figure 6 respectively for the comparison of the DHOBE-MR and DHOBE-SV method with the EWMA method.

From Table 1, it can be seen that the compensation effect of the DHOBE algorithm based controllers has no big difference with that of the EWMA controller as measured by the MSE and standard deviation.

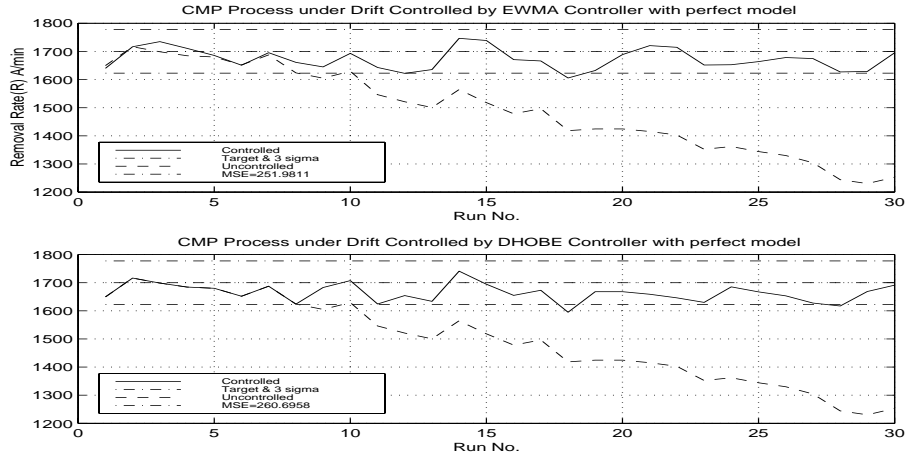


Figure 6: Comparison of the DHOBE-SV Controller with the EWMA Controller: Linear Perfect Model under Drifts

Method	\bar{y}	S_y	MSD
EWMA	1674.4	36.7	44.8
DHOBE-MR	1664.6	36.9	51.5
DHOBE-SV	1669.9	35.1	46.4

Table 1: Comparison of the EWMA Controller, the DHOBE-MR Controller and the DHOBE-SV Controller for Linear Perfect CMP Model

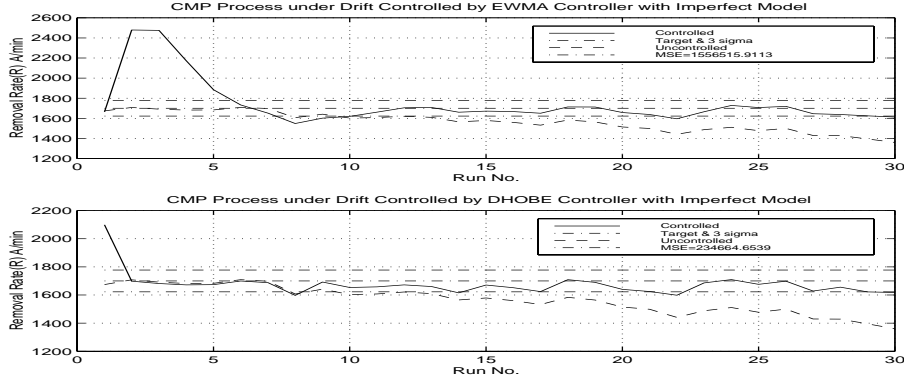


Figure 7: Comparison of the DHOBE-MR Controller and the EWMA Controller for Linear Imperfect Model under Drifts

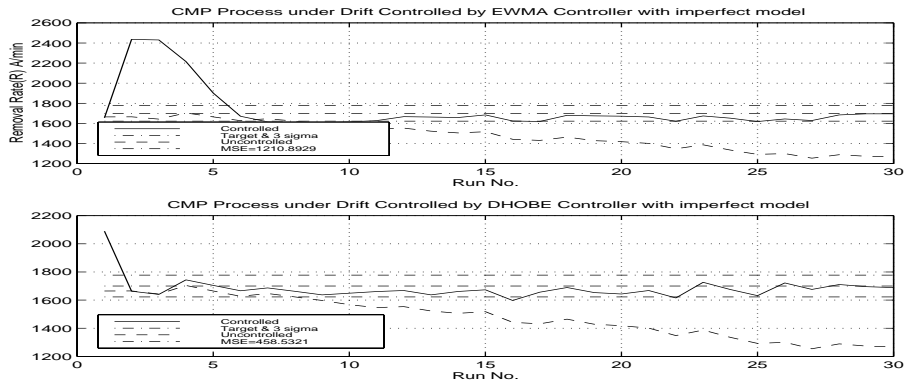


Figure 8: Comparison of the DHOBE-SV Controller and the EWMA Controller for Linear Imperfect Model under Drifts

But when model error exists, which is common in real applications, the DHOBE-MR and DHOBE-SV controlled processes have faster convergence characteristics (as seen from Figure 7 and 8). The output of the process changed significantly at the beginning for this case as the process model was taken as 80% of each parameters of the real process. The weight of the EWMA controller is 0.3 in this case. The DHOBE-MR and DHOBE-SV controllers are shown to track the target much faster than the EWMA controller under this circumstance.

This is also illustrated in Table 2 by considering the STD and MSD.

Simulations were also performed for the case when there is a large step disturbance (shift) at run 10. The step disturbance in the simulation was obtained by changing the model parameters of the real process significantly. The resulted shifting value equals to the target value + 350. The weight for the EWMA controller is selected as 0.43. The simulation results (Figure 9 and 10) also show that the DHOBE-MR and DHOBE-SV controller are better than the EWMA controller because the EWMA controller is suitable for slowly changing processes.

Method	\bar{y}	S_y	MSD
EWMA	1742.9	219.7	223.9
DHOBE-MR	1672.5	90.9	95.1
DHOBE-SV	1685.7	90.0	91.3

Table 2: Comparison of the EWMA controller, the DHOBE-MR Controller and the DHOBE-SV Controller for Linear Imperfect CMP Model

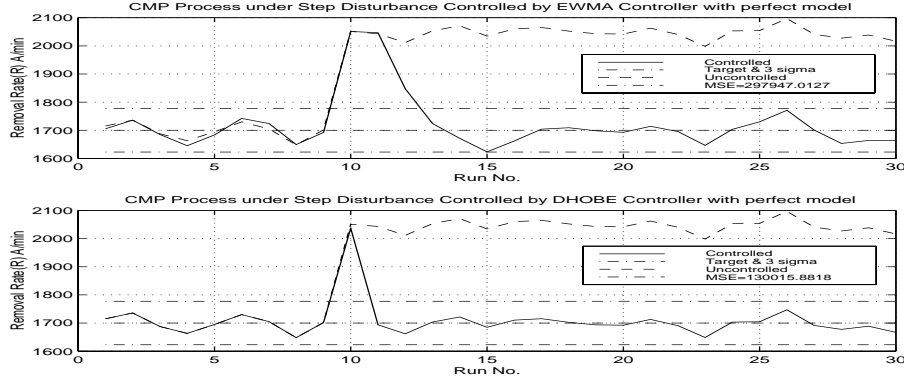


Figure 9: Comparison of the DHOBE-MR Controller and the EWMA Controller for Linear Perfect Model under Step Disturbance

It is hard for it to compensate for large variance in several runs.

The simulation data for this case are shown in Table 3. Therefore the performance of the DHOBE-MR and the DHOBE-SV controller is better or comparable to that of the EWMA controller. There is no big difference between the two controllers based on the DHOBE algorithm.

Method	\bar{y}	S_y	MSD
EWMA	1717.1	97.9	100.8
DHOBE-MR	1708.9	67.0	68.1
DHOBE-SV	1710.9	68.6	69.8

Table 3: Comparison of the EWMA Controller, the DHOBE-MR Controller and the DHOBE-SV Controller for CMP Model under Step Disturbance

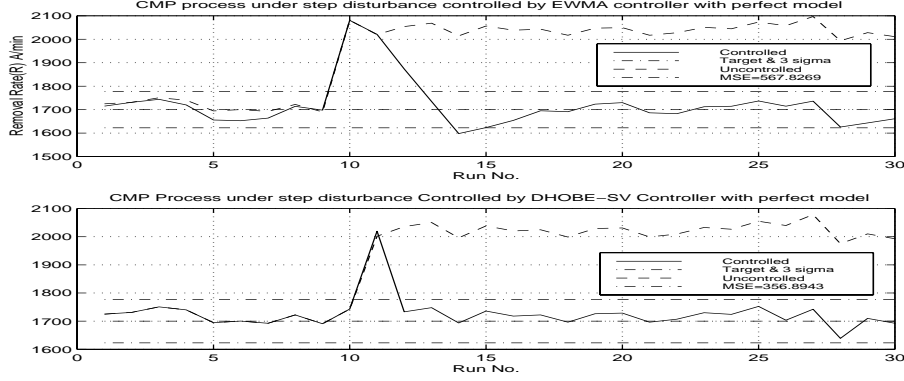


Figure 10: Comparison of the DHOBE-SV Controller and the EWMA Controller for Linear Perfect Model under Step Disturbance

3.2 Comparison of the DHOBE-MR and DHOBE-SV Controller with the OAQC Method

Detailed introductions about the OAQC method can be found in [2]. The process models obtained from experiments are considered as real process models with drifts and the models that we are going to use to approximate the real models are in almost-linear, fully quadratic or linear forms respectively. Performance was evaluated according to different kinds of approximate models. This provides us an opportunity to test the controller's robustness to model errors. There are two forms of the model with regards to the number of parameters to be tuned: CMP4x2 model and CMP3x2 model.

3.2.1 CMP 4×2 almost-linear Model

1) Process Model Considered as "Real"

The underlying "real" process is [2]:

$$\begin{aligned}
 y_1 = & 1563.5 + 159.3u_1 - 38.2u_2 + 178.9u_3 + 24.9u_4 - 67.2u_1u_2 - 46.2u_1^2 \\
 & - 19.2u_2^2 - 28.9u_3^2 - 12u_1t' + 116u_4t' - 50.4t' + 20.4t'^2 + \epsilon_{1,t}
 \end{aligned} \tag{25}$$

$$\begin{aligned}
 y_2 = & 254 + 32.6u_1 + 113.2u_2 + 32.6u_3 + 37.1u_4 - 36.8u_1u_2 + 57.3u_4t' \\
 & - 2.42t' + \epsilon_{2,t}
 \end{aligned} \tag{26}$$

where

$$t' = (t - 53)/53, \epsilon_{1,t} \sim N(0, 60^2), \epsilon_{2,t} \sim N(0, 30^2) \text{ ,and}$$

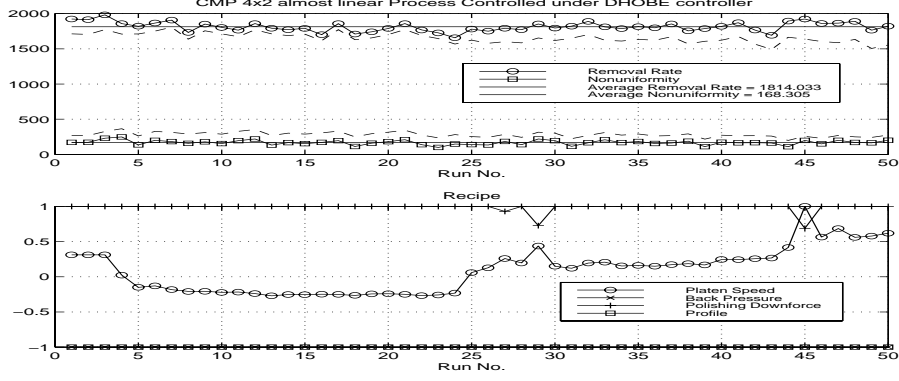


Figure 11: CMP4x2 Scenario 1 Controlled by the DHOBE-MR Controller

y_1 is the removal rate; its target value is 2000.

y_2 is the with-in wafer non-uniformity; its target value is 100.

u_1 is the platen speed.

u_2 is the back pressure.

u_3 is the polishing downforce.

u_4 is the profile.

It is a rather complex form as it includes both the quadratic and two factor interaction terms. All controllable factors $u_1 - u_4$ are scaled to $[-1,1]$ range. The target values for y_1 and y_2 , 2000 and 100 may be unrealistic. These values are set in order to evaluate the performance of the algorithm. For y_1 , the larger value the better; and for y_2 , the smaller the value the better.

2) Approximate Initial Models – Quadratic Models (Scenario 1)

The above “real” process model is unknown to us. So we have to use some models to approximate the ‘real’ process model. First, the fully quadratic model used in the controller is of the following form:

$$y_1 = 1600 + 150u_1 - 40u_2 + 180u_3 + 25u_4 - 30u_1^2 - 20u_2^2 - 25u_3^2 - 60u_1u_2 - 0.9t \quad (27)$$

and

$$y_2 = 250 + 30u_1 + 100u_2 + 20u_3 + 35u_4 - 30u_1u_2 + 0.05t \quad (28)$$

We can see that there is a big model error when compared with the real process model for both parameters and drifts.

3) Approximate Initial Models – Linear Models (Scenario 2)

If the quadratic and interaction terms are dropped then the above model is changed to a linear model. The constraints for the input and output are the same as in the quadratic model.

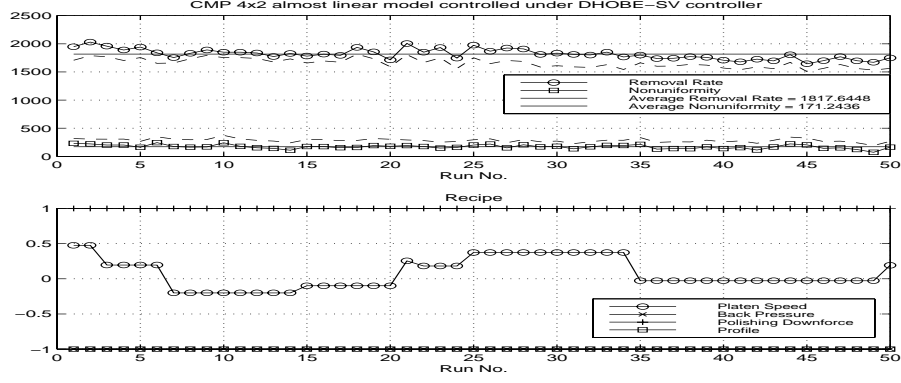


Figure 12: CMP4x2 Scenario 1 Controlled by the DHOBE-SV Controller

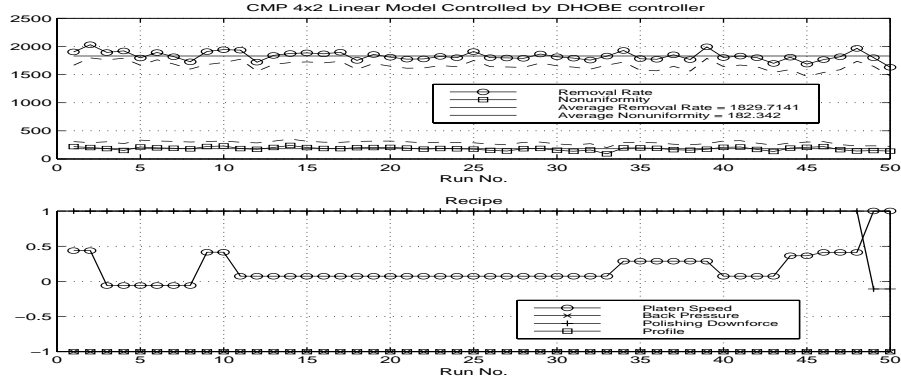


Figure 13: CMP4x2 Scenario 2 Controlled by the DHOBE-MR Controller

$$y_1 = 1600 + 150u_1 - 40u_2 + 180u_3 + 25u_4 - 0.9t \quad (29)$$

$$y_2 = 250 + 30u_1 + 100u_2 + 30u_3 + 35u_4 + 0.05t \quad (30)$$

As the real process model is not severely nonlinear, the control effect with the linear model is good (Figure 13 and 14).

4) Quadratic Models with Step Disturbance (Scenario 3)

The DHOBE algorithm was also tested when abrupt disturbances (shifts) happened. In this case the quadratic initial model was used and the constraints were the same as before. The abrupt shift for the first response happened at $t = 20$ with magnitude -100 and for the second response the shift happened at $t = 30$ with magnitude 50. The simulation results are shown in Figure 15 and Figure 16.

5) Performance Analysis

The OAQC method was simulated in [2] under exactly the same circumstances for the above 3 scenarios. The final results with regards to the statistical variance analysis were listed in

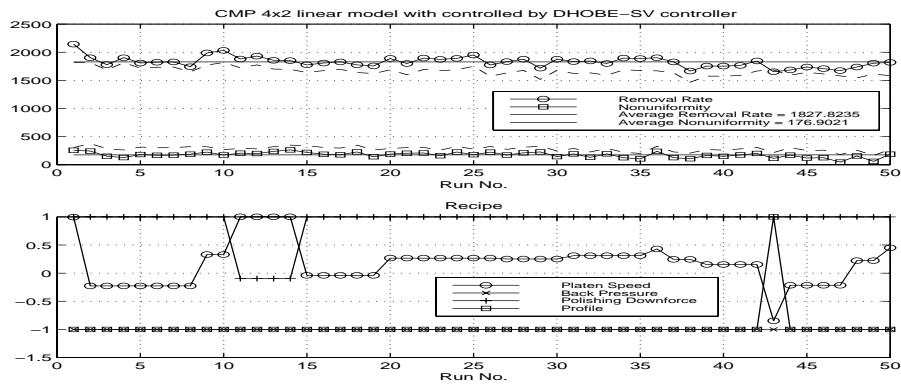


Figure 14: CMP4x2 Scenario 2 Controlled by the DHOBE-SV Controller

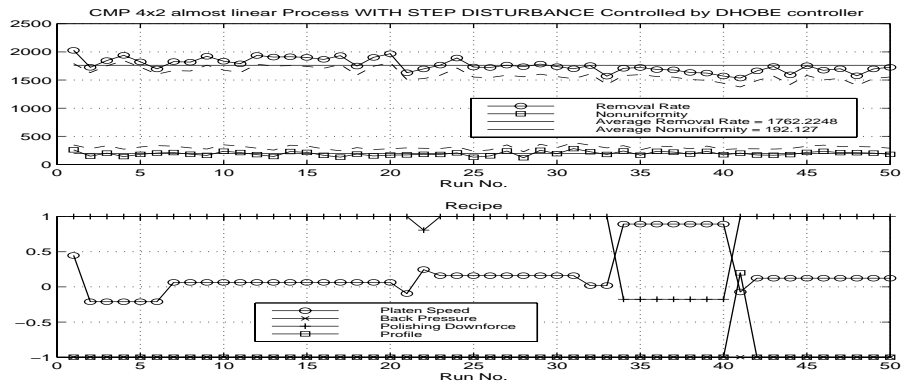


Figure 15: CMP4x2 Scenario 3 Controlled by DHOBE-MR

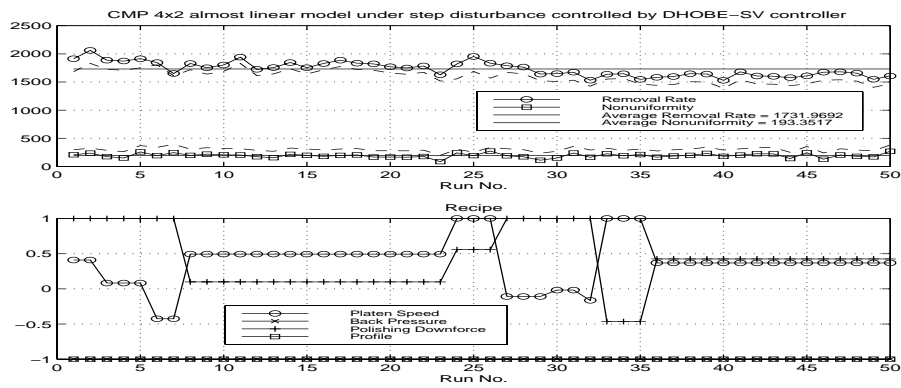


Figure 16: CMP4x2 Scenario 3 Controlled by DHOBE-SV

Scenario	Method	\bar{y}_1	\bar{y}_2	S_{y_1}	S_{y_2}	MSD_1	MSD_2
1	OAQC	1719.7	168.4	70.4	40.1	288.9	79.2
	DHOBE-MR	1754.7	157.3	84.5	35.0	259.7	67.5
	DHOBE-SV	1787.7	168.1	82.8	34.7	228.2	76.9
2	OAQC	1718.2	165.7	72.1	42.0	291.0	78.2
	DHOBE-MR	1781.9	165.0	84.5	36.1	234.2	74.8
	DHOBE-SV	1807.4	177.5	85.9	36.1	211.9	86.1
3	OAQC	1661.2	189.2	89.2	43.5	350.2	99.2
	DHOBE-MR	1741.4	189.1	108.7	35.6	280.8	96.0
	DHOBE-SV	1747.0	190.8	109.2	37.5	275.9	98.3

Table 4: Performance Measure of the OAQC Method, the DHOBE-MR and the DHOBE-SV Controllers for CMP 4x2 Models

Table 4. There might be some factors from the tuning of the controller but we can see the tendency, and thus prove the feasibility of the new controller.

Table 4 shows that for response 1, the mean values of the DHOBE algorithm based RtR controllers are normally better than the OAQC controller, but their standard deviations are larger than OAQC controller. For the second response, the mean values of the DHOBE algorithm based RtR controllers are larger but the standard deviations are better than the OAQC controller.

3.2.2 CMP Nonlinear 3×2 Model

In this section, the second CMP process model is used. It has only 3 controllable factors and the responses are removal rate (y_1) and within-wafer standard deviation (y_2).

1) Underlying “Real” Process Model

$$y_1 = 276.5 + 574.6u_1 + 616.3u_2 - 126.7u_3 - 1109.5u_1^2 - 286.1u_2^2 + 989.1u_3^2 - 52.9u_1u_2 - 156.9u_1u_3 - 550.3u_2u_3 - 10t + \epsilon_{1,t} \quad (31)$$

and

$$y_2 = 746.3 + 62.3u_1 + 128.6u_2 - 152.1u_3 - 289.7u_1^2 - 32.1u_2^2 + 237.7u_3^2 - 28.9u_1u_2 - 122.1u_1u_3 - 140.6u_2u_3 + 1.5t + \epsilon_{2,t} \quad (32)$$

where $\epsilon_{1,t} \sim N(0, 60^2)$, $\epsilon_{2,t} \sim N(0, 30^2)$.

Controllable factors are back pressure downforce (u_1), platen speed (u_2) and slurry concentration (u_3). They are all scaled to $[-1,1]$ range and the target values for y_1 and y_2 are 2200 and 400 respectively. These models are fitted to the results of a 32-wafer experimental

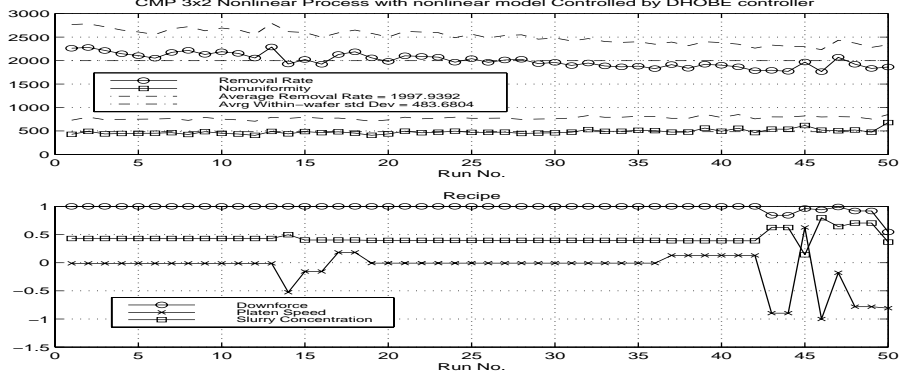


Figure 17: CMP3x2 Scenario 1 Controlled by the DHOBE-MR Controller

design and they served as “real” process model in the simulation. They are hard to control using a linear model controller as they contain large second-order coefficients. This model also shows the necessity of adopting the nonlinear controller.

2) Approximate Initial Models–Quadratic Models (Scenario 1)

The models we used to approximate the “real” process models at the beginning of runs are:

$$\begin{aligned}
 y_1 = & 2500 + 400u_1 + 500u_2 - 100u_3 - 800u_1^2 - 200u_2^2 + 1000u_3^2 - 40u_1u_2 \\
 & - 100u_1u_3 - 350u_2u_3 - 7t
 \end{aligned} \tag{33}$$

$$\begin{aligned}
 y_2 = & 600 + 50u_1 + 100u_2 - 100u_3 - 200u_1^2 - 50u_2^2 + 300u_3^2 - 30u_1u_2 - \\
 & 100u_1u_3 - 100u_2u_3 + 3t
 \end{aligned} \tag{34}$$

The simulation results are shown in Figure 17 and Figure 18.

3) Approximate Initial Models–Linear Models (Scenario 2)

The following simulations used a linear model to compensate for the nonlinear process model. This also represents the case that using linear models may not be adequate for nonlinear processes.

The linear models we used are:

$$y_1 = 2500 + 400u_1 + 500u_2 - 100u_3 - 7t \tag{35}$$

$$y_2 = 600 + 50u_1 + 100u_2 - 100u_3 + 3t \tag{36}$$

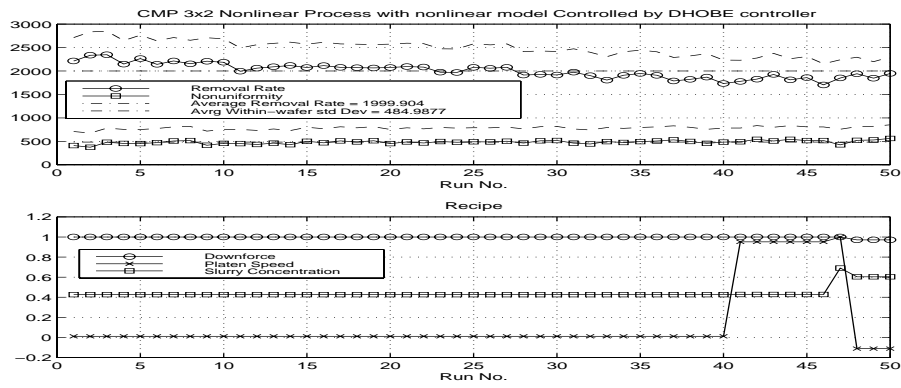


Figure 18: CMP3x2 Scenario 1 Controlled by the DHOBE-SV Controller

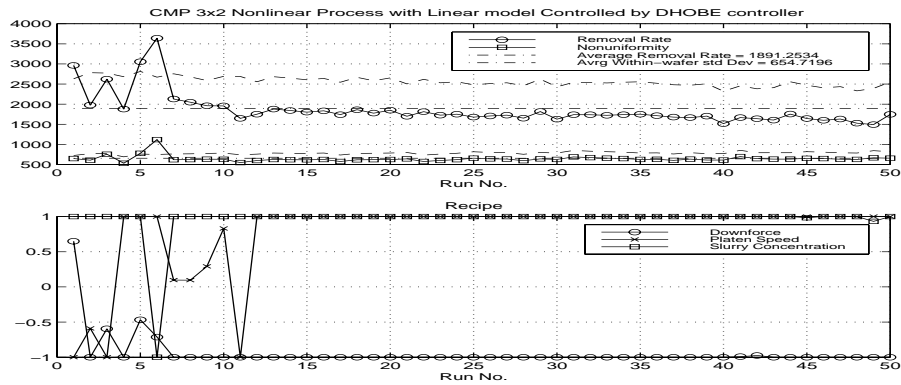


Figure 19: CMP3x2 Scenario 2 Controlled by the DHOBE-MR Controller

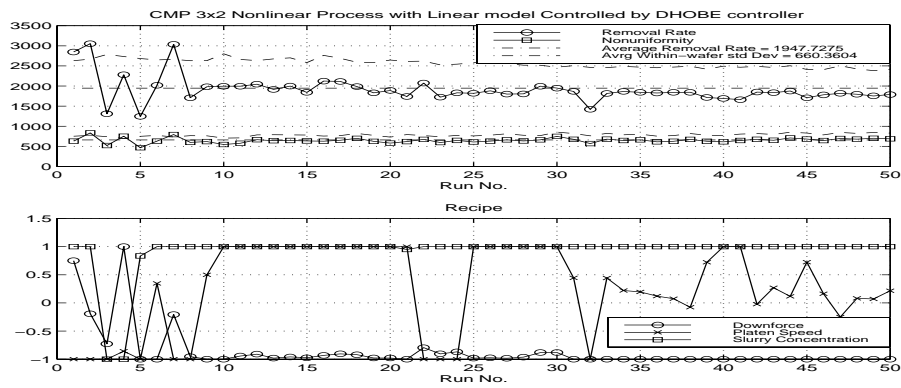


Figure 20: CMP3x2 Scenario 2 Controlled by the DHOBE-SV Controller

Scenario	Method	\bar{y}_1	\bar{y}_2	S_{y1}	S_{y2}	MSD_1	MSD_2
1	OAQC	2069.9	478.8	143.8	53.5	193.5	95.0
	DHOBE-MR	2005.5	490.5	139.7	41.2	235.9	98.6
	DHOBE-SV	2002.8	490.4	141.4	42.7	238.9	98.9
2	OAQC	1950.4	595.0	430.7	99.6	543.9	220.4
	DHOBE-MR	1921.5	663.9	457.0	99.9	568.4	271.9
	DHOBE-SV	1921.8	659.6	381.6	71.6	499.0	256.8

Table 5: Performance Measure of the OAQC Method, the DHOBE-MR and the DHOBE-SV Controllers for CMP3x2 Models

The simulation results are shown in Figure 19 and Figure 20.

The final results with regards to the statistical variance analysis were listed in Table 5. Still we can see that the performance of the three controllers are comparable to each other.

From Table 5 it can be seen that when using the nonlinear model as the controller’s model to compensate for the severe nonlinear processes, the mean value of response, standard deviation and mean square deviation are in the acceptable range. But when the internal model is linear, the compensation result is not good as expected. This also illustrates that the nonlinear internal model is necessary for such kind of cases no matter for which kind of control method.

4 Conclusions

We introduced two set-valued RtR control schemes based on the DHOBE algorithm. Applying the DHOBE algorithm reduces significantly the computation load for estimation. The only knowledge required from the real process is the strict bound of the noise instead of the detailed statistical properties of the noise. Moreover the rescue procedure ensures that under large disturbances the controller will return the process to target in a short number of runs. Finally, the control schemes in this paper are easy to implement. The DHOBE-MR controller is much simpler than the DHOBE-SV controller in general. Therefore in most cases we recommend the DHOBE-MR controller.

By simulation, it shows that the two controllers are robust to model and sensor errors. In comparison with the EWMA controller and the OAQC controller, they have comparative or better performance under various conditions. Furthermore it was shown that it is necessary to apply nonlinear RtR controllers for severe nonlinear processes by simulation. Future work will be done on the improvement of the estimation algorithm and application of the controllers on real processes.

References

- [1] E. Sachs, A. Hu, and A. Ingolfsson, "Run by run process control: combining SPC and feedback control", *IEEE Trans. Semiconduct. Manufact.*, vol. 8, pp. 26-43, Feb. 1995.
- [2] E. D. Castillo and J. Y. Yeh, "An Adaptive Run-to-Run Optimizing Controller for Linear and Nonlinear Semiconductor Processes", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 11, No. 2, pp. 285-295, May 1998.
- [3] A. Ingolfsson and E. Sachs, "Stability and sensitivity of an EWMA controller", *Journal of Quality Technology*, vol. 25, pp. 271-287, 1993.
- [4] J. S. Baras, N. S. Patel, "Designing Response Surface Model-Based Run-by-Run Controllers: A Worst Case Approach", *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 19, April 1996.
- [5] Z. Ning, etc, "A comparative analysis of run-to-run control algorithms in the semiconductor manufacturing industry", 1996 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, pp. 375-381, 1996.
- [6] M. Hankinson, T. Vincent, K. Irani, and P. Khargonekar, "Integrated real-time and run-to-run control of etch depth in reactive etching," *IEEE Trans. Semiconduct. Manufact.*, vol. 10, pp. 121-130, Feb. 1997.
- [7] D. Boning, W. Moyne, T. Smith, etc., "Run by run control of chemical-mechanical polishing," 1995 IEEE/CPMT Int'l Electronics Manufacturing Technology Symposium, pp. 81-87, 1995.
- [8] H. Deng "Run to Run Controller for Semiconductor Manufacturing," Master Thesis, 1999.
- [9] M. F. Cheung, S. Yurkovich and K. M. Passino, "An Optimal Volume Ellipsoid Algorithm for Parameter Set Estimation", *Proceedings of the 30th Conference on Decision and Control*, 1991.
- [10] C. Zhang, H. Deng and J. S. Baras, "The Set-Valued Run-to-Run Controller in Semiconductor Manufacturing Processes", technical report, 1999.
- [11] E. Fogel and Y. F. Huang, "On the value of information in system identification-bounded noise case", *Automatica*, vol. 18, no. 2, pp. 229-238, Mar. 1982.
- [12] S. Dasgupta and Y.F.Huang, "Asymptotically Convergent Modified Recursive Least-Squares with Data-dependent Updating and Forgetting Factor for Systems with Bounded Noise", *IEEE Transactions on Information Theory*, vol IT-33, No. 3, p383-392, May 1987.
- [13] A. K. Rao, Y. Huang, "Tracking characteristics of an OBE parameter-estimation algorithm," *IEEE Trans. Signal Processing*, vol. 41, no. 3, pp. 1140-1148, March 1993.

- [14] S. G. McCarthy and R. B. Wells, "Model order reduction for optimal bounding ellipsoid channel models", IEEE Trans. on Magnetics, vol. 33, no. 4, pp. 2552-2568, July 1997.

"The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government."