Efficient dynamic Monte Carlo algorithm for time-dependent catalytic surface chemistry

V. Rai* and H. Pitsch
Department of Mechanical Engineering, Stanford University, Stanford, California 94305, USA

A. Novikov
Department of Mathematics, Pennsylvania State University, Pennsylvania 16802, USA
(Received 7 August 2006; published 18 October 2006)

Several numerical algorithms for dynamic Monte Carlo simulations of surface chemistry have been proposed in the past. The variable step size method (VSSM) is commonly used for systems where the rate coefficients are constant in time, owing to its good efficiency. If rate coefficients vary in time, the first reaction method (FRM) has been shown to be more efficient. However, the cost of this algorithm to execute a reaction step depends on the considered lattice size, which can make this method inefficient for systems involving surface phenomena on different scales. Here we propose a general and efficient algorithm, the fast first reaction method (fFRM), which has the advantages of being applicable to systems with constant and time-varying rate coefficients, and of having a computational cost per reaction step that is independent of the lattice size. An additional feature of fFRM is that it is rejection-free, which means that once a reaction class is selected, a reaction of that type will be executed. A rejection-free variant of VSSM, called rVSSM, is also presented, which leads to an approximately 15% speedup compared with the VSSM algorithm for the considered example.

DOI: 10.1103/PhysRevE.74.046707 PACS number(s): 02.70.−c, 82.65.+r, 82.20.Wt

I. INTRODUCTION

Surface catalysis is important in many practical chemical applications ranging from catalytic converters to all types of fuel cells [1,2]. In these catalyst-mediated chemical systems, bonding of reaction intermediates to the catalyst surface typically weakens chemical bonds and lowers reaction barriers [3–6]. It is well known that some of the adsorbates exhibit electronic interactions among each other of attractive or repulsive nature [7–9]. In such situations, the commonly used environment-averaged mean-field approach breaks down [10,11]. An accurate dynamical description of these complex chemical systems is made possible using stochastic simulations such as dynamic Monte Carlo simulations (DMC). DMC simulations enable the efficient solution of the master equation, which governs the time evolution and the dynamics of a chemical system. The DMC approach was first introduced by Gillespie [12] as "the stochastic simulation algorithm." Gillespie's method has subsequently undergone several improvements to make such simulations applicable to a wide range of surface chemistry systems [13–24]. DMC is also used widely for studying reactive systems in the solution phase with very small numbers of molecules [25]. It is noted here that in such applications, unlike surface chemistry DMC simulations, the locations where reactions may potentially take place are usually not important. Tracking the positions efficiently is critical to the performance of algorithms applied to study surface chemistry, and in this paper, we are concerned primarily with this class of simulations.

The most widely used DMC algorithm for the simulation of surface chemistry is the variable step size method (VSSM) [22]. Different variants of this method have been proposed for systems with both time-varying and constant rate coefficients. Especially if rate coefficients are constant in time, the VSSMb algorithm of Lukkien *et al.* [23], and the continuous time Monte Carlo (CTMC) method of Reese *et al.* [24] have been demonstrated to have remarkable performance. In particular, it was shown that for these algorithms the computational cost per time step is independent of the lattice size. However, both these algorithms cannot be applied for systems with time-varying rate coefficients [10,23], and the more general VSSM algorithm is very inefficient for this case [22]. The first reaction method (FRM) is the method of choice when the rate coefficients change over time. An inherent characteristic of FRM is that the CPU time required for the computation of a single time step depends on the size of the simulated catalyst surface, i.e., on the number of catalyst sites [23]. Thus, the performance of FRM deteriorates as the geometrical area of the catalyst surface is increased. This dependence can easily be limiting, when systems are studied that involve surface phenomena on different scales. However, the performance of FRM is still superior to VSSM for nonconstant rate coefficients.

Here we present a new algorithm, the fast FRM (fFRM) algorithm, for which the computation time of a single time step is independent of the catalyst lattice size, and which can be applied for cases involving both constant and time-varying rate coefficients. In essence, fFRM is a new algorithm with the computational efficiency of VSSMb and with the generality of FRM. Details of fFRM are presented after a brief description of the DMC approach in general, and the VSSMb and FRM algorithms, in particular. Subsequently, the accuracy of the new algorithm is assessed for a case with time-varying rate coefficients. Finally, the performance of fFRM is compared with VSSMb, FRM, and rVSSM, which is a modified version of VSSMb with slightly better performance based on the rejection-free feature of fFRM.

*Electronic address: varun@stanford.edu

II. DMC APPROACH

In simulations of surface chemistry, the catalyst surface is represented by a number of *sites*, where each site corresponds to a potential location of an adsorbate on the real catalyst surface. The complete specification of adsorbates present at every site describes a *configuration* or *state of the system*, and is denoted by the configuration vector $\mathbf{c}$. The probability that the lattice at time t appears in a certain configuration $\mathbf{c}$ is denoted by $P_{\mathbf{c}}(t)$. Each elementary reaction in the mechanism constitutes a unique *reaction type* or *microprocess*. Not all microprocesses have to be possible for each configuration, but if a microprocess can take place at some location on the catalyst surface in the current configuration, then it is referred to as an *enabled event*. An *executed event* changes a configuration into a different configuration, and is characterized by both of these configurations. The time evolution and the dynamics of this system are described by the so-called master equation

$$\frac{dP_{\mathbf{c}}(t)}{dt} = \sum_{\mathbf{c}' \neq \mathbf{c}} [k_{\mathbf{c}\mathbf{c}'} P_{\mathbf{c}'}(t) - k_{\mathbf{c}'\mathbf{c}} P_{\mathbf{c}}(t)], \tag{1}$$

where $k_{\mathbf{c}\mathbf{c}'}$ is the rate coefficient of the event that transforms configuration $\mathbf{c}'$ into configuration $\mathbf{c}$. The master equation is a differential equation for the probability $P_{\mathbf{c}}(t)$ of finding the system in a configuration $\mathbf{c}$, which is derived from first principles. $P_{\mathbf{c}}$ is a scalar function that has to be solved in D dimensions, where D is the number of possible configurations. Obtaining a direct solution of this equation for the general case is not possible for more than just a few sites. For example, even for a very small system of 2 species and a $3 \times 3(100)$ lattice, the number of possible configurations is 3^9, and the solution becomes intractable.

The master equation, therefore, is usually solved using Monte Carlo methods. Here we provide only a brief discussion of the fundamentals of such methods. Solution methods have been described in detail by Binder and Heermann [26], and issues specific to catalytic chemistry have been discussed in Refs. [23,24,27]. In accordance with the definition of a rate coefficient, the probability of a state transition $\boldsymbol{c} \rightarrow \boldsymbol{c}'$ in an infinitesimal time interval dt is kdt [12], where the subscripts have been dropped from $k_{\mathbf{c}'\mathbf{c}}$ for convenience. The absolute time of occurrence of the event (or the time of reaction) is therefore a random variable, denoted here by T_{ToR}. The probability that the event will occur after the time $t+dt$ is then equal to the probability that the reaction occurs after time t times the probability that the reaction will not occur in the time between t and $t+dt$. This can be written as

$$P(T_{\mathrm{ToR}} \geq t + dt) = P(T_{\mathrm{ToR}} \geq t)(1 - kdt), \tag{2}$$

where $P(\,)$ denotes the probability of the event in parentheses. Rearranging Eq. (2) gives

$$\frac{P(T_{\mathrm{ToR}} \geq t + dt) - P(T_{\mathrm{ToR}} \geq t)}{dt} = -kP(T_{\mathrm{ToR}} \geq t), \tag{3}$$

or

$$\frac{dP(T_{\mathrm{ToR}} \geq t)}{dt} = -kP(T_{\mathrm{ToR}} \geq t). \tag{4}$$

The term on the left-hand side of Eq. (4) is negative of the probability density function (pdf) of T_{ToR}. Let T_0 be the current system time. To determine the time increment $\Delta T = T_{\mathrm{ToR}} - T_0$ to the next transition, the probability $P(\Delta T \geq \Delta t)$ that this increment is larger than a certain value Δt can be obtained by the integration of Eq. (4), allowing k to be time dependent, which yields

$$P(\Delta T \geq \Delta t) = \exp\left[- \int_{T_0}^{T_0 + \Delta t} k(s) ds \right]. \tag{5}$$

This is equivalent to the probability that the reaction will not occur within Δt. For the case of constant k, Eq. (5) simplifies to

$$P(\Delta T \geq \Delta t) = \exp(-k\Delta t). \tag{6}$$

The cumulative distribution function (cdf) of ΔT is then

$$F(\Delta t) = 1 - P(\Delta T \geq \Delta t). \tag{7}$$

From Eq. (7), ΔT can be obtained according to the inverse transform method [23,28]. If U is a uniform random number in (0,1), then $P(U < u) = u$, and with F^{-1} being the inverse of F follows

$$P[F^{-1}(U) \leq \Delta t] = P[U \leq F(\Delta t)] = F(\Delta t). \tag{8}$$

Hence $F(\Delta t)$ can be represented by $F^{-1}(U)$, where U is drawn from a uniform distribution, which can be written as

$$F(\Delta T) = U. \tag{9}$$

Moreover, Lukkien *et al.* [23] have obtained the following theorem based on Eq. (7).

Theorem 1. If enabled reactions are selected with a probability such that their time of reaction satisfies the distribution defined in Eq. (7), then the stochastic trajectory of states generated satisfies Eq. (1) exactly.

In the next section, we will first discuss the VSSMb and FRM algorithms, and then a new efficient algorithm, fFRM, which all satisfy Eq. (1) through selecting enabled events with a probability as required by Theorem 1.

III. DMC ALGORITHMS

A. Event lists

An important distinction between DMC algorithms for surface chemistry simulations and algorithms for solution phase is that for surface chemistry, the reactants (adsorbates or empty catalyst sites) have a unique spatial position. A complete description of a microprocess therefore inherently implies, among other things, specification of its exact location. All the algorithms described in this paper accomplish this by using so-called *event lists*.

Event lists are arrays of structures that store information about the events that can take place for a given state of the system. Each element of the event list corresponds to a single event. In VSSMb and fFRM, events are grouped according

to their reaction type, and so there is a separate event list for each reaction type. A single event list is sufficient for FRM. Furthermore, the information stored for all individual events depends on the details of the particular DMC algorithm. In VSSMb and fFRM, for instance, the location and the absolute time when the event first became enabled are stored, while in FRM the location and T_{ToR} are stored. New events are enabled and old events are disabled at every step as a simulation proceeds to generate trajectories of system states. Keeping the event lists updated to reflect only enabled events requires adding the newly enabled events, but also searching the event lists for disabled events and removing these. This is computationally very expensive, and must be avoided. Lukkien *et al.* [23] showed that removal of disabled events is not necessary at all. Instead, once an event is chosen, it is sufficient to check if this event is still enabled before its execution. If not, the event will be rejected, removed from the list, and a new event is chosen. This procedure significantly improves performance, and has been used for all algorithms described in this paper. Another strategy that has been used for all simulations reported here is that of *local update*, wherein after the execution of each event, the search for newly enabled reactions is conducted only within the *interaction radius*. The interaction radius is the size of the neighborhood of a site, within which the species can either interact or pair up with the species at that site, considering all possible microprocesses. The advantage of local updates is that they make the computational time needed to search for newly enabled reactions independent of the lattice size. Even the local update can be made quite efficient. The execution of an event does not necessarily imply enabling or disabling of every other reaction type. Lukkien *et al.* [23] suggested that if this dependency is precalculated in advance of the simulation, then performance gains could be achieved. A general approach for generating such dependencies has been described by Gibson *et al.* [25]. Following this idea, a precomputed dependency list is used when conducting the local update for the simulations reported in this paper.

B. VSSMb

The VSSMb method is an efficient algorithm for most cases, which do not involve time-varying rate coefficients [23]. A simple description of this method is presented here for comparison with fFRM. A more detailed discussion can be found elsewhere [23,24].

Let $\mathbf{c}$ denote the current configuration, and $\boldsymbol{L}_i$ the event list for reaction type i. Furthermore, let n_i denote the number of enabled and disabled events in $\boldsymbol{L}_i$, and k_i the rate coefficient for reaction type i, respectively. We define

$$\Gamma_{\mathbf{c}}^{(i)} = n_i k_i, \quad \Omega_{\mathbf{c}} = \sum_i n_i, \quad \text{and} \quad \Gamma_{\mathbf{c}} = \sum_i \Gamma_{\mathbf{c}}^{(i)}, \tag{10}$$

where the summations are over all reaction types.

Algorithm 1. VSSMb.

(1) *Initialize*. Scan initial surface state to set up event list $\boldsymbol{L}_i$ for each reaction type i.

(2) *Select a reaction type to execute*. The jth reaction type is selected based on a uniform random number U_1 in (0,1) such that

$$\frac{\sum_{i=1}^{j-1} \Gamma_{\mathbf{c}}^{(i)}}{\Gamma_{\mathbf{c}}} \le U_1 < \frac{\sum_{i=1}^{j} \Gamma_{\mathbf{c}}^{(i)}}{\Gamma_{\mathbf{c}}}. \tag{11}$$

(3) *Locate an event of reaction type j at a suitable position*.

(a) An event is selected randomly from the event list $\boldsymbol{L}_j$.

(b) Check if the selected event is still enabled at the selected location. If not, reject, remove the event from $\boldsymbol{L}_j$, and go to step 5.

(4) *Execute the selected event and add newly enabled events of type i to the list* $\boldsymbol{L}_i$.

(5) *Increment system time*. The increment can be calculated using

$$\Delta T = -\frac{\ln(U_2)}{\Omega_{\mathbf{c}}\Gamma_{\mathbf{c}}}, \tag{12}$$

where U_2 is a uniform random number in (0,1).

(6) *Repeat steps* 2–5.

VSSMb is restricted to the case of constant rate coefficients, but the computation time per event is independent of the lattice size. An alternative formulation of VSSMb, very similar to the algorithm just described, uses the concept of *classes* wherein every event is classified based on the adsorbates in its microenvironment [24].

A more general formulation of VSSMb, the VSSM algorithm, which is also applicable to time-varying rate coefficients can be found in Ref. [22]. VSSM differs from VSSMb only in the way the time increment is calculated (step 5). Without going into detail, it suffices to mention here that in VSSM, instead of using Eq. (12), the value of the time increment Δt is computed by solving

$$\exp\left[-\int_{T_0}^{T_0+\Delta t} \Gamma_{\mathbf{c}}(s)ds\right] = U, \tag{13}$$

where the rate coefficients k_i, and hence $\Gamma_{\mathbf{c}}$, may vary with time. In general, solving Eq. (13) takes $O(N)$ CPU time [22], where N is the number of lattice sites. Hence, simulations using VSSM are very expensive as the computational time per time step depends on $O(N)$. A more efficient algorithm for systems with time-dependent chemistry is the FRM algorithm as proposed by Jansen [22]. This method is discussed next.

C. FRM

Experimental techniques like cyclic voltammetry (CV) and temperature-programmed desorption (TPD) are commonly used for the characterization of chemical systems. Simulations of such experiments based on detailed chemistry present a way for rigorous validation of chemistry models. A common feature of such experiments is that they involve chemistry with time-varying rate coefficients because of variations in electrode potential or temperature. In such cases, as mentioned before, VSSMb is not applicable [10]. However, the FRM algorithm is an efficient method for

simulating such systems [18,22]. FRM relies on explicitly storing every single enabled event on the surface along with the corresponding T_{ToR} in a list. The list is constructed as a so-called priority queue, which means that all events are sorted according to their execution time T_{ToR} in a tree-based data structure. The next event selected is then the one which is the nearest in time, i.e., the event with the smallest T_{ToR}.

Algorithm 2. FRM.

(1) *Initialize*.

(a) Scan initial surface state for enabled events.

(b) T_{ToR} for each event is generated according to Eq. (7).

(c) The position and T_{ToR} of enabled events are stored in a single list of events, $\boldsymbol{L}$.

(2) *Event selection*.

(a) Select the event with the smallest T_{ToR}.

(b) Check if the selected event is still enabled. If not, remove the event from $\boldsymbol{L}$, and go to step 5.

(3) *Update state information*.

(a) Execute the selected event.

(b) Add newly enabled events to $\boldsymbol{L}$. Use Eq. (7) for generating the new T_{ToR} values.

(4) *Change system time to* T_{ToR} *of the just-considered reaction*.

(5) *Repeat steps 2–4.*

A priority queue is used to store the events in $\boldsymbol{L}$ based on the T_{ToR}. Such a storage scheme ensures that the selection process [step 2(a)] takes $O(1)$ operations: the event with the smallest T_{ToR} is the first element in $\boldsymbol{L}$. Further, for priority queues, the computational cost of each event removal and insertion process encountered in steps 2(b) and 3(b), respectively, is $O(\log_2 H)$, where H is the number of events in $\boldsymbol{L}$ and it scales with the number of lattice sites N. A close look at FRM reveals that the computationally demanding step is the removal/insertion of events, which takes $O(\log_2 H)$ operations. It is noteworthy that for a complex chemical mechanism, more than one microprocess may be enabled per site, and so the number of events in the event list H can even be larger than the number of lattice sites N. Thus, the computation time per simulated reaction in FRM depends on the logarithm of the lattice size [23], a fact that is potentially limiting for simulating systems of large lattice size.

IV. NEW DMC ALGORITHM: fFRM

A. Overview

The key idea is based on the observation that reaction times for the events of the *same* reaction type are mutually independent and identically distributed (iid), and the distribution is given by Eq. (7). From the perspective of simulating surface chemistry, this means that for all $n_{\mathrm{en},j}$ enabled events of the same reaction type j (and stored in the event list $\boldsymbol{L}_j$), the minimum T_{ToR} can be directly computed, even in the most general scenario of time-dependent rate coefficients, as will be discussed below. The computational gain comes from the fact that, unlike FRM, it is no longer necessary to generate and sort T_{ToR} for every single enabled event. Instead, the smallest T_{ToR} can be found by first determining the smallest time of reaction for each individual reaction type, and then finding the minimum of these times. This can be written as

$$\min(T_{\mathrm{ToR}} \ \forall \ \text{enabled events}) = \min_{j=1,r}(T_{\mathrm{ToR},j}), \tag{14}$$

where r is the number of reaction types, $T_{\mathrm{ToR},j}$ is the minimum of the times of reaction among all enabled events of the reaction type j, and the minimum on the right-hand side of Eq. (14) is found over all reaction types. The reaction type corresponding to the minimum value on the right-hand side of Eq. (14) is selected for the next time step. Further, since the enabled events of a specific reaction type are iid, the location of the site where the next event will be executed can be selected randomly from the event list of that reaction type. This idea is the core of fFRM. Clearly, fFRM achieves exactly the same task as FRM, but more efficiently. Note that storing T_{ToR} for individual events in a priority queue is not required in fFRM. Instead, the $T_{\mathrm{ToR},j}$ values for each reaction type j are stored in a priority queue. Thus, as opposed to FRM, where computational time of every time step depends on $\log_2 H$, in fFRM computational requirements per time step depend on $\log_2 r$, thereby making the performance of fFRM independent of the lattice size. The problem of finding the minimum time for all events of the same reaction type is considered next.

All enabled events in $\boldsymbol{L}_j$ are iid, and hence, their T_{ToR} have the same cdf, say $F_j(x)$. Then $1-F_j(x)$ is the probability that T_{ToR} for a specific enabled event in $\boldsymbol{L}_j$ is greater than x. Using the independence of these events, $\{1-F_j(x)\}^2$ is the probability that T_{ToR} for two of these events is greater than x, and so on. This implies that $\{1-F_j(x)\}^{n_{\mathrm{en},j}}$ is the probability that $T_{\mathrm{ToR},j}$, the shortest reaction time in reaction class j, is larger than x. The cdf $F_{1,j}(x)$ of $T_{\mathrm{ToR},j}$ is therefore given as

$$F_{1,j}(x) = 1 - \{1 - F_j(x)\}^{n_{\mathrm{en},j}}. \tag{15}$$

A general proof for this relation based on order statistics is presented in the Appendix.

B. Determination of $T_{\mathrm{ToR},j}$

Two examples are presented here to further clarify how to generate $T_{\mathrm{ToR},j}$ using Eq. (15) for reaction type j with the number of enabled events in $\boldsymbol{L}_j$ equal to $n_{\mathrm{en},j}$. In both these examples, ΔT denotes the time increment corresponding to $T_{\mathrm{ToR},j}$.

Example 1. Constant rate coefficient, k_j. The distribution for the time interval of occurrence of a specific reaction with a constant rate coefficient k_j was already derived in Eq. (6). Thus, using Eqs. (7), (9), and (15), ΔT can be calculated from

$$1 - \{1 - [1 - \exp(-k_j \Delta T)]\}^{n_{\mathrm{en},j}} = U.$$

Solving for ΔT gives

$$T_{\mathrm{ToR},j} = T_0 + \Delta T = T_0 - \frac{\ln(U')}{n_{\mathrm{en},j} k_j}, \tag{16}$$

where $U' = 1 - U$ is a uniform random number in (0,1) [29–31].

Example 2. Time-dependent $k_j(t)$. In systems with time-varying rate coefficients, the integral in Eq. (5) should be used. The variation of the rate coefficients with time will depend on the system being considered. Here we present an example of an electrochemical system in which the rate coefficients have an exponential dependence on the system time.

Useful information about details of elementary reaction steps and chemical species involved in electrochemical systems (systems involving electron transfer) can be obtained by the CV technique. The basic idea of CV is to study the current response of the system as the electrode potential is varied as a function of time. Different species and reactions usually produce specific "signatures" on such current-potential diagrams, which makes their identification possible. The rate coefficient for an electrochemical process depends on the electrode potential. Thus, in CV, the electrochemical rate coefficients are time dependent too. This dependence is usually expressed as the Butler-Volmer law

$$k_j(t) = k_j^0 \exp\left(\frac{\alpha F E(t)}{RT}\right), \tag{17}$$

where k_j is a temperature-dependent rate coefficient in Arrhenius form and F, R, and T are the Faraday's constant, the universal gas constant, and temperature, respectively. k_j^0 and α are constants. For a linear sweep CV with sweep rate ν, the electrode potential is

$$E(t) = E_0 + \nu t. \tag{18}$$

Thus, using Eq. (5) we have

$$P(\Delta T \geq \Delta t) = \exp\left[-B_j \int_{T_0}^{T_0+\Delta t} \exp(Cs)ds\right],$$

where $B_j = k_j^0 \exp(\alpha F E_0/RT)$, and $C = \alpha F \nu/RT$. Integration of the above equation combined with Eq. (7) gives the cdf for the time interval of occurrence for this reaction,

$$F(\Delta t) = 1 - \exp\left[-\frac{B_j}{C}\{\exp[C(T_0+\Delta t)] - \exp(CT_0)\}\right]. \tag{19}$$

Finally, using Eqs. (9) and (15) we get

$$T_{\mathrm{ToR},j} = T_0 + \Delta T = \frac{1}{C}\ln\left[\exp(CT_0) - \frac{C}{n_{\mathrm{en},j}B_j}\ln U'\right].$$

Note that this expression depends on the specific form of the rate coefficients, which is given here by Eq. (17). It is worth mentioning that for $\alpha\nu < 0$ (i.e., $C < 0$) using Eqs. (15) and (19) gives

$$\lim_{\Delta t \to \infty}[1 - F_{1,j}(\Delta t)] = \exp\left[\frac{n_{\mathrm{en},j}B_j}{C}\exp(CT_0)\right],$$

implying that there is a finite probability that the time increment for $T_{\mathrm{ToR},j}$ is infinitely large, or in other words, that none of the events of this reaction type will occur at all.

C. Algorithmic steps in fFRM

Algorithm 3. fFRM.

(1) *Initialize*.

(a) Scan initial surface state to set up event list $\boldsymbol{L}_j$ for each reaction type j.

(b) Draw $T_{\mathrm{ToR},j}$ according to Eq. (15) for $1 \leq j \leq r$.

(2) *Select i such that* $T_{\mathrm{ToR},i} = \min(T_{\mathrm{ToR},j})$ *for* $1 \leq j \leq r$.

(3) *Locate an event of reaction type i at a suitable position*.

(a) An event is selected randomly from the event list $\boldsymbol{L}_i$.

(b) Check if the selected event is still enabled at the selected location. If not, remove the event from $\boldsymbol{L}_i$, and restart step 3.

(4) *Update state information*.

(a) Execute the selected event.

(b) Add newly enabled events to $\boldsymbol{L}_j$ for $j \in \mathbb{S}_1$, where $\mathbb{S}_1$ denotes the set of reaction types for which at least one new event was enabled following step 4(a).

(c) Use local update to track the number of events that were disabled, and accordingly change $n_{\mathrm{en},j}$ for $j \in \mathbb{S}_2$, where $n_{\mathrm{en},j}$ is the number of enabled events in $\boldsymbol{L}_j$, and $\mathbb{S}_2$ denotes the set of reaction types for which at least one event was disabled following step 4(a).

(d) Redraw $T_{\mathrm{ToR},j}$ according to Eq. (15) with $n_{\mathrm{en},j}$, for $j \in \mathbb{S}_1 \cup \mathbb{S}_2$.

(5) *Change system time to* $T_{\mathrm{ToR},i}$.

(6) *Repeat steps* 2–5.

It is noteworthy that the way of storing and locating events for execution in fFRM is similar to that of VSSMb (steps 1 and 3), while the selection of the next microprocess in fFRM is similar to that in FRM (step 2). This way the computationally expensive task of dealing with priority queues for all events is avoided. Another characteristic of fFRM is that it is rejection-free: the use of $n_{\mathrm{en},j}$ in generating $T_{\mathrm{ToR},j}$ guarantees that a reaction type selected in step 2 will be executed by continuing step 3 until an enabled event is located. Note that all that is required is an accurate *count* of enabled events in each event list. So, as in VSSMb and FRM, there is no need to search for the disabled events in the event lists and remove them. In fact, the disabled events are still left in the event lists, and dealt with in the same way as explained in Sec. III A. Thus, keeping $n_{\mathrm{en},j}$ updated at every step requires nominal computational cost because of using a local search approach. It will be shown in Sec. VI that this also leads to an improvement in performance compared with VSSMb, since the selection step (step 2 of VSSMb and fFRM) does not have to be redone when disabled events are encountered. The same approach can be applied to modify VSSMb to make it rejection-free as well. In this modified VSSMb, which we will refer to as rVSSM, $n_{\mathrm{en},i}$ is used in place of n_i in Eq. (10), and steps 3 and 4 are replaced with the corresponding steps of fFRM, with the exception of step 4(d).

We end this section with a discussion of two other efficient DMC algorithms applicable to systems with time-dependent rate coefficients. The first one is the generalized waiting time (GWT) Monte Carlo algorithm proposed by Prados *et al.* [30]. The GWT algorithm is essentially the

same as the VSSM algorithm [22], but with an efficient, although approximate, approach for computing the time increment. In GWT, the time domain is discretized into intervals of the same or varying lengths and the total transition rate Γ_c is considered as constant inside each of these intervals. With this approximation, a simple iterative scheme is set up for computing the time increment. Thereby, the need to solve Eq. (13) is avoided. Intervals of smaller length improve the accuracy of this approach, but come at the cost of increased number of iterations for determining the time increment, meaning more CPU time per time step. Further, the efficiency of the iterative scheme diminishes for small values of Γ_c. Another efficient algorithm, very similar to fFRM in concept, has been described by Segers [31]. Segers' algorithm is a hybrid of VSSM and FRM, and its computational cost per time step is also independent of the lattice size. The main difference between Segers' algorithm and fFRM is that in the former algorithm the time increment for reaction type j is computed from the distribution $1-\exp[-\int_{T_0}^{T_0+\Delta t} n_j k_j(s)ds]$. The use of n_j, the total number of enabled and disabled events in the event list $\boldsymbol{L}_j$, instead of $n_{\mathrm{en},j}$ causes the time evolution in Segers' algorithm to be approximate, whereas the time evolution in fFRM is always exact.

V. ACCURACY OF fFRM

A model involving the electrochemical adsorption and desorption of an ion on a (100) catalyst surface was used to assess the accuracy of fFRM. The system has both time-dependent rate coefficients and adsorbate interactions. The elementary steps in the model mechanism are [32]

$$A^- + * \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} A^* + e^-. \tag{20}$$

It is assumed that the rate coefficients of the elementary reactions steps depend on the electrode potential according to the Butler-Volmer expression. Accordingly, the rate coefficients can be expressed as

$$k_1 = k_1^0 \exp\left(\frac{\alpha FE}{RT}\right)\exp\left(-\frac{n_A \beta \epsilon_{AA}}{RT}\right), \tag{21}$$

$$k_{-1} = k_{-1}^0 \exp\left(-\frac{(1-\alpha)FE}{RT}\right)\exp\left(\frac{n_A(1-\beta)\epsilon_{AA}}{RT}\right), \tag{22}$$

where * and A^* represent an empty catalyst site and an adsorbed species, respectively. ϵ_{AA} is the repulsive interaction between two A^* occupying neighboring sites, n_A is the number of neighboring sites occupied by an A^* species, α is the transfer coefficient, β is the Bronsted-Hammond factor specifying the extent to which the repulsive interaction influences the transition state, and E is the electrode potential as defined in Eq. (18). The potential is varied linearly with a sweep rate of $\nu = 100$ mV s^{-1}. Note that because the atomic arrangement of the considered surface is (100), every atom has four nearest neighbors. Thus, n_A can take integer values between 0 and 4. This leads to five different rate constants each for the adsorption as well as the desorption processes. So $r=10$ for this model system. Results of simulations from FRM and fFRM along with the FRM results of Koper *et al.* [32] are presented in Fig. 1 and show good agreement. All simulations were done on a 256×256 square lattice. There was no appreciable change in results by increasing the lattice size. It is worth mentioning that the plateau observed at coverage around $\theta_A \approx 0.5$ is due to the formation of an ordered 2×2 adsorbate overlayer, which is a direct consequence of adsorbate interactions. Mean-field approximations have been found to fail in capturing such phenomena [32].

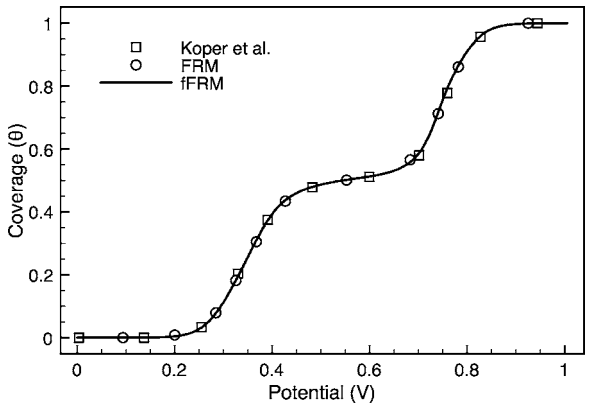

FIG. 1. Coverage-potential curve for the mechanism given by Eqs. (20)–(22). The solid line and circles are simulation results for fFRM and FRM, respectively. Squares are FRM simulation results of Koper *et al.* [32]. For clarity, only a few points are shown for the FRM simulation results of this work as well as of Ref. [32]. Parameter values used: $k_1^0 = 0.02$ s^{-1}, $k_{-1}^0 = 10^4$ s^{-1}, $T = 300$ K, $\epsilon_{AA} = 4RT$, $\nu = 100$ mV s^{-1}, $\alpha = 0.5$, $\beta = 0.5$.

VI. PERFORMANCE OF DMC ALGORITHMS

In order to facilitate comparison with VSSMb, which establishes the benchmark in performance for systems with constant rate coefficients, but which cannot be applied for simulations of systems with time-varying rate coefficients, the reaction mechanism used for the comparison of the performance of the algorithms discussed above is [24]

$$A + * \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} A^*,$$

$$B + * \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} B^*,$$

$$C + * \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} C^*,$$

$$A^* \overset{k_4}{\rightarrow} B^*,$$

$$B^* \overset{k_5}{\rightarrow} C^*.$$

Note that, although the exact values of the CPU times will change for a different chemical mechanism, the dependence

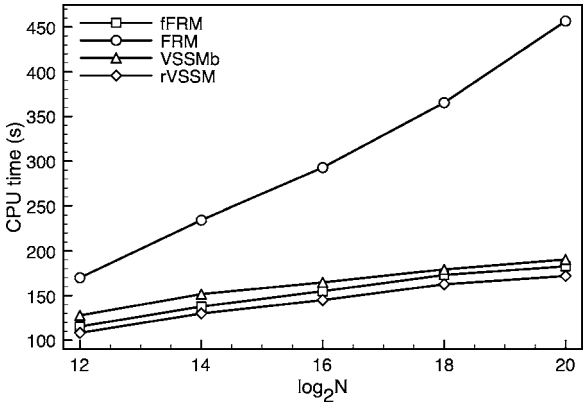

FIG. 2. CPU time needed by different algorithms for simulating 5×10^6 reaction steps at steady state. The chemical mechanism is discussed in the text. The leftmost data points correspond to a 64 $\times$64 square lattice while the rightmost data points correspond to a 1024×1024 square lattice.

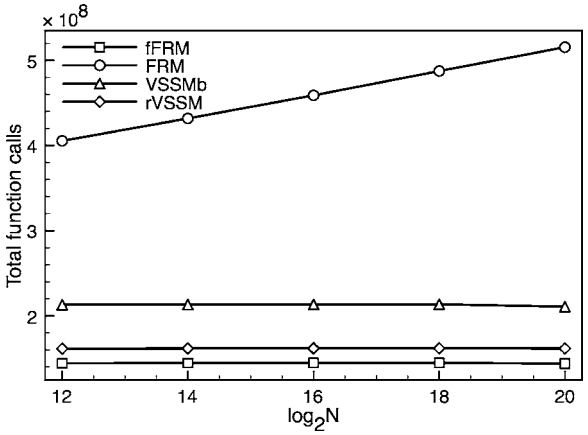

FIG. 3. Total number of function calls for simulating 5×10^6 reaction steps at steady state of the chemical mechanism discussed in Sec. VI. Different algorithms use different functions, and so a lower number of function calls does not necessarily translate into lower computational cost. Nevertheless, within the same algorithm, the number of function calls is a direct indicator of computational cost.

of the CPU times on the lattice size for the different algorithms will not change, and hence, the conclusions presented below will hold in general. Values of the rate coefficients (s^{-1}) used for the simulations are $k_1=2$, $k_{-1}=0.01$, $k_2=1.0$, $k_{-2}=0.02$, $k_3=3$, $k_{-3}=10$, $k_4=1$, $k_5=10$. The gas phase concentrations are held constant at $y_A=0.8$, $y_B=0.1$, and $y_C=0.1$. All the rate coefficients of this chemical mechanism are constant, and hence, VSSMb can also be applied. All simulations were performed on a 3.0 GHz Pentium Xeon processor with 1 MB L2 cache. Time needed for initialization and I/O was excluded from the results. A plot of the CPU time versus $\log_2 N$ is shown in Fig. 2. The solid curves show the CPU time needed for simulating 5×10^6 reactions in steady state. FRM clearly has a very marked dependence on the lattice size. As expected from the algorithm, the dependence is almost linear in $\log_2 N$. It is also interesting to note that even for a relatively small lattice size of 64×64 ($\log_2 N=12$), FRM is substantially more costly than both other algorithms. The computational cost for both VSSMb and fFRM should be expected to be independent of lattice size. However, both show a weak dependence. This dependence is attributed to the effect of the CPU cache [33]. The larger cluster sizes lead to more cache misses, and hence to a slight increase in computational time for increasing cluster sizes. The same small increase in computational cost can also be observed for FRM as a slight increase over the linear behavior in $\log_2 N$ for large N. The influence of the cache is confirmed by the plot of the total number of function calls versus $\log_2 N$ in Fig. 3. Although the number of total function calls remains almost constant for fFRM, the CPU time increases with lattice size, demonstrating that only the time per function call increases, not the number of function calls. It is worth mentioning here that a lower number of function calls for an algorithm does not necessarily mean a lower computational cost, since different algorithms make use of different functions. Nevertheless, within the same algorithm, the number of function calls is a direct indicator of computational cost. Note that the performance of fFRM is slightly better than VSSMb throughout the simulations owing to the rejection-free character of the fFRM algorithm. A similar improvement in performance is obtained for rVSSM as well. The number of function calls is significantly lower for rVSSM as compared with VSSMb, which leads to lower computational cost per simulated reaction step as can be seen in Fig. 2. Although the application of rVSSM is still limited to systems with constant rate coefficients, it can be expected that rVSSM can lead to marked performance gains over VSSMb, especially for systems that give rise to event lists with a large fraction of disabled events. On the other hand, for complex chemical systems with a large number of reaction types, the computational cost of updating $n_{\mathrm{en},j}$ may be substantial, in which case VSSMb might have a better performance than rVSSM.

VII. CONCLUSIONS

A general DMC algorithm, fFRM, was developed for efficient simulations of surface chemistry. In fact, to the best of our knowledge, fFRM is the first DMC algorithm for which the performance is independent of the lattice size even for systems with time-varying rate coefficients, which makes it applicable to the study of a wide range of chemical systems on large lattice structures. This significant improvement is achieved by an algorithmic formulation that makes it possible to forego the need to store events in a priority queue data structure, a step that contributes to most of the computational cost associated with using the FRM algorithm. Further, fFRM is also rejection-free in that the execution of a reaction type, which is selected to occur next, is guaranteed. The major advantage of fFRM is that it has approximately the same CPU cost as VSSMb, but in contrast is generally applicable to systems with constant and time-varying rate coefficients. It is also shown that the rejection-free feature of the fFRM algorithm can be combined with the VSSMb algorithm, which then leads to the rejection-free rVSSM, which, in the example considered here, leads to an approximately

15% CPU cost reduction. rVSSM, like VSSMb, is applicable only to systems with constant rate coefficients.

ACKNOWLEDGMENTS

We thank the reviewers for pointing out some relevant references. This work was supported by Honda R&D Co., Ltd., Wako, Japan. A.N. was supported by NSF Grant No. DMS-0604600, and his work was done at CTR, Stanford University.

APPENDIX

The most basic order statistics problem can be posed as, "*Given a set of random variables* $X_1, X_2, \ldots, X_n$, *reorder the set so that* $Y_1 \leq Y_2 \leq \cdots \leq Y_n$." Y_i is called the ith-order statistic. Clearly, $Y_1 = \min(X_i)$. Let $F(x)$ denote the cdf of X_i, which are assumed to be iid. Realizing that if $Y_i \leq x$, then *at least* i of the n random variables are smaller or equal to x, the cdf of Y_i is given by [34]

$$\begin{aligned} F_i(x) &= P(Y_i \leq x) \\ &= \sum_{m=i}^{n} \binom{n}{m} \{F(x)\}^m \{1 - F(x)\}^{n-m} \\ &= \int_0^{F(x)} \frac{n!}{(i-1)!\,(n-i)!} s^{i-1}(1-s)^{n-i} ds. \end{aligned} \tag{A1}$$

The last equality can easily be verified using integration by parts. Specifically, for $i=1$ in Eq. (A1) we have

$$F_1(x) = 1 - \{1 - F(x)\}^n, \tag{A2}$$

in accordance with Eq. (15).

All DMC algorithms are designed so that Eq. (4), or equivalently Eq. (7), is satisfied for every event. FRM explicitly satisfies this requirement at every step by generating the times for individual events according to Eq. (7). The following theorem proves that fFRM achieves the same task.

Theorem 2. The cdf of execution times for each enabled event in fFRM is the cdf given by Eq. (7).

Proof. Let us consider the group of enabled reactions in reaction type j. If the complete order statistics for T_{ToR} of the events of this group is generated according to Eq. (A1), the probability that the T_{ToR} for a specific reaction in this group is the ith-order statistic Y_i is $1/n_{\mathrm{en},j}$. Note that the cdf $F(x)$ in Eq. (A1) is given by Eq. (7). Evidently the pdf for the T_{ToR} of this particular reaction is given by

$$f(x) = \frac{1}{n_{\mathrm{en},j}} \sum_{i=1}^{n_{\mathrm{en},j}} \frac{dF_i(x)}{dx}. \tag{A3}$$

Now the pdf of Y_i can be obtained from Eq. (A1) as

$$\begin{aligned} \frac{dF_i(x)}{dx} &= \frac{d}{dx} \int_0^{F(x)} \frac{n!}{(i-1)!\,(n-i)!} s^{i-1}(1-s)^{n-i} ds \\ &= i\binom{n}{i} F^{i-1}(1-F)^{n-i} \frac{dF(x)}{dx}. \end{aligned} \tag{A4}$$

Using the result of Lemma 3 in Eq. (A3) gives

$$f(x) = \frac{1}{n_{\mathrm{en},j}} \frac{dF(x)}{dx} n_{\mathrm{en},j} = \frac{dF(x)}{dx}. \tag{A5}$$

This shows that the cdf of the T_{ToR} for the specific event under consideration, and hence for any other enabled event of this reaction type, is $\int_0^x f(x)dx = F(x)$. ■

Lemma 3. $\sum_{i=1}^{n} i\binom{n}{i} F(x)^{i-1}[1-F(x)]^{n-i} = n$.

Proof. Using the binomial theorem we have

$$1 = \{F(x) + [1 - F(x)]\}^n = \sum_{i=0}^{n} \binom{n}{i} F(x)^i [1 - F(x)]^{n-i}. \tag{A6}$$

Differentiation of both sides of the above equation with respect to x gives

$$0 = \frac{dF}{dx} \sum_{i=0}^{n} \binom{n}{i} F^{i-1}(1-F)^{n-i-1}[i - nF],$$

which can be rewritten as

$$\frac{1}{(1-F)} \sum_{i=1}^{n} \binom{n}{i} i F^{i-1}(1-F)^{n-i} = \frac{n}{(1-F)} \sum_{i=0}^{n} \binom{n}{i} F^i (1-F)^{n-i},$$

where the variable x was dropped for convenience. Finally, using Eq. (A6) in the equation above we find that

$$\sum_{i=1}^{n} i\binom{n}{i} F^{i-1}(1-F)^{n-i} = n.$$

[1] R. Imbihl and G. Ertl, Chem. Rev. (Washington, D.C.) **95**, 697 (1995).
[2] L. Carrette, K. A. Friedrich, and U. Stimming, Fuel Cells **1**, 5 (2001).
[3] A. Damjanovic and V. Brusic, Electrochim. Acta **12**, 615 (1967).
[4] N. M. Markovic, T. J. Schmidt, V. Stamenkovic, and P. N. Ross, Fuel Cells **1**, 105 (2001).
[5] K. Kinoshita, *Electrochemical Oxygen Technology* (John Wiley & Sons, New York, 1992).
[6] R. A. Sidik and A. B. Anderson, J. Electroanal. Chem. **528**, 69 (2002).
[7] J. X. Wang, N. M. Markovic, and R. R. Adzic, J. Phys. Chem. B **108**, 4127 (2004).
[8] J. Wintterlin, R. Schuster, and G. Ertl, Phys. Rev. Lett. **77**, 123 (1996).
[9] N. M. Markovic, H. A. Gasteiger, B. N. Grgur, and P. N. Ross, J. Electroanal. Chem. **467**, 157 (1999).
[10] M. T. M. Koper, A. P. J. Jansen, R. A. van Santen, J. J. Lukkien, and P. A. J. Hilbers, J. Chem. Phys. **109**, 6051 (1998).

[11] R. Kissel-Osterrieder, F. Behrendt, and J. Warnatz, Sym. (Int.) Combust., [Proc.] **27**, 2267 (1998).
[12] D. Gillespie, J. Phys. Chem. **81**, 2340 (1977).
[13] T. Nordmeyer and F. Zaera, Chem. Phys. Lett. **183**, 195 (1991).
[14] T. Nordmeyer and F. Zaera, J. Chem. Phys. **97**, 9345 (1992).
[15] R. Danielak, A. Perera, M. Moreau, M. Frankowicz, and R. Karpal, Physica A **229**, 428 (1996).
[16] R. Kissel-Osterrieder, F. Behrendt, and J. Warnatz, Proc. Combust. Inst. **28**, 1323 (2000).
[17] V. P. Zhdanov, Phys. Rev. E **67**, 042601 (2003).
[18] D. S. Mainardi, S. R. Calvo, A. P. J. Jansen, J. J. Lukkien, and P. B. Balbuena, Chem. Phys. Lett. **382**, 553 (2003).
[19] G. Korniss, M. A. Novotny, and P. A. Rikvold, J. Comput. Phys. **153**, 488 (1999).
[20] P. Araya, W. Porod, R. Sant, and E. E. Wolf, Surf. Sci. Lett. **208**, L80 (1988).
[21] A. S. McLeod and L. F. Gladden, J. Catal. **173**, 43 (1998).
[22] A. P. J. Jansen, Comput. Phys. Commun. **86**, 1 (1995).
[23] J. J. Lukkien, J. P. L. Segers, P. A. J. Hilbers, R. J. Gelten, and A. P. J. Jansen, Phys. Rev. E **58**, 2598 (1998).
[24] J. S. Reese, S. Raimondeau, and D. G. Vlachos, J. Comput. Phys. **173**, 302 (2001).
[25] M. A. Gibson and J. Bruck, J. Phys. Chem. A **104**, 1876 (2000).
[26] K. Binder and D. W. Heermann, *Monte Carlo Simulation in Statistical Physics* (Springer, New York, 1992).
[27] G. Brown, P. A. Rikvold, S. J. Mitchell, and M. A. Novotny, in *Interfacial Electrochemistry: Theory, Experiment, and Applications*, edited by A. Wieckowski (Marcel Dekker, New York, 1999), p. 47.
[28] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications* (Springer-Verlag, New York, 2003).
[29] If U is a uniform random number in (0,1), then is $U'=1-U$.
[30] A. Prados, J. J. Brey, and B. Sanchez-Rey, J. Stat. Phys. **89**, 709 (1997).
[31] J. P. L. Segers, Ph.D. thesis, Eindhoven University of Technology, 1998 (unpublished).
[32] M. T. M. Koper, A. P. J. Jansen, and J. J. Lukkien, Electrochim. Acta **45**, 645 (1999).
[33] Y. Shim and J. G. Amar, Phys. Rev. B **71**, 115436 (2005).
[34] N. Balakrishnan and C. R. Rao, *Order Statistics: Theory & Methods* (Elsevier Science B. V., New York, 1998).