Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

REAL-TIME DATA ASSIMILATION

Shoko Suzuki

IBM Research - Tokyo
1623-14 Shimotsuruma
Yamato-shi, Kanagawa 242-8502, Japan

Takayuki Osogami

IBM Research - Tokyo
1623-14 Shimotsuruma
Yamato-shi, Kanagawa 242-8502, Japan

ABSTRACT

We investigate the idea of using the information obtained by observing a real system while simulating the real system to improve the accuracy of a prediction about the real system made based on the result of the simulation. Our approach runs multiple simulators simultaneously in parallel, where parameters of a simulation model are varied across the simulators. Based on the observation from a real system, some of the simulators are replicated, and others are terminated. We verify the effectiveness of our approach with numerical experiments. In addition, we provide a theoretical justification for our approach, using kernel density estimation.

1 INTRODUCTION

Simulation is an effective way to make predictions of near future for complex systems, including transportation systems (Nagel, Esser, and Rickert 1999, Cetin, Nagel, Raney, and Voellmy 2002) and atmospheric systems (Molteni, Buizza, Palmer, and Petroliagis 1996, Toth and Kalnay 1997). The quality of the predictions depends on the accuracy of the simulation model and the values of the parameters in the simulation model. The accuracy of the values of the parameters can be improved with data assimilation (Lewis, Lakshmivarahan, and Dhall 2006, Evensen 1994, Evensen 2009, Hunt, Kalnay, Kostelich, Ott, Patil, Sauer, Szunyogh, Yorke, and Zimin 2004), where predictions with simulation are compared against what are realized in the corresponding real system, and the values of the parameters are adjusted accordingly. (Craig, Goldstein, Rougier, and Seheult 2001, Goldstein and Rougier 2004) formalized Bayesian forecasting using simulators where the forecasts are adjusted by the past information of the real system. Notice, however, that simulation of the complex systems are time-consuming particularly when a detailed simulation model is constructed to make a prediction of high accuracy. As a result, the information used to fine-tune the parameters of a simulation model or to adjust the forecasts might be outdated, or additional information might become available, by the time predictions are made based on the results of the simulation. In this paper, we investigate the new idea of incorporating the information that becomes available after simulation starts into the simulation model to improve the quality of the predictions.

We consider Monte Carlo simulation of large scale systems. An example of such simulation is multi-agent simulation (Shoham and Leyton-Brown 2008), including transportation systems (Nagel, Esser, and Rickert 1999, Cetin, Nagel, Raney, and Voellmy 2002) and financial systems (Raberto, Cincotti, Focardi, and Marchesi 2001). Another example of large-scale Monte Carlo simulation is in ensemble forecasting of the weather (Molteni, Buizza, Palmer, and Petroliagis 1996, Toth and Kalnay 1997, Epstein 1969, Lorenz 1969). Notice that mesoscopic simulation of the weather (Treinish and Praino 2004) is quite time-consuming even with today's supercomputers. In general, a real system has many hidden variables that can be neither observed nor explicitly modeled in simulation. Such hidden variables are often modeled as random noise in simulation. As such, the class of Monte-Carlo simulation considered in this paper is quite broad.

Because Monte Carlo simulation is stochastic, multiple simulation runs with different random seeds are needed to make a predication of high accuracy and to provide a statistical guarantee on the accuracy of the

prediction. We will exploit this structure of multiple simulation runs to incorporate the latest information into the results of simulation. The multiple simulations might be run in parallel or in sequence, but we assume that sufficient computational resources are available to run the simulations in parallel.

The key idea in our approach is to run simulations in parallel not only with different random seeds but also with different values of parameters. Depending on the information that becomes available while the simulations are run, we will terminate some of the simulation runs and make independent copies of some of the remaining simulation runs. The independent copies of a simulation run will typically use the same values of parameters but different random seeds. Other ways are also possible, for example we can use the value of parameters slightly differs from the original simulator.

Observe that the information obtained after the simulations start might indicate that some of the values of parameters are more likely to represent the latest conditions of the real system than the others. Then we should terminate those simulation runs that are based on the values of parameters that do not well represent the real system and instead allocate more computational resources to other simulation runs by making their independent copies. The independent copies will enhance the level of statistical guarantee on the prediction made with the simulation runs based on the values of parameters that appear to well represent the latest conditions of the real system. Besides, this reselection step can be viewed as parameter-tuning.

Terminating a simulation run does not always pay off, however, because all of the computational effort devoted to a simulation run would be wasted. The results with a simulation run can provide useful information even if the particular values of the parameters used by the simulation run do not exactly represent the conditions of the real system. It is hence nontrivial which simulation runs should be terminated when particular information becomes available.

We propose a specific algorithm that determines which simulation runs should be terminated and which simulation runs should be used to replicate independent copies given the information that has become available while the simulations are run (see Section 2). Our numerical study shows that the proposed algorithm is effective in making predictions of high accuracy (see Section 3). The proposed algorithm and the support for its effectiveness constitute the primary contribution of this paper. A secondary contribution of this paper is a theoretical justification of the proposed algorithm. We will provide rationale for key steps of the proposed algorithm, using kernel density estimation (see Section 4).

Note that the prior work such as (Craig, Goldstein, Rougier, and Seheult 2001, Goldstein and Rougier 2004) has investigated the idea of using the information obtained by observing a real system to calibrate the predictions made based on the results of simulation. However, in the prior work, the observation of the real system was made *before* running the simulation for making prediction about future. On the other hand, in this paper, we investigate the idea of using the observation of the real system made *during* the simulation. Using such latest information from the real system, we select simulation runs to be terminated or to be copied, so that the predictions can be made based on the results of simulation runs that are more likely to be close to the real system than others. We also use the observation from the real system to calibrate the predictions in a way similar to the prior work. In the calibration, however, we assume no specific parametric model of a real system for robustness. Our non-parametric approach is in contrast to, for example, (Craig, Goldstein, Rougier, and Seheult 2001), where the calibration is made based on the assumption that simulation output form a Gaussian process. As a result, their approach would not be effective when an unexpected change happens in the real system or more simply when the real system cannot be well modeled with a Gaussian process.

2 PROPOSED APPROACH

Consider a Monte-Carlo simulator that simulates a real system in super-real time. Specifically, the simulator runs $N > 1$ times faster than the real system. The results with the simulator depend on the particular values of specifiable parameters, θ, as well as on the particular realizations of random numbers inherent to the simulator.

Let $S(t)$ be the vector of characteristic values of the state, which we will refer to as the characteristic vector, of a simulator at simulation time t. Here, simulation time records the time that has passed in the simulation since the simulation started. Namely, simulation time is Nt when time t has passed in the real world since the simulation started. For simplicity, we assume that real time is zero when simulation time is zero. Notice that $S(t)$ is a random vector for Monte-Carlo simulation. For example, in the case of traffic simulation, $S(t)$ might represent the number of vehicles in each region at simulation time t, where the simulated space is classified into multiple regions.

Simulation is run until simulation time reaches T, and $S(T)$ is used to make a prediction about the real system at time T, where observe that the prediction can be made at time $T/N < T$. Notice that $S(T)$ depends on the parameters θ that are estimated based on the information available only before simulation starts. For simulation of a complex system with a detailed model, N is often small, and the θ might be outdated at real time T/N, when the prediction is made.

In this section, we present our algorithm for using the latest information to improve the accuracy of the prediction. In Section 2.1, we discuss the general framework of our algorithm. Key elements of the algorithm are specified in Section 2.2 and Section 2.3.

2.1 General Framework of Our Algorithm

In our proposed approach, we run $k > 1$ simulators simultaneously in parallel. The simulator differs only in their parameters. For $i = 1, \ldots, k$, let θ_i be the values of the parameters for the i-th simulator. Notice that we can estimate the values of true parameters based on available information. When we have a high confidence about the estimation, we would set each θ_i close to the estimated values. Otherwise, we would set a few of θ_i close to the estimated values and set the others differently.

At some real-time t_0, we observe the real system. Let $s(t_0)$ be the characteristic vector of the observed state. Notice that $s(t_0)$ has the same semantics as the corresponding vector, $S(t_0)$, for a simulated system. The observation might have been scheduled or triggered by some external conditions.

We make sure that the characteristic vector $S_i(t_0)$ has been recorded for each i at real time $t_0/N < t_0$. When the observation at time t_0 is scheduled, it is straightforward to record $S_i(t_0)$. Otherwise, $S_i(t)$ can be recorded with sufficiently small intervals, so that $S_i(t_0')$ is recorded for $t_0' \approx t_0$. In the following, we assume that the observation is scheduled for simplicity.

Following the observation at real time t_0, we apply a step of reelecting simulation runs. For the purpose of reselection, we will introduce weight w_i for each simulator i. Each w_i is determined based on the distance, $d(s(t_0), S_i(t_0))$, between $s(t_0)$ and $S_i(t_0)$. In general, the closer $S_i(t_0)$ is to $s(t_0)$, the larger the w_i is. Then we normalize w_i such that

$$\sum_{i=0}^{k} w_i = 1. \tag{1}$$

We provide specific examples of the distance functions and the corresponding weights in Section 2.2, where we also discuss examples of weights that depend not only on $d(s(t_0), S_i(t_0))$ but also on other information about the simulators. Now we take k independent samples from the distribution that assigns probability w_i on simulator i for $i = 1, \ldots, k$. Note that samples are taken with replacement, so that some of the simulators are selected multiple times, and others are not selected at all. We represent these selected simulators with $\{i^* : i = 1, \ldots, k\}$.

Then we repeatedly copy all of the relevant information about a simulator to another, deleting the original information of the latter simulator, until the set of k simulators becomes $\{i^* : i = 1, \ldots, k\}$. After reordering, we now have k simulators such that the i-th simulator has the parameters, θ_{i^*}, and the characteristic vector, $S_i(t_0) = S_{i^*}(t_0-)$, where $S_{i^*}(t_0-)$ denotes the characteristic vector of simulator i^* immediately before the reselection is conducted.

Each simulator i is resumed from the state having the characteristic values, $S_i(t_0) = S_{i^*}(t_0-)$, using the parameters, θ_{i^*}, and independent random numbers. For some i and j, i^* and j^* denote the same simulator,

so that $S_i(t_0) = S_j(t_0)$. However, this does not mean that $S_i(t) = S_j(t)$ for $t > t_0$, because the two simulators use independent random numbers.

Observe that the reselection step eliminates some of unreliable simulation runs and replicate some of reliable ones in the following sense. Simulator i is more likely to be reliable than simulator j, if $S_i(t_0)$ is closer to $s(t_0)$ than $S_j(t_0)$ is. Although the result from a Monte-Carlo simulation is random, the closeness between $S_i(t_0)$ and $s(t_0)$ is an indicator of the reliability of simulator i. Hence, the simulator i having a large weight w_i is more likely to be reliable than the simulators having small weights. As a result, we are likely to select reliable simulators more often than the others. Note that this step works as parameter tuning as well as the tuning of condition $S(t_0)$.

We can perform the reselection step multiple times before simulation time reaches T. Multiple reselection steps can increase the accuracy of the prediction.

At real time T/N, we make a prediction about a real system based on the results from simulations, specifically $S_i(T)$ for $i = 1,\ldots,k$. We provide an example of an estimator of $s(T)$ in Section 2.3.

One of the important features of our algorithm is that we apply nonparametric approach; no assumptions are made for real system. Another importance feature is that reselection step can be taken whenever possible. These become effective especially when unexpected changes (e.g. traffic accsident) happen in real-system. If we apply reselection step right after the changes, the prediction at later time wouldn't far apart from the real situation.

2.2 Examples of Weights

In this section, we provide examples of weight w_i. A simple example of weight w_i is

$$w_i \propto \exp\left(-\frac{d(s(t_0),S_i(t_0))}{\sigma^2}\right), \tag{2}$$

for a distance function d and a constant σ. When the distance in Equation (2) is the Euclidean distance, this weight is proportional to a Gaussian function.

It turns out that the following more general form can provide more robustness than Equation (2) and allows us to make more accurate predictions in our proposed algorithm:

$$w_i \propto \sum_{j=1}^{k} f_1\left(d_1(S_i(t_1),S_j(t_1))\right) f_0\left(d_0(s(t_0),S_j(t_0))\right) f_\theta\left(d_\theta(\theta_i,\theta_j)\right) \tag{3}$$

where $t_1 \equiv Nt_0$; f_1, f_θ, and f_0 are decreasing functions; each of d_1, d_θ, and d_0 denotes the distance between the two arguments. For example, we can use the following weight:

$$w_i \propto \sum_{j=1}^{k} \exp\left(-\frac{d_1(S_i(t_1),S_j(t_1))}{{\sigma_1}^2} - \frac{d_0(s(t_0),S_j(t_0))}{{\sigma_0}^2} - \frac{d_\theta(\theta_i,\theta_j)}{{\sigma_\theta}^2}\right) \tag{4}$$

for constants σ_1, σ_θ, and σ_0. We will investigate the weight of the form (4) in the following sections.

An intuition behind superiority of Equation (3) and Equation (4) is that, unlike Equation (2), these equations take into account the following possibilities. First, even if $S_i(t_0)$ is not close to $s(t_0)$, simulator i might be reliable if θ_i is close to θ_j and $S_j(t_0)$ is close to $s(t_0)$. Likewise, simulator i might be reliable if its latest characteristic vector $S_i(t_1)$ is close to $S_j(t_1)$ and $S_j(t_0)$ is close to $s(t_0)$. Also, even if $S_i(t_0)$ is close to $s(t_0)$, simulator i might not be reliable if θ_i is close to θ_j (or $S_i(t_1)$ is close to $S_j(t_1)$) and $S_j(t_0)$ is far from $s(t_0)$. In Section 3, we verify the effectiveness of Equation (4). And in Section 4, we describe the theoretical justification of Equation (3) using Bayes' theorem.

Note that Equation (3) reduces to Equation (2) if f_1 and f_θ are delta functions, and f_0 is an exponential function. Specifically, we obtain Equation (2) from Equation (3) by letting $f_1(d(S_i(t_1),S_j(t_1)))$ be 1 if $i = j$ and 0 otherwise, letting $f_\theta(d(\theta_i,\theta_j))$ be 1 if $i = j$ and 0 otherwise, and letting $f_0(x) = \exp(-x/\sigma^2)$ for $x = d_0(s(t_0),S_j(t_0))$. Also, Equation (4) is a generalization of Equation (2) because in general $\exp(-x^2/\sigma^2)$ approaches asymptotically to delta function $\delta(x)$ as $\sigma \to 0$.

2.3 Example of Prediction

In this section, we present examples of how to predict $s(T)$ from simulation results. At real time T/N, we obtain $S_i(T)$ for $i = 1,\ldots,k$. A simple way of predicting $s(T)$ is the ensemble average:

$$\frac{1}{k}\sum_{i=1}^{k} S_i(T). \tag{5}$$

We find that this simple ensemble average can lead to a sufficiently accurate prediction for some applications, because we have already discarded unreliable simulators and replicated reliable ones at the reselection step. Specifically, we will see in Section 3 the results of numerical experiments, where the prediction with the ensemble average and the reselection step has higher accuracy that that without the reselection step.

The prediction with (5) might not always be sufficiently accurate. If the inaccuracy stem from the fact that observation of real system is outdated by the time prediction is made, one could perform another reselection step when the prediction is made (i.e., obtain $s(T/N)$ at real time T/N). Then we calculate the weights using $s(T/N)$ and $S_i(T/N)$, following the instructions in Section 2.2 (particularly, Equation 3). After the reselection with these weights, the ensemble average is calculated using (5), which we use as a prediction of $s(T)$. The reselection incorporates the updated real time situation $s(T/N)$ into the prediction and can improve the accuracy of the prediction.

3 NUMERICAL EXPERIMENTS

Now we will study the effectiveness of our proposed approach with numerical experiments. To isolate the effect of the proposed approach from the complications that can arise from simulation, we consider a simple model of simulation.

Specifically, let the characteristic vector $S(t)$ from simulation be a one-dimensional geometric Brownian motion. That is,

$$S(t) = \exp\left(\mu t + \sigma W_t\right) \tag{6}$$

for $t \geq 0$, where W_t represents the Wiener process that has the following probability density function:

$$f(x) = \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{x^2}{2t}\right) \tag{7}$$

for $-\infty < x < \infty$. Here, μ and σ are parameters of the simulator. We assume that the characteristic vector $s(t)$ of a real system is normalized such that

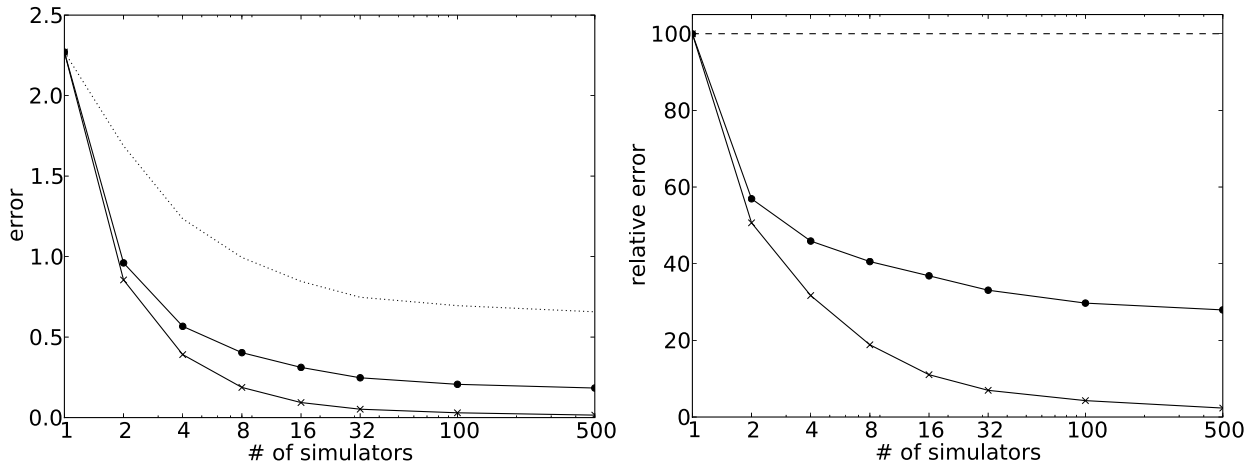
$$s(t) = 1 \tag{8}$$

for $t \geq 0$.

Our interpretation is that σ represents the volatility of the results from the simulator, which is inherent to the simulation model under consideration. We will fix σ at particular values in our experiments. On the other hand, μ represents the accuracy of the estimated parameters of the simulation model. Namely, if the simulation model has a parameter θ, and its estimated value $\hat{\theta}$ is used in the simulation, then we have $\mu = \hat{\theta} - \theta$.

We evaluate the error that our approach and standard approach, respectively, have in predicting $s(T)$. For each data point, we repeat predicting $s(T)$ multiple times and calculate the average L2 error. Specifically, let $\hat{s}_i(T)$ be the prediction by a particular approach in the i-th run. Then we define

$$\text{error} = \sqrt{\frac{1}{M}\sum_{i=1}^{M} \left(s(t) - \hat{s}_i(T)\right)^2}, \tag{9}$$



(a) error (b) relative improvement

Figure 1: Effectiveness of our approach with simple weight, where $N = 2$, $T = 1.0$, and $\sigma = 0.01$. (a) Error with standard approach (dotted line) and error with the proposed approach (solid line). (b) Relative error against the standard approach.

where M is the number of runs that are used to calculate the average. We will specify M in each experiment.

The first set of experiments study the error in our approach, using the simple weight introduced in Equation (2). Specifically, in this experiment, we set

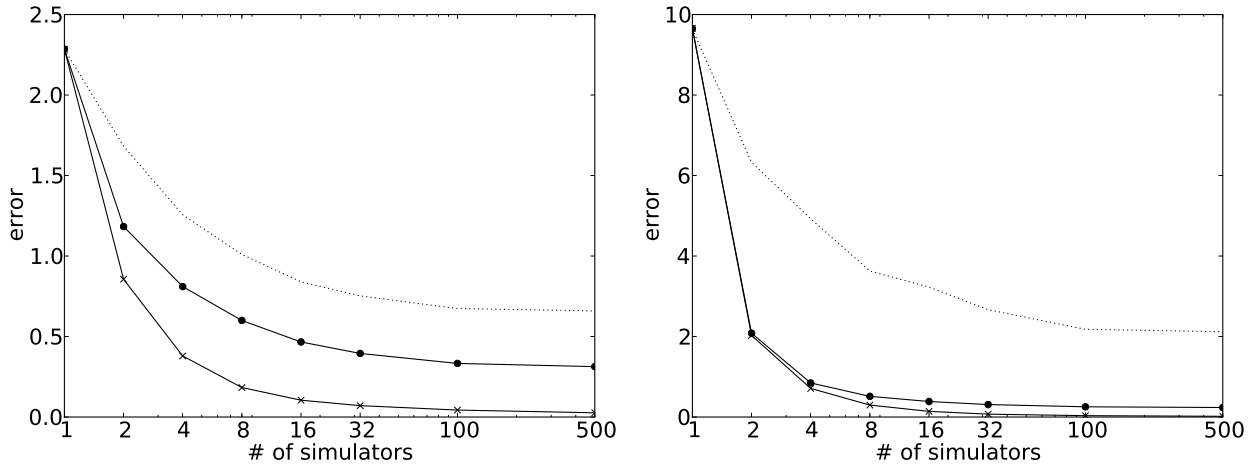
$$w_i \propto \exp\left(-\frac{|\log(s(t_0)) - \log(S_i(t_0))|^2}{\sigma_w^2}\right), \tag{10}$$

where $\sigma_w^2 = 0.01$. We find that our results are not very sensitive to the particular value of σ_w^2.

Figure 1(a) shows the error in the prediction made with three different approaches. Here, we set $T = 1.0$ and $N = 2$, so that the simulator runs twice as fast as the real system. We fix $\sigma = 0.01$ and generate μ from the standard normal distribution. Note that μ is generated independently for each run and for each of the simulators running in parallel. Two solid lines (one with crosses and the other with circles) show the error with our approach, where the horizontal axis shows k, the number of simulators running in parallel. The solid line with crosses performs the reselection step at the time when the prediction is made (i.e., $t_0 = T/N$). The solid line with circles performs reselection earlier (specifically, $t_0 = 0.2T/N$). In both cases, the prediction is made with the ensemble average of the k simulators as shown in Equation (5). The dotted line show the error with the standard approach of running k simulators in parallel with no reselection step, where the prediction is made with the ensemble average of the k simulators. Here, the error given by Equation (9) is calculated with $M = 100,000/k$, which allows us to evaluate the error with sufficiently high accuracy for our purposes.

Observe in Figure 1(a) that the error tends to decrease with k both with the proposed approach and with the standard approach, but the error with the proposed approach (solid) decreases faster than that with the standard approach (dotted). The error with our proposed approach (solid with crosses) can be made smaller than 0.5 using only $k = 4$ simulators, while the standard approach (dotted) has the error greater than 0.5 even with $k = 500$. To take a closer look, Figure 1(b) shows the relative error, where the error with the standard approach is normalized at 100. In particular, when $k = 500$, we find that the error with the standard approach (dotted) is 42 times larger than that with our proposed approach (solid with crosses).

In Figure 1, we can also see that the error can be made smaller by performing the reselection step at the time when the prediction is made (solid with crosses) than by performing the reselection step earlier (solid with circles). In particular, when $k = 500$, the error with $t_0 = 0.2T/N$ is 12 times larger than that with $t_0 = T/N$.



(a) $N = 3$, $T = 1.0$ (b) $N = 3$, $T = 1.5$

Figure 2: Sensitivity of the error with our approach to N and T.

Figure 2 shows the sensitivity of our results to the particular values of N and T. The settings for Figure 2 are the same as those for Figure 1 except that we set $N = 3$ in Figure 2(a) and $T = 1.5$ in Figure 2(b).

Observe that the error with our approach increases when we increase the speed of the simulator from $N = 2$ (Figure 1(a)) to $N = 3$ (Figure 2(a)). The increase in the error stems from the fact that the faster simulators ($N = 3$) make observation earlier (specifically, $t_0 = T/3$ or $t_0 = 0.2T/3$) than the slower simulators ($N = 2$) in our settings. So that the results with the slower simulators are based on more recent observation than the faster simulators. Notice that the results of the simulation are obtained more quickly when we use the faster simulator (i.e., $N = 3$) rather than the slower simulator. Hence, one should not conclude that the slower simulator is superior to the faster simulator simply because the slower simulator has the smaller error in this experiment.

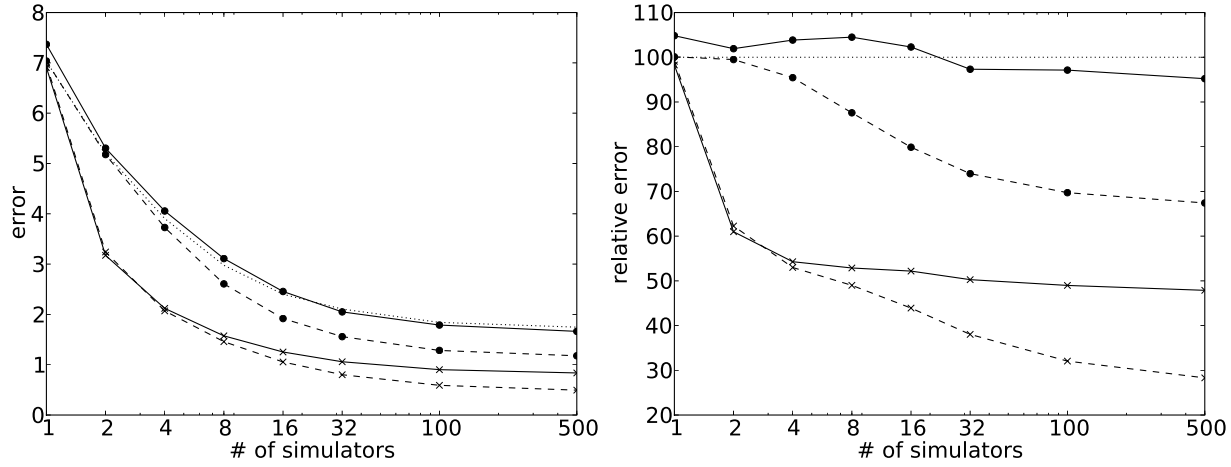
When T is large (Figure 2(b)), the accurate prediction becomes difficult with the standard approach (dotted). Notice that the value of the error with the standard approach shown in Figure 2(b) is significantly larger than those shown in Figure 1 and Figure 2(a). On the other hand, our approach (solid) successfully reduces the error particularly with $k \geq 4$.

When the volatility, σ, of the simulator is high, the accurate prediction can become difficult even with our approach. The second set of experiments verifies the effectiveness of the weight of the form shown in Equation (4) in this difficult case. Specifically, we consider a sophisticated weight as follows:

$$w_i \propto \sum_{j=1}^{k} \exp\left(-\frac{|\log(S_i(t_1)) - \log(S_j(t_1))|^2}{{\sigma_1}^2} - \frac{|\log(s(t_0)) - \log(S_j(t_0))|^2}{{\sigma_0}^2} - \frac{|\mu_i - \mu_j|^2}{{\sigma_\mu}^2}\right) \tag{11}$$

with ${\sigma_0}^2 = 1$, ${\sigma_1}^2 = 0.1$ and ${\sigma_\mu}^2 = 3$. Before we choose these particular values of ${\sigma_0}^2$, ${\sigma_1}^2$, and ${\sigma_\mu}^2$, we have tried several sets of values and found that these values result in smaller error than the others. Our approach would benefit from a systematic technique for tuning these values.

In Figure (3), we use the same settings as those in Figure 1 except that we now set $\sigma = 1.0$. Similar to Figure 1, the dotted line shows the error with the standard approach, and the two solid lines (one with circles and the other with crosses) show the error with the proposed approach. The two dashed lines show the error with the proposed approach using the sophisticated weight shown in Equation (11). Notice that the reselection step is performed at the time when the prediction is made (i.e., $t_0 = T/N$) for the dashed line with crosses, and earlier (i.e., $t_0 = 0.2T/N$) for the dashed line with circles. Here, the error given by Equation (9) is calculated with a large value of $M = 1,000,000/k$ to evaluate the error with sufficiently high accuracy for our purposes.



(a) error (b) relative error

Figure 3: Effectiveness of the sophisticated weight, where $N = 2$, $T = 1.0$, and $\sigma = 1.0$.

Observe that the proposed approach with simple weight does not appear to have any advantage over the standard approach if the reselection step is performed at $t_0 = 0.2\,T/N$ (solid line with circles), although it is still effective if the reselection step is performed at $t_0 = T/N$ (solid line with crosses). The sophisticated weight can reduce the error in such cases where the simple weight is not effective. To take a closer look, Figure 3 shows the relative error, where the error with the standard approach is normalized at 100. Observe that the sophisticated weight can reduce the error up to 30% as compared to the simple weight.

The effectiveness of the sophisticated weight for a large σ may be understood as follows. With a large σ, the prediction of simulators has high variability, and fewer simulators have $S(t_0)$ that is close to $s(t_0)$. Then the other information such as θ_i and $S_i(t_1)$ becomes useful in evaluating the reliability of each simulator. In Section 4, we provide a theoretical justification for the sophisticated weight shown in Equation (4).

4 THEORETICAL JUSTIFICATION

In this section, we provide a theoretical justification for our approach, particularly our specific choices of the weight given in Section 2.2 and the estimator given in Section 2.3. To account for these choices, we assume that a realization of a characteristic vector, $s(t)$, of a real system is a sample from a random vector having a probability density function $p_{s(t)}(\cdot)$. The purpose of our simulation is to estimate

$$\mathsf{E}[s(T)] \equiv \int x\, p_{s(T)}(x)\, dx, \tag{12}$$

which can be used to predict $s(T)$.

A simulator is built to estimate $\mathsf{E}[S(T)]$. We assume that simulator i having parameter θ_i produces a characteristic vector, $S_i(t)$, having a probability density function $p_{S(t)}(\cdot \mid \theta = \theta_i)$ for $0 \le t \le T$. Prior to running simulation, we can examine the likelihood of θ. Let p_θ be the probability density function for the prior distribution of θ. If we pick a sample from the random θ having p_θ, then the characteristic vector, $S(T)$, of the simulator using the sample θ has the following expectation:

$$\mathsf{E}[S(T)] = \int\int x\, p_{S(T)}(x \mid \theta = z)\, p_\theta(z)\, dx\, dz = \int\int x\, p_{S(T),\theta}(x,\theta)\, dx\, dz, \tag{13}$$

where $p_{S(T),\theta}$ denotes the probability density function of $(S(T),\theta)$. Our approach can be justified when we assume that the marginal distribution of $S(T)$ agrees with the distribution of $s(T)$.

A naive approach would run k simulations with a common parameter, θ_0, to obtain k characteristic vectors, $S_i^{(\omega)}(T)$ for $i = 1,\ldots,k$. In this section, a sample from a random variable is denoted by appending (ω) at the upper right of the random variable. Then $\mathsf{E}[s(T)]$ is estimated with

$$\tilde{s}(T) \equiv \frac{1}{k}\sum_{i=1}^{k} S_i^{(\omega)}(T). \tag{14}$$

This naive approach might be justified if we can choose θ_0 such that $p_{S(T)}(\cdot \mid \theta = \theta_0) \equiv p_{s(T)}(\cdot)$. In this case, $\tilde{s}(T)$ would be an unbiased estimator of $\mathsf{E}[s(T)]$.

Unlike the naive approach, our approach runs k simulations with varying parameters, θ_i for $i = 1,\ldots,k$. This is because we do not have a complete knowledge about the parameter. Based on the prior knowledge about p_θ, we should choose those θ_i that has high value of $p_\theta(\theta_i)$ more likely than others. Notice, however, that we should adjust our belief about the likelihood of the parameters after obtaining the characteristic vector, $s^{(\omega)}(t_0)$, from the real system and comparing $s^{(\omega)}(t_0)$ to each $S_i^{(\omega)}(t_0)$. Namely, the simulators whose characteristic vectors are close to $s^{(\omega)}(t_0)$ are more likely to have the right parameters than the others.

4.1 A Rationale for Weight

We can now provide a rationale for our reselection step with weight given by Equation (4). The reselection step is performed at real time t_0, and, by that time, we have obtained the information $\mathscr{F}_{t_0}$ from the observation of a real system and from simulators. In particular, $\mathscr{F}_{t_0}$ includes $s^{(\omega)}(t_0)$, $S_i^{(\omega)}(t_0)$, and $S_i^{(\omega)}(t_1)$ for $i = 1,\ldots,k$. Given $\mathscr{F}_{t_0}$, we want to delete those simulators whose latest characteristic vectors have low likelihood and replicate other simulators.

More specifically, our approach seeks to select k simulators, allowing repetition, such that the probability that simulator i is selected is proportional to $p_{s(t_1),\theta}(S_i^{(\omega)}(t_1),\theta_i \mid \mathscr{F}_{t_0})$, where $p_{s(t_1),\theta}(\cdot|\mathscr{F}_{t_0})$ is the probability density function of $(s(t_1),\theta)$ given $\mathscr{F}_{t_0}$. Namely,

$$w_i \propto p_{s(t_1),\theta}(S_i^{(\omega)}(t_1),\theta_i|\mathscr{F}_{t_0}) \tag{15}$$

. We can show that the reselection using the weight given by Equation (4) corresponds to the case where $p_{s(t_1),\theta}(\cdot,\cdot \mid \mathscr{F}_{t_0})$ is estimated using a kernel density estimation with a Gaussian kernel.

To estimate $p_{s(t_1),\theta}(\cdot,\cdot \mid \mathscr{F}_{t_0})$, we let $\mathscr{F}_{t_0}^-$ denote the all of the information contained in $\mathscr{F}_{t_0}$ except the information that $s(t_0) = s^{(\omega)}(t_0)$. Then we have

$$\begin{aligned} p_{s(t_1),\theta}(\cdot,\cdot \mid \mathscr{F}_{t_0}) &= p_{s(t_1),\theta}(\cdot,\cdot \mid \mathscr{F}_{t_0}^-, s(t_0) = s^{(\omega)}(t_0)) &(16)\\ &= \frac{p_{s(t_1),s(t_0),\theta}(\cdot, s^{(\omega)}(t_0),\cdot \mid \mathscr{F}_{t_0}^-)}{p_{s(t_0)}(s^{(\omega)}(t_0) \mid \mathscr{F}_{t_0}^-)}, &(17)\end{aligned}$$

where $p_{s(t_1),s(t_0),\theta}$ denotes the probability density function of $(s(t_1),s(t_0),\theta)$. We can estimate $p_{s(t_1),\theta}(\cdot,\cdot \mid \mathscr{F}_{t_0})$ by first estimating $p_{s(t_1),s(t_0),\theta}(\cdot,\cdot,\cdot \mid \mathscr{F}_{t_0}^-)$ and using Equation (17). Notice that we need not estimate the constant $p_{s(t_0)}(s^{(\omega)}(t_0) \mid \mathscr{F}_{t_0}^-)$ in Equation (17), because we only need to know what $p_{s(t_1),\theta}(\cdot,\cdot \mid \mathscr{F}_{t_0})$ is proportional to (recall Equation (15)).

We assume that $s(t_0)$ is not deterministic even after we observe an $s(t_0)$ from the real system (i.e., the distribution of $s(t_0)$ given $\mathscr{F}_{t_0}$ does not have the support at a single point). Our interpretation is that the observation $s(t_0)$ has a random noise, so that $s^{(\omega)}(t_0)$ is a sample from the random $s(t_0)$.

In the method of kernel density estimation, density function $p(x)$ of every point x is estimated using sample data $x_1,x_2,\cdots,x_k$ by

$$p(x) = \frac{1}{k}\sum_{i=1}^{k} k(x,x_i), \tag{18}$$

where $k(x,x_i)$ is a kernel function that satisfies $\int_{-\infty}^{+\infty} k(x,x_i)dx = 1$. In general, $k(x,x_i)$ is large when x and x_i are close to each other. A popular kernel function is a Gaussian kernel:

$$k(x,x_i) \propto \exp\left(-\frac{|x-x_i|^2}{\sigma^2}\right), \tag{19}$$

where $|\cdot|$ is the Euclidean distance and σ is a specifiable parameter. When x and x_i are vectors, we can also let σ be a vector, in which case the multiplication and division in Equation (19) are defined component-wise. In our case, given the samples $\{(S_i^{(\omega)}(t_1), S_i^{(\omega)}(t_0), \theta_i) : i = 1,\ldots,k\}$, we can estimate $p_{s(t_1),s(t_0),\theta}(x,y,z \mid \mathscr{F}_{t_0})$ with the following Gaussian kernel:

$$k\left((x,y,z),(S_i^{(\omega)}(t_1), S_i^{(\omega)}(t_0), \theta_i)\right) \propto \exp\left(-\frac{|x-S_i^{(\omega)}(t_1)|^2}{{\sigma_{t_1}}^2} - \frac{|y-S_i^{(\omega)}(t_0)|^2}{{\sigma_{t_0}}^2} - \frac{|z-\theta_i|^2}{{\sigma_\theta}^2}\right). \tag{20}$$

By Equation (17), Equation (18), and Equation (20), we obtain

$$p_{s(t_1),\theta}(x,z \mid \mathscr{F}_{t_0}) \propto \frac{1}{k}\sum_{i=1}^{k} \exp\left(-\frac{|x-S_i^{(\omega)}(t_1)|^2}{{\sigma_{t_1}}^2} - \frac{|s^{(\omega)}(t_0)-S_i^{(\omega)}(t_0)|^2}{{\sigma_{t_0}}^2} - \frac{|z-\theta_i|^2}{{\sigma_\theta}^2}\right). \tag{21}$$

Observe that Equation (21) is a particular realization of Equation (4).

Kernel density estimation is not the only approach for density estimation, and other methods for density estimation can be used in our approach as long as $p_{s(t_1),s(t_0),\theta}(\cdot \mid \mathscr{F}_{t_0})$ can be estimated with the available information, including $\{(S_i^{(\omega)}(t_1), S_i^{(\omega)}(t_0), \theta_i) : i = 1,\ldots,k\}$. Another popular method of density estimation is k-Nearest-Neighbor (NN) density estimation method (Loftsgaarden and Quesenberry 1965). An advantage of Kernel density estimation over k-NN density estimation is computational efficiency. Kernel density estimation however might require fine-tuning of its parameters.

4.2 A Rationale for Estimator

Next, we provide a rationale for estimating $\mathsf{E}[s(T)]$ with the ensemble average (5). We consider the case where we estimate $\mathsf{E}[s(T)]$ immediately after the reselection step at real time t_0 (i.e., $T = t_1 = Nt_0$). Because we have obtained the information, $\mathscr{F}_{t_0}$, our goal is to estimate

$$\mathsf{E}[s(T)|\mathscr{F}_{t_0}] = \int x\, p_{s(T)}(x|\mathscr{F}_{t_0})\, dx. \tag{22}$$

Conditioning on and integrating over θ, we obtain

$$\mathsf{E}[s(T)|\mathscr{F}_{t_0}] = \int\int x\, p_{s(T),\theta}(x,z \mid \mathscr{F}_{t_0})\, dx\, dz. \tag{23}$$

Now, recall that we have estimated w_i in Section 4.1 so that it is proportional to $p_{s(T),\theta}(S_i^{(\omega)}(T), \theta_i \mid \mathscr{F}_{t_0})$. Assuming that $p_{s(T),\theta}(S(T), \theta \mid \mathscr{F}_{t_0})$ is estimated accurately, we can conclude that the ensemble average

$$\frac{1}{k}\sum_{i=1}^{k} S_{i^*}^{(\omega)}(T) \tag{24}$$

is an unbiased estimator of $\mathsf{E}[s(T)|\mathscr{F}_{t_0}]$.

Note that we apply the same formulation of Bayesian forecasting as (Craig, Goldstein, Rougier, and Seheult 2001) in Equation (22) apart from the critical difference that Craig et al. use this formulation with

past information that has been obtained before simulators start. Our approach and Craig et al. both estimate the posterior density $p_{s(T)}(x|\mathscr{F}_{t_0})$ in Equation (22). In (Craig, Goldstein, Rougier, and Seheult 2001), the posterior density function is estimated under the assumption of Gaussian distributions, while we estimated the posterior density with samples using nonparametric density estimation like Kernel density estimation. This makes the proposed algorithm general, especially appropriate for those complex systems like traffic simulation.

5 CONCLUSION

We have proposed an idea of terminating some of the simulation runs and replicating the others based on the information obtained during the simulation (i.e., real-time data assimilation). Using a simple model of simulation, we have shown that the accuracy of the prediction made with multiple runs of Monte Carlo simulation can be improved with a simple algorithm of real-time data assimilation. We expect that this paper initiates new lines of research toward an application of real-time data assimilation to improving the accuracy of predictions made with simulation of complex real systems, including predictions about weather and traffic conditions.

The simple algorithm studied in this paper has many deficiencies that need to be overcome when real-time data assimilation is applied to simulation of more complex systems. For example, the sophisticated weight (11) is justified with density estimation, which becomes difficult for a high dimensional space. Hence, when the vector $S(t)$ (respectively, θ) has a high dimension, we might want to apply dimensionality reduction to $S^{\omega}(t_1)$ (respectively, θ) before (11) is used. Notice that a high dimensional $S^{(\omega)}(t_0)$ can help in increasing the accuracy of prediction when the corresponding $s^{(\omega)}(t_0)$ is available, so that $S(t)$ can have a high dimension.

Our numerical experiments suggest that real-time data assimilation can tremendously improve the accuracy of predictions made with simulation when it is applied appropriately. Our expectation is that real-time data assimilation will become a key technique to make highly accurate predictions with simulation of complex systems after many issues about real-time data assimilation are addressed.

ACKNOWLEDGMENTS

This work was supported by "Promotion program for Reducing global Environmental loaD through ICT innovation (PREDICT)" of the Ministry of Internal Affairs and Communications, Japan.

REFERENCES

Cetin, N., K. Nagel, B. Raney, and A. Voellmy. 2002. "Large-scale multi-agent transportation simulations". *Computer Physics Communications* 147 (1-2): 559–564.

Craig, P. S., M. Goldstein, J. C. Rougier, and A. H. Seheult. 2001. "Bayesian forecasting for complex systems using computer simulators". *Journal of American Statistical Association* 96:717–729.

Epstein, E. S. 1969. "Stochastic dynamic prediction". *Tellus* 21 (6): 739–759.

Evensen, G. 1994. "Sequential data assimilation with a non-linear quasi-geostrophic model using Monte Carlo methods to forecast error statistics". *Journal of Geophysical Research* 99 (10): 143–162.

Evensen, G. 2009. *Data Assimilation: The Ensemble Kalman Filter*. 2nd ed. Springer.

Goldstein, M., and J. Rougier. 2004. "Probabilistic formulations for transferring inferences from mathematical models to physical systems". *SIAM Journal on Scientific Computing* 26 (2): 467–487.

Hunt, B. R., E. Kalnay, E. J. Kostelich, E. Ott, D. J. Patil, T. Sauer, I. Szunyogh, J. A. Yorke, and A. V. Zimin. 2004. "Four-dimensional ensemble Kalman filtering". *Tellus* 56A (4): 273–277.

Lewis, J. M., S. Lakshmivarahan, and S. Dhall. 2006. *Dynamic Data Assimilation: A Least Squares Approach*. Cambridge University Press.

Loftsgaarden, D. O., and C. P. Quesenberry. 1965. "A nonparametric estimate of a multivariate density function". *The Annals of Mathematical Statistics* 36:1049–1051.

Lorenz, E. N. 1969. "Atmospheric predictability as revealed by naturally occurring analogues". *Journal of the Atmospheric Sciences* 26:636–646.

Molteni, F., R. Buizza, T. N. Palmer, and T. Petroliagis. 1996. "The ECMWF Ensemble Prediction System: Methodology and validation". *Quarterly Journal of the Royal Meteorological Society* 122 (529): 73–119.

Nagel, K., J. Esser, and M. Rickert. 1999. "large-scale traffic simulations for transportation planning". In *Annual Reviews of Computational Physics VII*, 151–202. World Scientific Publishing Company.

Raberto, M., S. Cincotti, S. M. Focardi, and M. Marchesi. 2001. "Agent-based simulation of a financial market". *Physica A: Statistical Mechanics and its Applications* 299 (1-2): 319–327.

Shoham, Y., and K. Leyton-Brown. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

Toth, Z., and E. Kalnay. 1997. "Ensemble Forecasting at NCEP and the Breeding Method". *Monthly Weather Review* 125 (12): 3297–3319.

Treinish, L. A., and A. P. Praino. 2004. "Applications and Implementation of a Mesoscale Numerical Prediction and Visualization System". In *Proceedings of the 20th Conference on Weather Analysis and Forecasting / 16th Conference on Numerical Weather Prediction*.

AUTHOR BIOGRAPHIES

SHOKO SUZUKI is a researcher at IBM Research - Tokyo. She received her Ph.D. in Physics from the University of Tokyo in 2004. Her research interests include data mining, machine learning, text mining and statistics. Especially manifold regularization, dimensional reduction and regression regularization are her current focusing area. Her email address is e30126@jp.ibm.com.

TAKAYUKI OSOGAMI is an advisory researcher at IBM Research - Tokyo. He received his Ph.D. in Computer Science from Carnegie Mellon University in August 2005, and a B.Eng. degree in Electronic Engineering from the University of Tokyo in 1998. His research interests include building mathematical tools for analyzing and optimizing stochastic models using diffusion processes, point processes, Markov decision processes, Markov chains, phase-type distributions, extreme-value distributions, and other random objects. A primary focus of his current research is optimization of stochastic models over multiple periods with a consideration of risks. He is also investigating applications of semidefinite programming in the context of such stochastic optimization. His email address is osogami@jp.ibm.com.