Journal of Computational Physics 231 (2012) 7795–7814

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

# Hierarchical fractional-step approximations and parallel kinetic Monte Carlo algorithms

Giorgos Arampatzis [a], Markos A. Katsoulakis [b,*], Petr Plecháč [c], Michela Taufer [d], Lifan Xu [d]

[a] Department of Applied Mathematics, University of Crete and Foundation of Research and Technology-Hellas, Greece
[b] Department of Mathematics and Statistics, University of Massachusetts at Amherst, Amherst, MA 01003, USA
[c] Department of Mathematical Sciences, University of Delaware, Newark, DE 19716, USA
[d] Department of Computer Science, University of Delaware, Newark, DE 19716, USA

ARTICLE INFO

*Article history:*
Received 24 May 2011
Received in revised form 4 April 2012
Accepted 10 July 2012
Available online 31 July 2012

*Keywords:*
Kinetic Monte Carlo method
Parallel algorithms
Markov semigroups
Operator splitting
Graphical Processing Unit (GPU)

ABSTRACT

We present a mathematical framework for constructing and analyzing parallel algorithms for lattice kinetic Monte Carlo (KMC) simulations. The resulting algorithms have the capacity to simulate a wide range of spatio-temporal scales in spatially distributed, non-equilibrium physiochemical processes with complex chemistry and transport micro-mechanisms. Rather than focusing on constructing exactly the stochastic trajectories, our approach relies on approximating the evolution of *observables*, such as density, coverage, correlations and so on. More specifically, we develop a spatial domain decomposition of the Markov operator (generator) that describes the evolution of all observables according to the kinetic Monte Carlo algorithm. This domain decomposition corresponds to a decomposition of the Markov generator into a hierarchy of operators and can be tailored to specific hierarchical parallel architectures such as multi-core processors or clusters of Graphical Processing Units (GPUs). Based on this operator decomposition, we formulate parallel *Fractional step kinetic Monte Carlo* algorithms by employing the Trotter Theorem and its randomized variants; these schemes, (a) are *partially asynchronous* on each fractional step time-window, and (b) are characterized by their *communication schedule* between processors.

The proposed mathematical framework allows us to rigorously justify the numerical and statistical consistency of the proposed algorithms, showing the convergence of our approximating schemes to the original serial KMC. The approach also provides a systematic evaluation of different processor communicating schedules. We carry out a detailed benchmarking of the parallel KMC schemes using available exact solutions, for example, in Ising-type systems and we demonstrate the capabilities of the method to simulate complex spatially distributed reactions at very large scales on GPUs. Finally, we discuss *work load balancing* between processors and propose a re-balancing scheme based on probabilistic mass transport methods.



## 1. Introduction

Kinetic Monte Carlo algorithms have proved to be an important tool for the simulation of out-of-equilibrium, spatially distributed processes. Such models arise in physiochemical applications ranging from materials science and catalysis, to complex biological processes. Typically the simulated models involve chemistry and/or transport micro-mechanisms for

* Corresponding author.
*E-mail addresses:* garab@math.uoc.gr (G. Arampatzis), markos@math.umass.edu (M.A. Katsoulakis), plechac@math.udel.edu (P. Plecháč), taufer@cis.udel.edu (M. Taufer), xulifan@udel.edu (L. Xu).

0021-9991/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2012.07.017

atoms and molecules, e.g., reactions, adsorption, desorption processes and diffusion on surfaces and through complex media, [20,3,7]. Furthermore, mathematically similar mechanisms and corresponding kinetic Monte Carlo simulations arise in agent-based, evolutionary games problems in epidemiology, ecology and traffic networks, [36].

The simulation of stochastic lattice systems using kinetic Monte Carlo (KMC) methods relies on the direct numerical simulation of the underlying continuous time Markov Chain (CTMC). Since such stochastic processes are set on a lattice (square, hexagonal, etc.) $\Lambda_N$ with $N$ sites, they have a discrete, albeit high-dimensional, configuration space $\mathcal{S}$ and necessarily have to be of jump type describing transitions between different configurations $\sigma \in \mathcal{S}$. Mathematically, a CTMC is a stochastic process $S_t$ defined completely in terms of the local transition rates $c(\sigma, \sigma')$ which determine the updates (jumps) from any current state $S_t = \sigma$ to a (random) new state $\sigma'$. In the context of the spatially distributed applications we are interested here, the local transition rates will be denoted as

$$c(\sigma, \sigma') = c(x, \omega; \sigma), \tag{1}$$

which correspond to an updating micro-mechanism from a current configuration $S_t = \sigma$ of the system to a new configuration $\sigma^{x,\omega}$ by performing an update in a neighborhood of each site $x \in \Lambda_N$. Here $\omega$ is an index for all possible configurations $\mathcal{S}_x$ that correspond to an update at a neighborhood $\Omega_x$ of the site $x$; we refer to Section 2 for specific examples.

Heuristically, the probability of a transition over an infinitesimal time interval $\delta t$ is $\mathbb{P}(S_{t+\delta t} = \sigma^{x,\omega} \,|\, S_t = \sigma) = c(x, \omega; \sigma)\delta t + o(\delta t)$. More precisely, the local transition rates (1) define the total rate

$$\lambda(\sigma) = \sum_{x \in \Lambda_N} \sum_{\omega \in \mathcal{S}_x} c(x, \omega; \sigma), \tag{2}$$

which is the intensity of the exponential waiting time for a jump to be performed when the system is currently at the state $\sigma$. Once this exponential "clock" signals a jump, then the system transitions from the state $\sigma$ to a new configuration $\sigma^{x,\omega}$ with probability

$$p(\sigma, \sigma^{x,\omega}) = \frac{c(x, \omega; \sigma)}{\lambda(\sigma)}. \tag{3}$$

Thus the full stochastic evolution is completely defined. On the other hand, the evolution of the entire system at any time $t$ is described by the transition probabilities $P(\sigma, t; \zeta) := \mathbb{P}(S_t = \sigma \,|\, S_0 = \zeta)$,where $\zeta \in \mathcal{S}$ is any initial configuration. The transition probabilities corresponding to the local rates (1) satisfy the *master equation*, [11],

$$\partial_t P(\sigma, t; \zeta) := \sum_{\sigma', \sigma' \neq \sigma} c(\sigma', \sigma) P(\sigma', t; \zeta) - \lambda(\sigma) P(\sigma, t; \zeta), \tag{4}$$

where $P(\sigma, 0; \zeta) = \delta(\sigma - \zeta)$ and $\delta(\sigma - \zeta) = 1$ if $\sigma = \zeta$ and zero otherwise. The implementation of the KMC method is based on efficient calculation of (2) and (3), and was first developed in [6], known as a BKL algorithm, for stochastic lattice Ising models, and in [12] known as Stochastic Simulation Algorithm (SSA) for reaction systems. However, as it is evident from formulas (2) and (3), the algorithms are inherently serial as updates are done at one site $x \in \Lambda_N$ at a time, while on the other hand the calculation of (2) depends on information from the entire spatial domain $\Lambda_N$. For these reasons it seems, at first glance, that KMC algorithms cannot be parallelized easily.

However, Lubachevsky, in [23], proposed an *asynchronous* approach for parallel KMC simulation in the context of Ising systems, in the sense that different processors simulate independently parts of the physical domain, while inconsistencies at the boundaries are corrected with a series of suitable rollbacks. This method relies on uniformization of the total rates over each processor, see also [14] for the use of uniformization in the parallel simulation of general CTMC. Thus the approach yields a *null-event* algorithm, [20], which includes rejected moves over the entire domain of each processor. Furthermore, Lubachevsky proposed a modification in order to incorporate the BKL algorithm in his parallelization method, which was implemented and tested in [18]. This is a partially rejection-free (still asynchronous) algorithm, where BKL-type rejection-free simulations are carried out in the interior of each processor, while uniform rates were used at the boundary, reducing rejections over just the boundary set. However, in spite of the proposed improvements, these asynchronous algorithms may still have a high number of rejections for boundary events and rollbacks, which considerably reduce the parallel efficiency, [34]. Advancing processors in time in a synchronous manner over a fixed time-window can provide a way to mitigate the excessive number of boundary inconsistencies between processors and ensuing rejections and rollbacks in earlier methods. Such *synchronous* parallel KMC algorithms were proposed and extensively studied in [9,34,26,29]. However, several costly global communications are required at each cycle between all processors, whenever a boundary event occurs in any one of them, in order to avoid errors in the inter-processor communication and rollbacks, [29].

As we will discuss further in this paper, many of the challenges in parallel KMC can be addressed by abandoning the earlier perspective on creating a parallel KMC algorithm with the exactly same rates (and hence the generator and master equation) as the serial algorithm, see [25] for a discussion on exact algorithms. This is a very natural idea in the numerical analysis of continuum models such as Ordinary and Partial Differential Equations (ODE/PDE). First, in [35] the authors propose an *approximate* algorithm, in order to create a parallelization scheme for KMC. It was recently demonstrated [29,4], that this method is very promising: boundary inconsistencies are resolved in a straightforward fashion, while there is an absence

of global communications in contrast to synchronous relaxation schemes discussed earlier. Finally, we note that, among the parallel algorithms tested in [29], the approximate algorithm had the highest parallel efficiency.

Here we develop a general mathematical framework for *parallelizable approximations* of the KMC algorithm. Our approach, rather than focusing on constructing exactly stochastic trajectories in (2) and (3), relies on approximating the evolution of *observables* $f = f(\sigma)$. Observables of extended, spatially distributed systems such as coverage, surface roughness, correlations, etc. are defined as continuous bounded functions defined on the configuration space, that is $f \in C_b(\mathcal{S})$, see for instance (46). Typically in KMC we need to compute expected values of such observables, that is quantities such as

$$u(\zeta, t) := \mathbb{E}_\zeta[f(S_t)] = \sum_\sigma f(\sigma) P(\sigma, t; \zeta), \tag{5}$$

conditioned on the initial data $\zeta$. By a straightforward calculation using (4) we obtain that the observable (5) satisfies the initial value problem

$$\partial_t u(\zeta, t) = \mathcal{L} u(\zeta, t), \quad u(\zeta, 0) = f(\zeta), \tag{6}$$

where the operator $\mathcal{L} : C_b(\mathcal{S}) \to C_b(\mathcal{S})$ is known as the *generator* of the CTMC, [21] and in the case of (1) it is:

$$\mathcal{L} f(\sigma) = \sum_{\sigma'} c(\sigma, \sigma')[f(\sigma') - f(\sigma)] = \sum_{x \in \Lambda_N} \sum_{\omega \in \mathcal{S}_x} c(x, \omega; \sigma)[f(\sigma^{x,\omega}) - f(\sigma)]. \tag{7}$$

We then can write (5), as the action of the Markov semi-group $e^{t\mathcal{L}}$ associated with the generator $\mathcal{L}$ and the process $\{S_t\}_{t \geqslant 0}$, [21], on the observable $f$:

$$u(\zeta, t) = \mathbb{E}_\zeta[f(S_t)] = e^{t\mathcal{L}} f(\zeta). \tag{8}$$

Next, we develop a spatial domain decomposition of the initial value problem (6) and the corresponding generator $\mathcal{L}$: the lattice $\Lambda_N$ is partitioned into subsets $C_m$ such that the diameter $\operatorname{diam} C_m > L$, where $L$ is the range of particle interactions as they appear in the local rates (1), see for example (41); we can group the sets $\{C_m\}_{m=1}^M$ in such a way that there is no interaction between sites in the sets $C_m$ that belong to the same group, for instance the lattice is divided into two *sub-lattices* described by the index sets $\mathcal{I}^B$ and $\mathcal{I}^W$, (black vs. white in Fig. 1(a)), hence we have

$$\Lambda_N = \Lambda_N^B \cup \Lambda_N^W := \bigcup_{m \in \mathcal{I}^B} C_m^B \cup \bigcup_{m \in \mathcal{I}^W} C_m^W. \tag{9}$$

By *restricting* the generator $\mathcal{L}$ onto each of the sub-lattices/-sets in the domain decomposition (9) we obtain a corresponding generator decomposition:

$$\mathcal{L} = \mathcal{L}^B + \mathcal{L}^W := \sum_{m \in \mathcal{I}^B} \mathcal{L}_m^B + \sum_{m \in \mathcal{I}^W} \mathcal{L}_m^W. \tag{10}$$

Using the domain and generator decomposition in (9) and (10), we rewrite (6):

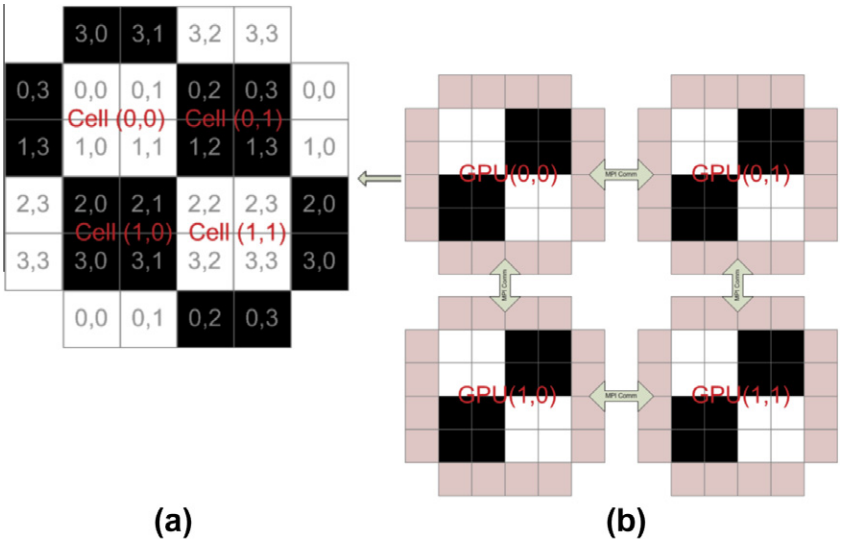



**Fig. 1.** (a) Lattice decomposition in (9) using the checkerboard scheme mapped onto a single multi-threading processing unit (e.g., GPU). The integer cell coordinates also indicate communication through boundary buffer regions. In practice other partitionings may result in a lower communication overhead. (b) Hierarchical lattice partitioning on a cluster of processing units.

$$\partial_t u(\zeta,t) = \mathcal{L}u(\zeta,t) = \mathcal{L}^B u(\zeta,t) + \mathcal{L}^W u(\zeta,t), \quad u(\zeta,0) = f(\zeta), \tag{11}$$

which allows us to approximate the solution $u$ by approximating the semigroup propagator (8) through the Trotter Theorem [37] for the approximation of semigroups corresponding to operator sums, applied to the operator $\mathcal{L} = \mathcal{L}^B + \mathcal{L}^W$:

$$u(\zeta,t) = e^{t\mathcal{L}} f(\zeta) = \lim_{n\to\infty} \left[ e^{\frac{t}{n}\mathcal{L}^B} e^{\frac{t}{n}\mathcal{L}^W} \right]^n f(\zeta). \tag{12}$$

In turn, (12) gives rise to the *Lie* splitting approximation for the propagator for a single time-step $\Delta t$:

$$e^{\Delta t\mathcal{L}} \approx e^{\Delta t\mathcal{L}^B} e^{\Delta t\mathcal{L}^W}, \quad \text{where } \Delta t = \frac{T}{n}, \tag{13}$$

while an $n$-fold iteration-see (21) provides an approximation up to any time $t \leqslant T$, as (12) shows. We note that we refer to (13) as a Lie scheme due to its analogy to similar splitting schemes for Differential Equations, [13].

On the other hand, when the simulated systems exhibit interactions of length $L$, and the subsets $C_m$ in (9) are selected such that their diameter $\operatorname{diam} C_m > L$, then we readily see that the generators $\mathcal{L}_k^B$, $\mathcal{L}_l^B$ *commute* for $k,l \in \mathcal{I}^B$, $k \neq l$:

$$\mathcal{L}_k^B \mathcal{L}_l^B - \mathcal{L}_l^B \mathcal{L}_k^B = 0, \quad \text{for all } k,l \in \mathcal{I}^B, \ \ k \neq l.$$

Hence, [37], for each of the terms in (13), we have the exact formula,

$$e^{\Delta t\mathcal{L}^B} e^{\Delta t\mathcal{L}^W} = \prod_{m\in\mathcal{I}^B} e^{\Delta t\mathcal{L}_m^B} \prod_{m\in\mathcal{I}^W} e^{\Delta t\mathcal{L}_m^W}. \tag{14}$$

Expression (14) is a key result for our method as it implies that the KMC solvers corresponding to the semigroup $e^{\Delta t\mathcal{L}^B}$ (resp. $e^{\Delta t\mathcal{L}^W}$) can be simulated *exactly* by breaking down the task into separate processors/threads for each $m \in \mathcal{I}^B$ (resp. $m \in \mathcal{I}^W$). Therefore, this scheme is *partly asynchronous* allowing us to run independently on each fractional time-step window $\Delta t$, and on every processor. The resulting computational framework consisting of the hierarchical decomposition (10). Relation (13) permits to input as the algorithm's kernel any preferred optimized serial KMC algorithm. A single time step of the parallel algorithm is thus easily described in the following steps, following (13) and (14):

*Step 1–Evolution by* $\mathcal{L}^B$: Simulate independent Markov processes $\{S_t^m\}_{t\geqslant 0}$, $m \in \mathcal{I}^B$ by a kinetic Monte Carlo kernel running on non-communicating processors that correspond to each $C_m$ for time $\Delta t$.

*Step 2–Local synchronization*: communicate configurations $\sigma^B$ from overlapping domains $\bar{C}_m^B \cap \bar{C}_n^W$ in order to update configurations $\sigma^W$.

*Step 3–Evolution by* $\mathcal{L}^W$: Simulate independent Markov processes $\{S_t^m\}_{t\geqslant 0}$, $m \in \mathcal{I}^W$ by a KMC *kernel* on non-communicating processors that correspond to each $C_m$ for time $\Delta t$.

*Step 4–Local synchronization*: communicate configurations $\sigma^W$ from overlapping domains $\bar{C}_m^B \cap \bar{C}_n^W$ in order to update configurations $\sigma^B$.

The connection of the abstract evolution problem (6) with the KMC algorithm for simulating CTMC plays a crucial role for developing a general hierarchical framework that is derived from the ideas of dimensional splitting, or equivalently of the domain decomposition applied to the underlying lattice. Focusing on approximating observables is natural in many applications in which estimating functions that do not depend on paths is a primary computational goal. On the other hand in many applications the interaction range $L$ is short range, i.e., involving only a few nearest neighbors, and thus the domain decomposition leads to an efficient parallel implementation since the communication between subdomains is confined to relatively small boundary layers in neighboring subdomains.

In the proposed *fractional step KMC* (FS-KMC) schemes, processor communication is straightforward at the end of each fractional time-step while no global communications or rollbacks are involved. In Section 5 we show that the hierarchical structure of FS-KMC can be easily implemented for very general physiochemical processes modeled by lattice systems, allowing users to input as the algorithm's KMC kernel their preferred serial algorithm. This flexibility and hierarchical structure are key advantages for tailoring our framework to particular parallel architectures with complex memory and processor hierarchies, e.g., clusters of GPUs. Furthermore, the mathematical framework of FS-KMC allows us to rigorously prove the numerical and statistical consistency of the proposed algorithms, while on the other hand it provides a systematic evaluation of different processor communication schedules. Indeed, in Section 3 the numerical and statistical consistency of the proposed algorithms is rigorously justified by the Trotter Theorem, [37], [13] showing the convergence of our approximating schemes to the original serial KMC algorithm, interpreted as convergence to the underlying Markov operator. Using the Random Trotter Theorem [19] we show that the approximation schemes with a randomized schedule, including the one in [35] as a special case, are numerically consistent in the approximation limit; that is, as the time step in the fractional step scheme converges to zero, it converges to a continuous time Markov Chain that has the same master equation and generator as the original serial KMC. In Section 4 we show that the proposed mathematical framework can allow the study of controlled-error approximation properties of fractional step KMC schemes, as well as the systematic evaluation of different processor communicating schedules, comparing for instance the scheme in [35] to the Lie scheme (21). Finally, in Section 6 we discuss work-load balancing between processors and propose a re-balancing scheme based on probabilistic mass transport methods, [10], which is particularly well-suited for the proposed fractional step KMC methods. In Section 7 we present detailed benchmarking of the proposed parallel algorithms using analytically available exact solutions, for instance, in Ising-type systems

and demonstrate the capabilities of the method to simulate complex spatially distributed molecular systems, such as CO oxidation on a catalytic surface.

## 2. Fractional step kinetic Monte Carlo algorithms

The proposed parallelization approach for KMC methods relies on approximating the evolution of relevant *observables* to spatio-temporal processes, such as density, coverage, correlations, etc. We develop a domain decomposition of the Markov operator – usually called a *generator* in the stochastic processes literature [21] – that describes the evolution of all observables according to the kinetic Monte Carlo algorithm. In this sense, our approach is not directly focusing on the exact construction of stochastic trajectories through (2) and (3) as in most of the earlier work discussed in Section 1. Before we proceed to the description of the proposed methods we review some notation and mathematical concepts from CTMC, as well as some examples of physicochemical processes which are typically simulated via KMC methods.

First, we define the configuration space of the Markov processes underlying the KMC algorithms in (2) and (3). We consider a $d$-dimensional lattice $\Lambda_N$ with $N$ lattice sites. We restrict our discussion to lattice gas models where the order parameter or the spin variable takes value in a finite countable set $\Sigma = \{0, 1, \ldots, K\}$. At each lattice site $x \in \Lambda_N$ an order parameter (a spin variable) $\sigma(x) \in \Sigma$ is defined. The states in $\Sigma$ correspond to occupation of the site $x \in \Lambda_N$ by different species. For example, if $\Sigma = \{0, 1\}$ the order parameter models the classical lattice gas with a single species occupying the site $x$ when $\sigma(x) = 1$ and with the site being vacant if $\sigma(x) = 0$. The set of the order parameters over the entire lattice, $\sigma = \{\sigma(x) : x \in \Lambda_N\}$, is called a configuration. We denote $\{S_t\}_{t \geqslant 0}$ the stochastic process of the KMC with values in the configuration space $\mathcal{S} = \Sigma^{\Lambda_N}$, i.e. a configuration $\sigma$ is the "snapshot" of the system at time $t$. Our primary focus is on modeling the basic physicochemical processes of adsorption, desorption, diffusion and reactions between different species and below we present them as particular examples.

We next turn our attention to the dynamics. First, the *local* dynamics is described by an updating mechanism and corresponding transition rates $c(x, \omega; \sigma)$ in (1), such that the configuration at time $t$, $S_t = \sigma$ changes into a new configuration $\sigma^{x,\omega}$ by an update in a neighborhood of the site $x \in \Lambda_N$. Here $\omega \in \mathcal{S}_x$, where $\mathcal{S}_x$ is the set of all possible configurations that correspond to an update at a neighborhood $\Omega_x$ of the site $x$. For example, if the modeled process is a diffusion of the classical lattice gas a particle at $x$, i.e., $\sigma(x)$ can move to any unoccupied nearest neighbor $y$ of $x$, i.e., $\Omega_x = \{y \in \Lambda_N \,|\, |x - y| = 1\}$ and $\mathcal{S}_x$ is the set of all possible configurations $\mathcal{S}_x = \Sigma^{\Omega_x}$. Practically, the sample paths $\{S_t\}_{t \geqslant 0}$ are constructed via KMC, that is through the procedure described in (2) and (3).

Observables of extended, spatially distributed systems such as coverage, surface roughness, correlations, etc. are defined as continuous bounded functions $f \in C_b(\mathcal{S})$ defined on the configuration space. Typically in KMC we need to compute expected values of such observables such as (8), where $\mathcal{L}$ is the generator (7), and the propagator of the observable $f = f(\zeta)$ in time is given by the Markov semi-group $e^{t\mathcal{L}}$ associated with the generator $\mathcal{L}$ and the process $\{S_t\}_{t \geqslant 0}$, [21].

Finally, we present a few examples relevant to the processes modeled here. We refer, for instance, to [20,3,7] for a complete discussion of the physical processes.

EXAMPLES.

1. *Adsorption/desorption for single species particles*. In this case spins take values in $\sigma(x) \in \Sigma = \{0, 1\}$, $\Omega_x = \{x\}$, $\mathcal{S}_x = \{0, 1\}$ and the update represents a spin flip at the site $x$, i.e., for $z \in \Lambda_N$
$$\sigma^{x,\omega}(z) \equiv \sigma^x(z) = \begin{cases} \sigma(z) & \text{if } z \neq x, \\ 1 - \sigma(x) & \text{if } z = x. \end{cases}$$
2. *Diffusion for single species particles*. The state space for spins is $\sigma(x) \in \Sigma = \{0, 1\}$, $\Omega_x = \{y \in \Lambda_N \,|\, |x - y| = 1\}$ includes all nearest neighbors of the site $x$ to which a particle can move. Thus the new configuration $\sigma^{x,\omega} = \sigma^{(x,y)}$ is obtained by updating the configuration $S_t = \sigma$ from the set of possible local configuration changes $\{0, 1\}^{\Omega_x}$ using the specific rule, also known as spin exchange, which involves changes at two sites $x$ and $y \in \Omega_x$
$$\sigma^{x,\omega}(z) \equiv \sigma^{(x,y)}(z) = \begin{cases} \sigma(z) & \text{if } z \neq x, y, \\ \sigma(x) & \text{if } z = y, \\ \sigma(y) & \text{if } z = x. \end{cases}$$
The transition rate is then written as $c(x, \omega; \sigma) = c(x, y; \sigma)$. The resulting process $\{S_t\}_{t \geqslant 0}$ defines dynamics with the total number of particles $(\sum_{x \in \Lambda_N} \sigma(x))$ conserved, sometimes referred to as Kawasaki dynamics.
3. *Multicomponent reactions*. Reactions that involves $K$ species of particles are easily described by enlarging the spin space to $\Sigma = \{0, 1, \ldots, K\}$. If the reactions occur only at a single site $x$, the local configuration space $\mathcal{S}_x = \Sigma$ and the update is indexed by $k \in \Sigma$ with the rule
$$\sigma^{x,\omega}(z) \equiv \sigma^{(x,k)}(z) = \begin{cases} \sigma(z) & \text{if } z \neq x, y, \\ k & \text{if } z = x. \end{cases}$$
The rates $c(x, \omega; \sigma) \equiv c(x, k; \sigma)$ define probability of a transition $\sigma(x)$ to species $k = 1, \ldots, K$ or vacating a site, i.e., $k = 0$, over $\delta t$.

4. *Reactions involving particles with internal degrees of freedom*. Typically a reaction involves particles with internal degrees of freedom, and in this case several neighboring lattice sites may be updated at the same time, corresponding to the degrees of freedom of the particles involved in the reaction. For example, in a case such as CO oxidation on a catalytic surface, [22], when only particles at a nearest-neighbor distance can react we set $\sigma(x) \in \Sigma = \{0, 1, \ldots, K\}$, $\Omega_x = \{y \in \Lambda_N \,|\, |x - y| = 1\}$ and the set of local updates $\mathcal{S}_x = \Sigma^{\Omega_x}$. Such $\mathcal{S}_x$ contains all possible reactions in a neighborhood of $x$. When reactions involve only pairs of species, the rates can be indexed by $k, l \in \Sigma$, or equivalently $\mathcal{S}_x = \Sigma \times \Sigma$. Then the reaction rate $c(x, \omega; \sigma) = c(x, y, k, l; \sigma)$ describes the probability per unit time of $\sigma(x) \to k$ at the site $x$ and $\sigma(y) \to l$ at $y$, i.e., the updating mechanism

$$\sigma^{x,\omega}(z) \equiv \sigma^{(x,y,k,l)}(z) = \begin{cases} \sigma(z) & \text{if } z \neq x, y, \\ k & \text{if } z = x, \\ l & \text{if } z = y, \end{cases}$$

where $|x - y| = 1$.

### 2.1. Domain decomposition and hierarchical structure of the generator

The generator of the Markov process $\{S_t\}_{t \geqslant 0}$ given in a general form in (7) is our starting point for the development of parallel algorithms based on geometric partitioning of the lattice. The lattice $\Lambda_N$ is decomposed into non-overlapping cells $C_m$, $m = 1, \ldots, M$ such that

$$\Lambda_N = \bigcup_{m=1}^{M} C_m, \quad C_m \cap C_n = \emptyset, \;\; m \neq n. \tag{15}$$

With each set $C_m$ a larger set $\bar{C}_m$ is associated by adding sites to $C_m$ which are connected with sites in $C_m$ by interactions or the updating mechanism, see Fig. 1(a). More precisely, we define the range of interactions $L$ for the set $C_m$ and the closure of this set

$$\bar{C}_m = \{z \in \Lambda_N \,|\, |z - x| \leqslant L, x \in C_m\}, \quad \text{where } L = \max_{x \in C_m} \{\operatorname{diam} \Omega_x\}.$$

In many models the value of $L$ is independent of $x$ due to translational invariance of the model. The boundary of $C_m$ is then defined as $\partial C_m = \bar{C}_m \setminus C_m$. This geometric partitioning induces a decomposition of the generator (7)

$$\mathcal{L}f(\sigma) = \sum_{x \in \Lambda_N} \sum_{\omega \in \mathcal{S}_x} c(x, \omega; \sigma)[f(\sigma^{x,\omega}) - f(\sigma)] \tag{16}$$

$$= \sum_{m=1}^{M} \sum_{x \in C_m} \sum_{\omega \in \mathcal{S}_x} c(x, \omega; \sigma)[f(\sigma^{x,\omega}) - f(\sigma)] \tag{17}$$

$$= \sum_{m=1}^{M} \mathcal{L}_m f(\sigma). \tag{18}$$

The generators $\mathcal{L}_m$ define new Markov processes $\{S_t^m\}_{t \geqslant 0}$ on the *entire* lattice $\Lambda_N$.

For example, in many models, [20], the interactions between particles are of the two-body type with the nearest-neighbor range and therefore the transition rates $c(x, \omega; \sigma)$ depend on the configuration $\sigma$ only through $\sigma(x)$ and $\sigma(y)$ with $|x - y| = 1$. Similarly the new configuration $\sigma^{x,\omega}$ involve changes only at the sites in this neighborhood. Thus the generator $\mathcal{L}_m$ updates the lattice sites at most in the set $\bar{C}_m = \{z \,|\, |x - z| = 1, x \in C_m\}$, see Fig. 1(a). Consequently the processes $\{S_t^m\}_{t \geqslant 0}$ and $\{S_t^{m'}\}_{t \geqslant 0}$ corresponding to $\mathcal{L}_m$ and $\mathcal{L}_{m'}$ are independent provided $\bar{C}_m \cap \bar{C}_{m'} = \emptyset$. Therefore, the decomposition (16) allows us to define independent processes which yields an algorithm suitable for parallel implementation, in particular, in the case of short-range interactions when the communication overhead can be handled efficiently.

In general, if the lattice $\Lambda_N$ is partitioned into subsets $C_m$ such that the diameter $\operatorname{diam} C_m > L$, where $L$ is the range of interactions, we can group the sets $\{C_m\}_{m=1}^{M}$ in such a way that there is no interaction between sites in the sets $C_m$ that belong to the same group. For the sake of simplicity we assume that the lattice is divided into two *sub-lattices* described by the index sets $\mathcal{I}^B$ and $\mathcal{I}^W$, (black vs. white in Fig. 1(a)), hence we have the domain decomposition (9). Other lattice partitionings are also possible and may be more suitable for specific micro-mechanisms in the KMC or the computer architecture. Returning to (9), the sub-lattices *induce* a corresponding splitting of the generator in (10). This simple observation has key consequences for simulating the process $\{S_t\}_{t \geqslant 0}$ in parallel, as well as formulating different related algorithms: the processes $\{S_t^m\}_{t \geqslant 0}$ corresponding to the generators $\mathcal{L}_m^B$ are mutually independent for different $m \in \mathcal{I}^B$, and thus can be simulated in parallel; similarly we can handle the processes belonging to the group indexed by $\mathcal{I}^W$. However, there is still communication between these two groups as there is non-empty overlap between the groups due to interactions and updates in the sets $\partial C_m$, $\partial C_{m'}$ when $m \in \mathcal{I}^B$ and $m' \in \mathcal{I}^W$ and the cells are within the interaction range $L$. To handle this communication we next introduce a fractional step approximation of the Markov semigroup $e^{t\mathcal{L}}$ associated with the process $\{S_t\}_{t \geqslant 0}$.

### 2.2. Fractional step kinetic Monte Carlo algorithms

The focus of our proposed schemes is the simulation of mean observables such as (5), which solves (6). Using the domain and generator decomposition in (9) and (10), we rewrite (6) as:

$$\partial_t u(\zeta,t) = \mathcal{L}u(\zeta,t) = \mathcal{L}^B u(\zeta,t) + \mathcal{L}^W u(\zeta,t), \quad u(\zeta,0) = f(\zeta), \tag{19}$$

which allows us to approximate the solution $u$ by approximating the semigroup propagator (8) through the Trotter Theorem [37] for the approximation of semigroups corresponding to operator sums. We note that the Trotter Theorem has found wide application in the numerical ODE/PDE analysis, e.g., [13]. Similarly, the key tool for our analysis is a deterministic as well as a stochastic version of the Trotter formula, [19], applied to the operator $\mathcal{L} = \mathcal{L}^B + \mathcal{L}^W$:

$$u(\zeta,t) = e^{t\mathcal{L}} f(\zeta) = \lim_{n\to\infty} \left[ e^{\frac{t}{n}\mathcal{L}^B} e^{\frac{t}{n}\mathcal{L}^W} \right]^n f(\zeta). \tag{20}$$

The proposed parallel scheme uses the fact that the action of the operator $\mathcal{L}^B$ (and similarly of $\mathcal{L}^W$) can be distributed onto independent processing units in (22). Thus to reach a time $T$ we define a time step $\Delta t = \frac{T}{n}$ for a fixed value of $n$ and alternate the evolution by $\mathcal{L}^B$ and $\mathcal{L}^W$. More precisely, (20) gives rise to the *Lie* splitting approximation for $n \gg 1$:

$$e^{T\mathcal{L}} \approx \left[ e^{\Delta t\mathcal{L}^B} e^{\Delta t\mathcal{L}^W} \right]^n, \quad \text{where } \Delta t = \frac{T}{n}. \tag{21}$$

The approximation can be quantified through an error expansion for each time $\Delta t$, as in (38). On the other hand, since the simulated systems exhibit short-range interactions $L$, and the subsets $C_m$ in (9) where chosen such that the diameter $\operatorname{diam} C_m > L$, the generators $\mathcal{L}_k^B$, $\mathcal{L}_l^B$ *commute* for $k,l \in \mathcal{I}^B$, $k \neq l$: $\mathcal{L}_k^B\mathcal{L}_l^B - \mathcal{L}_l^B\mathcal{L}_k^B = 0$, for all $k,l \in \mathcal{I}^B$, $k \neq l$. Hence, [37], for each of the terms in (21), we have the exact formula

$$e^{\Delta t\mathcal{L}^B} e^{\Delta t\mathcal{L}^W} = \prod_{m\in\mathcal{I}^B} e^{\Delta t\mathcal{L}_m^B} \prod_{m\in\mathcal{I}^W} e^{\Delta t\mathcal{L}_m^W}. \tag{22}$$

Then the expression (22) implies that the KMC solvers corresponding to the semigroup $e^{\Delta t\mathcal{L}^B}$ (resp. $e^{\Delta t\mathcal{L}^W}$) can be simulated *exactly* by breaking down the task into separate processors/threads for each $m \in \mathcal{I}^B$ (resp. $m \in \mathcal{I}^W$). Therefore, this scheme is *partly asynchronous* allowing us to run independently on each fractional time-step window $\Delta t$, and on every processor, a serial KMC simulation, called a *kernel*. The resulting computational framework consisting of the hierarchical decomposition (10) and (21) permits to input as the algorithm's kernel any preferred optimized serial KMC algorithm.

Finally, we emphasize that due to (21), the resulting process $\{\tilde{S}_t\}_{t\geqslant 0}$ is an *approximation* of the process $\{S_t\}_{t\geqslant 0}$ and we discuss its features and properties in the next two sections.

## 3. Processor Communication Schedule and Random Trotter Products

A key feature of the fractional step methods is the *Processor Communication Schedule* (PCS) that dictates the order with which the hierarchy of operators in (16) are applied and for how long. For instance, in (21) the processors corresponding to $\mathcal{L}^B$ (resp. $\mathcal{L}^W$) do not communicate, hence the processor communication within the algorithm occurs *only* each time we have to apply $e^{\frac{T}{2n}\mathcal{L}^B}$ or $e^{\frac{T}{2n}\mathcal{L}^W}$. Therefore we can define as the PCS the (deterministic) jump process $X = X(t)$, $t \in [0,T]$, where $[0,T]$ is the simulated time window and taking values in the set $\mathcal{X} = \{1,2\}$, where we assign the value 1 (resp. 2) to $W$ (resp. $B$):

$$X(t) = 1, \quad \frac{2kT}{n} \leqslant t < \frac{(2k+1)T}{n}, \tag{23}$$

$$X(t) = 2, \quad \frac{(2k+1)T}{n} \leqslant t < \frac{(2k+2)T}{n}. \tag{24}$$

for all $k = 0,\ldots,n-1$. Processor communication occurs at jump times, while in the remaining time the processors operate independently and do not communicate. In an analogous way we can define the PCS for the *Strang* splitting scheme (25),

$$e^{T\mathcal{L}} \approx \left[ e^{\frac{T}{2n}\mathcal{L}^B} e^{\frac{T}{n}\mathcal{L}^W} e^{\frac{T}{2n}\mathcal{L}^B} \right]^n, \tag{25}$$

with the scheduling process

$$X(t) = 1, \quad \frac{2kT}{2n} \leqslant t < \frac{(2k+1)T}{2n}, \tag{26}$$

$$X(t) = 2, \quad \frac{(2k+1)T}{2n} \leqslant t < \frac{(2k+3)T}{2n}, \tag{27}$$

$$X(t) = 1, \quad \frac{(2k+3)T}{2n} \leqslant t < \frac{(2k+4)T}{2n}, \tag{28}$$

for all $k = 0,\ldots,n-1$.

### 3.1. Random fractional step methods

In both cases above (21) and (25), the communication schedule is fully deterministic, relying on the Trotter Theorem (20). On the other hand, we can construct stochastic PCS based on the *Random Trotter Product* Theorem, and as we show below the sub-lattice algorithm proposed in [35] is a fractional step algorithm with stochastic PCS.

The Random Trotter Product Theorem, [19], extends (20) as follows: We consider a sequence of semigroups $e^{T\mathcal{L}_\xi}$ with corresponding operators $\mathcal{L}_\xi$ where $\xi$ is in the index set $\mathcal{X}$, assuming for simplicity $\mathcal{X}$ is finite, although a much more general setting is possible, (34). Consider also a stochastic jump process $X = X(t)$ with $\mathcal{X}$ as its state space. For each of its trajectories we denote by $\xi_0, \xi_1, \ldots \xi_n$ the (typically random) sequence of states visited by the stochastic process $X(t)$ and $\tau_0, \tau_1, \ldots, \tau_n$ the corresponding (also typically random) jump times

$$X(t) = \xi_0, \quad 0 \leqslant t < \tau_0, \tag{29}$$
$$X(t) = \xi_1, \quad \tau_0 \leqslant t < \tau_1, \tag{30}$$
$$\ldots \tag{31}$$
$$X(t) = \xi_k, \quad \tau_{k-1} \leqslant t < \tau_k. \tag{32}$$

We additionally define as $N(t)$ the number of jumps up to time $t$. We assume that $X(t)$ is selected so that it has an ergodic behavior, i.e., there is a probability measure $\mu(d\xi)$ such that for all bounded functions $g$ we have that

$$\lim_{t\to\infty} \frac{1}{t} \int_0^t g(X(s))\, ds = \int g(\xi)\mu(d\xi). \tag{33}$$

For example, if $X(t)$ is a Markov process then under suitable conditions, (33) will hold, where $\mu$ will be the stationary distribution of $X(t)$, [21]. Conversely, it is well-known that for a given $\mu$ we can construct in a non-unique way Markov processes $X(t)$ which satisfy the condition (33), [21]. Now we can state the Random Trotter Product Theorem, [19], in analogy to (20):

$$e^{T\bar{\mathcal{L}}} = \lim_{n\to\infty} \left[ e^{\frac{\tau_0}{n}\mathcal{L}_{\xi_0}} e^{\frac{\tau_1-\tau_0}{n}\mathcal{L}_{\xi_1}} \cdots e^{\frac{nT-\tau_{N(nt)}}{n}\mathcal{L}_{\xi_{N(nt)}}} \right], \tag{34}$$

where the operator $\bar{\mathcal{L}}$ is defined on any bounded function as

$$\bar{\mathcal{L}}g = \int \mathcal{L}_\xi \mu(d\xi). \tag{35}$$

It is clear that (21) is a special case of (34) when $\tau_k - \tau_{k-1} = 1$ and $\xi_{2k} = 1$, $\xi_{2k+1} = 2$ for all $k$. Similarly, we can also view (25) as a deterministic analogue of (34).

On the other hand, in the context of the parallel fractional step algorithms for KMC introduced here, the random process (29) can be interpreted as a stochastic PCS. For example, the sub-lattice (SL) parallelization algorithm for KMC, introduced in [35], is a fractional step algorithm with stochastic PCS: indeed, in this method the lattice is divided into sub-lattices, for instance as in (9), $\Lambda_N = \Lambda_N^B \cup \Lambda_N^W$. Each sub-lattice is selected *at random* and advanced by KMC over a fixed time window $\Delta t$. Then a new random selection is made and again the sub-lattice is advanced by $\Delta t$, and so on. The procedure is parallelizable as cells $C_m^B$, $C_m^W$ within each sub-lattice do not communicate. This algorithm is easily recast as a fractional step approximation, when in (29) we select deterministic jump times $\tau_k$ and random variables $\xi_k$:

$$\frac{\tau_k - \tau_{k-1}}{n} = \Delta t, \quad \text{and} \quad P(\xi_k = 1) = P(\xi_k = 2) = \frac{1}{2}. \tag{36}$$

As in (23), here we assign the value 1 (resp. 2) to the $W$ (resp. $B$) sub-lattice. Furthermore, we can easily calculate (35) to obtain

$$\bar{\mathcal{L}}g = \frac{1}{2}(\mathcal{L}^B + \mathcal{L}^W),$$

which is just a time rescaling of the original operator $\mathcal{L}$. Thus the SL algorithm is rewritten as the fractional step approximation with the stochastic PCS (36) as

$$e^{T\bar{\mathcal{L}}} \approx e^{\frac{\tau_0}{n}\mathcal{L}_{\xi_0}} e^{\frac{\tau_1-\tau_0}{n}\mathcal{L}_{\xi_1}} \cdots e^{\frac{nT-\tau_{N(nt)}}{n}\mathcal{L}_{\xi_{N(nt)}}}. \tag{37}$$

From the numerical analysis viewpoint, our re-interpretation of the SL algorithm in [35] as a fractional step scheme allows us to also provide a mathematically rigorous justification that it is a *consistent* estimator of the serial KMC algorithm, due to the Random Trotter Theorem (34). That is, as the time step in the fractional step scheme converges to zero, it converges to the continuous time Markov Chain that has the same master equation and generator as the original serial KMC. Finally, the (deterministic) Trotter Theorem (20) also implies that the Lie and the Strang schemes are, in the numerical analysis sense, consistent approximations of the serial KMC algorithm.

## 4. Controlled error approximations of KMC

In this section we present a formal argument for the error analysis of the fractional step approximations for KMC, which suggests the order of convergence of the schemes, as well as the restrictions on the fractional step KMC time step $\Delta t$. In the decomposition (10) the operators are linear operators on the high, but finite-dimensional configuration space $\mathcal{S}$, hence by the standard error analysis of splitting schemes, see [13], we have

$$e^{\Delta t\mathcal{L}} - e^{\Delta t\mathcal{L}^B}e^{\Delta t\mathcal{L}^W} = [\mathcal{L}^B, \mathcal{L}^W]\frac{(\Delta t)^2}{2} + \mathcal{O}(\Delta t^3), \tag{38}$$

where we readily see that the term $[\mathcal{L}^B, \mathcal{L}^W] := \mathcal{L}^B\mathcal{L}^W - \mathcal{L}^W\mathcal{L}^B$ is the Lie bracket (commutator) of the operators $\mathcal{L}^B$, $\mathcal{L}^W$. This Lie bracket captures the effect of the boundary regions $\bar{C}^B_m \cap \bar{C}^W_n$ through which we have processor communication: if there was no communication the Lie bracket would be exactly zero.

Furthermore, instead of (21) we can consider the *Strang-type* splitting (25). As in the ODE case, [13], this is expected to yield a higher order error term $\mathcal{O}(\Delta t^3)$ instead of the second order approximation in (38), in the following sense:

$$e^{\Delta t\mathcal{L}} - e^{\frac{\Delta t}{2}\mathcal{L}^B}e^{\Delta t\mathcal{L}^W}e^{\frac{\Delta t}{2}\mathcal{L}^B} = \left\{\frac{1}{12}[\mathcal{L}^W, [\mathcal{L}^W, \mathcal{L}^B]] - \frac{1}{24}[\mathcal{L}^B, [\mathcal{L}^B, \mathcal{L}^W]]\right\}(\Delta t)^3 + \mathcal{O}(\Delta t^4). \tag{39}$$

Such calculations suggest that the Strang splitting leads to a more accurate scheme, which is balanced by more complicated boundary local communication in the same time window $\Delta t$, as is evident when comparing (21) and (25).

Next, we briefly comment on the error estimation suggested by the calculation (38) and return to the rigorous numerical analysis in [1]. In order to obtain an estimate in the right-hand side of (38) which is independent of the system size $N$, it is essential to obtain an upper bound on the total number of jumps up to the time $T$. This is a key point related to the *extensivity* of the system and to the fact that the weak error analysis is restricted (as it should be physically) to mesoscopic observables satisfying (54). We observe the dependence of the error on mesoscopic observables in the following subsection. In the context of coarse-graining, in [16] an analogous estimate was shown rigorously using a Bernstein-type argument applied to the discrete derivatives, in the spirit of (54), of the solutions to the backward Kolmogorov equation. We refer to such bounds as "Bernstein-like" due to their similarity to gradient estimates for linear and nonlinear parabolic PDEs.

### 4.1. Error analysis and comparison between random and deterministic PCS

In this section we further demonstrate the use of the operator splitting formulation as a numerical analysis tool by comparing the time-step of $\Delta t$ the random PCS introduced in [35] to the deterministic Lie PCS introduced in (21). A similar comparison can be made for the Strang scheme (25). A detailed discussion including rigorous error estimates for mesoscopic observables such as (54), which are independent of the lattice size $N$ will be discussed in [1].

Here we focus on the example of adsorption/desorption discussed in Section 2. The generator in the one space dimension is decomposed as in (10)

$$\mathcal{L}^B f(\sigma) = \sum_{x\in\Lambda} c^B(x,\sigma)(f(\sigma^x) - f(\sigma)),$$

and

$$\mathcal{L}^W f(\sigma) = \sum_{x\in\Lambda} c^W(x,\sigma)(f(\sigma^x) - f(\sigma)),$$

where

$$c^B(x,\sigma) = \begin{cases} c(x,\sigma), & x \in \Lambda^B_N \\ 0, & \text{otherwise} \end{cases} \qquad c^W(x,\sigma) = \begin{cases} c(x,\sigma), & x \in \Lambda^W_N \\ 0, & \text{otherwise} \end{cases}$$

and the sub-lattices $\Lambda^B_N$, $\Lambda^W_N$ are defined in (9). The rates $c(x,\sigma)$ of the corresponding generator (7) for the case of Arrhenius adsorption/desorption are given by

$$c(x,\sigma) = c_a(1-\sigma(x)) + c_d\sigma(x)\exp(-\beta U(x,\sigma)), \tag{40}$$

where $c_a$ and $c_d$ are the adsorption and desorption constants respectively, [7]. The desorption potential $U = U(x,\sigma)$ is defined as

$$U(x,\sigma) = \sum_{y\neq x} J(x-y)\sigma(y), \tag{41}$$

where $J = J(x-y)$ is the lateral interaction potential; for simplicity we assume that the range of interactions is $L$, while in typical simplified nearest neighbor models $L = 1$. Similarly we define diffusion dynamics with Arrhenius dynamics, [15].

First we discuss the error analysis for the Lie splitting scheme. For given finite lattice size $N$, in the decomposition (10) the operators are linear operators on the high, but finite-dimensional configuration space $\mathcal{S}$, hence by the standard error analysis of Lie splitting schemes, we obtain (38). A more careful study of the commutator reveals that the generator decomposition (10) induces significant cancellations in the evaluation of the generator: indeed, we define

$$C_m^o = C_m \setminus \partial C_m, \quad C_m = C_m^o \cup C_m^\partial,$$

where in Section 2 we introduced $\partial C_m = \bar{C}_m \setminus C_m$ and $\bar{C}_m = \{z \in \Lambda_N \,|\, |z-x| \leqslant L, x \in C_m\}$. Thus, in (10) we obtain the further decomposition

$$\mathcal{L}^B = \mathcal{L}^{B,o} + \mathcal{L}^{B,\partial} := \sum_{m \in \mathcal{I}^B} \mathcal{L}_m^{B,o} + \mathcal{L}_m^{B,\partial}, \tag{42}$$

where $\mathcal{L}_m^{B,o}$, $\mathcal{L}_m^{B,\partial}$ is the restriction of $\mathcal{L}^B$ on $C_m^o$ and $C_m^\partial$ respectively. Analogously we define $\mathcal{L}^W = \mathcal{L}^{W,o} + \mathcal{L}^{W,\partial}$. We now return to the evaluation of the commutator

$$[\mathcal{L}^B, \mathcal{L}^W] = [\mathcal{L}^{B,\partial}, \mathcal{L}^{W,\partial}] + [\mathcal{L}^{B,o}, \mathcal{L}^{W,o}] + [\mathcal{L}^{B,\partial}, \mathcal{L}^{W,o}] + [\mathcal{L}^{B,o}, \mathcal{L}^{W,\partial}]. \tag{43}$$

However, due to the lack of communication between generators beyond the interaction range, we have that

$$[\mathcal{L}^{B,o}, \mathcal{L}^{W,o}] = 0, \quad [\mathcal{L}^{B,\partial}, \mathcal{L}^{W,o}] = 0, \quad [\mathcal{L}^{B,o}, \mathcal{L}^{W,\partial}] = 0,$$

thus we readily get

$$[\mathcal{L}^B, \mathcal{L}^W] = [\mathcal{L}^{B,\partial}, \mathcal{L}^{W,\partial}] = \sum_{m \in \mathcal{I}^B} \sum_{\substack{l \in \mathcal{I}^W \\ |l-m|=1}} [\mathcal{L}_m^{B,\partial}, \mathcal{L}_l^{W,\partial}]. \tag{44}$$

The formula (44) captures the processor communication between boundary regions of $\bar{C}_m^B$, $\bar{C}_n^W$. But more importantly, when combined with (38), it suggests the limitations on the time window $\Delta t$ of the Lie scheme (21), denoted for differentiation by $\Delta t_{\mathrm{Lie}}$, in order to obtain a given error tolerance TOL. In that sense it is useful to obtain an upper bound on (44). Indeed, we readily obtain:

$$\begin{aligned}[\mathcal{L}^B, \mathcal{L}^W] f(\sigma) = &\sum_{\substack{m \in \mathcal{I}^B, l \in \mathcal{I}^W \\ |l-m|=1}} \sum_{x,y} [c^B(x,\sigma) c^W(y,\sigma^x) - c^B(x,\sigma^y) c^W(y,\sigma)] f((\sigma^x)^y) - \sum_{x,y} c^B(x,\sigma)[c^W(y,\sigma^x) - c^W(y,\sigma)] f(\sigma^x) \\ &- \sum_{x,y} c^W(y,\sigma)[c^B(x,\sigma) - c^B(x,\sigma^y)] f(\sigma^y),\end{aligned} \tag{45}$$

where all summations are over $x \in C_m^{B,\partial}$, $y \in C_l^{W,\partial}$. For *mesoscopic observables*, such as the mean coverage

$$f(\sigma) = \frac{1}{N} \sum_{x \in \Lambda} \sigma(x), \tag{46}$$

we obtain

$$[\mathcal{L}^B, \mathcal{L}^W] f(\sigma) = \sum_{\substack{m \in \mathcal{I}^B, l \in \mathcal{I}^W \\ |l-m|=1}} \sum_{x,y} c^W(y,\sigma)[c^B(x,\sigma) - c^B(x,\sigma^y)] \frac{1-2\sigma(x)}{N} + \sum_{x,y} c^B(x,\sigma)[c^W(y,\sigma^x) - c^W(y,\sigma)] \frac{1-2\sigma(y)}{N}, \tag{47}$$

where all summations are over $x \in C_m^{B,\partial}$, $y \in C_l^{W,\partial}$. Therefore, due to the *cancellation* of all interior components $\mathcal{L}^{B,o}$, $\mathcal{L}^{W,o}$ in (44), we obtain the bound for the case of the interaction range $L = 1$,

$$|[\mathcal{L}^B, \mathcal{L}^W] f(\sigma)| \sim O\left(\frac{M \cdot L}{N}\right) = O\left(\frac{1}{q}\right), \tag{48}$$

where $q$ is the size of each cell $C_m$, and $O(1)$ depends on the physical parameters in the rate (40). The local error analysis in (38) and (48) can be propagated up to a prescribed time $T = N_{\mathrm{Lie}} \Delta t_{\mathrm{Lie}}$ Therefore, for the simulation of the mesoscopic observable $f$ up to the time $T$ within a given error tolerance TOL, (38) and (48)give the *observable*-dependent relation for the Lie time step

$$\mathrm{TOL} \sim T \cdot |[\mathcal{L}^B, \mathcal{L}^W] f(\sigma)| \Delta t_{\mathrm{Lie}} \sim T \cdot O\left(\frac{1}{q}\right) \Delta t_{\mathrm{Lie}}. \tag{49}$$

Next, using the fractional step formulation, we analyze in the same spirit as for the Lie scheme, the random PCS (36) proposed in [35]. For notational simplicity we set $A_1 = \mathcal{L}^W$, $A_2 = \mathcal{L}^B$. Then the local error operator $E^{\Delta t}$ can also be calculated as in (38):

$$\text{Local Error} = E^{\Delta t} := e^{\Delta t A_{\xi_1}} e^{\Delta t A_{\xi_2}} - e^{\Delta t (A_1 + A_2)}$$
$$= \left( I + (A_{\xi_1} + A_{\xi_2})\Delta t + \frac{1}{2}(A_{\xi_1}^2 + 2A_{\xi_1}A_{\xi_2} + A_{\xi_2}^2)\Delta t^2 \right) - \left( I + (A_1 + A_2)\Delta t + \frac{1}{2}(A_1 + A_2)^2 \Delta t^2 \right) + O(\Delta t^3). \quad (50)$$

The mean value of the error over the sequence of independent random variables $\xi = (\xi_i, i = 1, \ldots, n)$ of the PCS (36) on an observable $f = f(\sigma)$, $s \in \mathcal{S}$ can be explicitly evaluated:

$$\mathbb{E}_\xi[E^{\Delta t} f] = \frac{1}{4}(A_1 - A_2)^2 f \Delta t^2 + O(\Delta t^3) = \frac{1}{4}(\mathcal{L}^B - \mathcal{L}^W)^2 f \Delta t^2 + O(\Delta t^3).$$

As in (48), for the mesoscopic observable $f(\sigma) = \frac{1}{N}\sum_{x\in\Lambda}\sigma(x)$, we obtain, after disregarding the higher order local error $O(\Delta t^3)$,

$$(\mathcal{L}^B - \mathcal{L}^W)^2 f(\sigma) \sim O(1), \quad (51)$$

where $O(1)$ depends on the physical parameters in the rate (40). Similarly to (49), for the simulation of the mesoscopic observable $f$ up to the same prescribed time $T = N_{\text{Random}}\Delta t_{\text{Random}}$, within the same error tolerance TOL, (38) and (51) give the *observable*-dependent relation for the random PCS time step

$$\text{TOL} \sim T \cdot |(\mathcal{L}^B - \mathcal{L}^W)^2 f(\sigma)| \Delta t_{\text{Random}} \sim T \cdot O(1)\Delta t_{\text{Random}}. \quad (52)$$

Comparing the random and the Lie PCS through (49) and (52) implies that in order the two schemes to conform (in the mean) to the same tolerance TOL, their respective time steps should be selected so that

$$\Delta t_{\text{Lie}} \sim O(q)\Delta t_{\text{Random}}. \quad (53)$$

This relation in turn suggests that the Lie scheme (21) is expected to parallelize better than the random PCS (36) since it allows a $q$-times larger time step $\Delta t$ for the same accuracy, keeping in mind that during each time step processors do not communicate.

A similar analysis is possible for general mesoscopic observables $f = f(\sigma)$, $s \in \mathcal{S}$, e.g., spatial correlations, that satisfy

$$\sum_{x\in\Lambda_N} |f(\sigma^x) - f(\sigma)| \leqslant C, \quad (54)$$

where $C$ is a constant independent of $N$, see the formulation and estimates for coarse-grained stochastic systems in [16]. We revisit this issue, as well as the rigorous derivation of $N$-independent error bounds in place of the expansions (38) and (39)in the upcoming publication [1]. Such estimates can also allow a detailed analysis on the balance between accuracy and local processor communication for PCS such as (21), (25) and (36).

## 5. Hierarchical structure of fractional step algorithms and implementation on GPUs

The fractional step framework allows a hierarchical structure to be easily formulated and implemented, which is a key advantage for simulating in parallel architectures with complex memory hierarchies and processing units. The Graphical Processing Unit (GPU) architecture is inherently different from a traditional CPU architecture. GPUs are massively parallel multi-threaded devices capable of executing a large number of active threads concurrently. A GPU consists of multiple streaming multiprocessors (MP), each of which contains multiple scalar processor cores. For example, NVIDIA's C2050 GPU architecture contains 14 such multiprocessors, each of which contains 32 cores, for a total of 448 cores which can handle up to 24 k active threads in parallel. A GPU has several types of memory which are differently organized compared to the traditional hierarchical CPU memory, most notably the main device memory (global memory) shared between all the multiprocessors and the on-chip memory shared between all cores of a single multiprocessor (shared memory). The memory sizes and access speeds depend on the type of GPU. For instance, the memory size of the NVIDIA C2050 GPU is 3 GB while the memory size of the NVIDIA C2070 GPU is 6 GB.

From the perspective of a GPU programmer writing a code for NVIDIA GPU's, the GPU is treated as a co-processor to the main CPU. Programs are written in C and linked to the CUDA libraries [33]. A function that executes on the GPU, called a GPU kernel, consists of multiple threads executing code in a single instruction, multiple data (SIMD) fashion. That is, each thread in a GPU kernel executes the same code, but on different data. Further, threads can be grouped into thread blocks. This abstraction takes advantage of the fact that threads executing on the same multiprocessor can share data via on-chip shared memory, allowing some degree of cooperation between threads in the same block [33]. A major drawback in GPU programming is the slow communication between GPU global memory and the main memory of the CPU, compared to the communication within a GPU. Programmers address this problem by maximizing the amount of arithmetic intensive computations performed on GPU, minimizing the communication between CPU and GPU, and allowing the communication latency to be hidden by overlapping with execution. Communication among GPUs, although costly, is enabled by APIs such as OpenMP and features available in CUDA 2.2+ such as portable pinned memory, when the communication is among GPUs connected to the same shared-memory computer node. When the communication takes place among GPUs across nodes of a cluster, message passing paradigms such as MPI can serve the same scope.

In our parallelization of the KMC method, we redefine the data structures to represent lattice sites in the simulation so that the whole simulated system is cut into equal-sized black and white coarse cells like a chessboard in (9). For instance, Fig. 1(a) shows a simple example in which we map a $4 \times 4$ lattice sites into $2 \times 2$ cells, each cell containing $2 \times 2$ sites. One GPU thread is assigned to one cell. Coverage information of the whole lattice is stored in an array located in the GPU global memory so that all the threads can access the information related to their neighboring sites across MPs. The GPU kernel performing the KMC simulation over the whole lattice by using the Lie scheme (21) and the decomposition (10), is sequentially launched twice for each synchronization time step $\Delta t$ to work on the black and white cells respectively. The execution times for lattices of different sizes are compared in Fig. 2, where we take as a reference a sequential KMC-kernel, which is a direct numerical implementation of (2) and (3). The same kernel is then used for the implementation on GPUs where we compare times for different choices of $\Delta t$. We remark that the KMC kernel is not optimized by techniques such as the BKL algorithm, [6,20], which is also manifested in the scaling with respect to the size of the lattice $N$. However, the same kernel is used in the fractional step algorithm thus here we present comparisons between the same KMC algorithms, one serial and one parallelized by the fractional step approach. Clearly any optimized KMC kernel can be used without difficulty in our framework.

The size of lattices that can be simulated on a single GPU is limited by memory, thus in order to simulate large systems it will be necessary to employ a cluster of GPUs communicating, for instance, through an MPI protocol. We will demonstrate next how fractional step KMC algorithms can be tailored to an architecture that involves multiple GPUs. We return to the formulation in (10), and consider the sub-lattice decomposition (9). In this formulation each one of the coarse-cells $C_m^B$ or $C_m^W$ are simulated on a single GPU. Within each one of the GPUs we have the same lattice decomposition as in (9), see Fig. 1(b), namely

$$C_m^B = C_m^{BB} \cup C_m^{BW} := \bigcup_{l=1}^{L} D_{ml}^{BB} \cup \bigcup_{l=1}^{L} D_{ml}^{BW}, \tag{55}$$

and similarly we define a decomposition for $C_m^W$. Each one of the (sub-)lattices $D_{ml}^{BB}$ and $D_{ml}^{BW}$ corresponds to individual threads within the GPU. Next, (9) and (55) define *nested sub-lattices*, which yield a hierarchical decomposition of the operator $\mathcal{L}$ into (10) and

$$\mathcal{L}_m^B = \mathcal{L}_m^{BB} + \mathcal{L}_m^{BW} := \sum_{l=1}^{L} \mathcal{L}_{ml}^{BB} + \sum_{l=1}^{L} \mathcal{L}_{ml}^{BW}, \tag{56}$$

and similarly we also define the decomposition for $\mathcal{L}_m^W$. Finally, schemes such as (21) and (25) give rise to fractional step algorithms based on the nested decompositions (10) and (56). In this case, boundary communication, see Fig. 1(b), plays a key

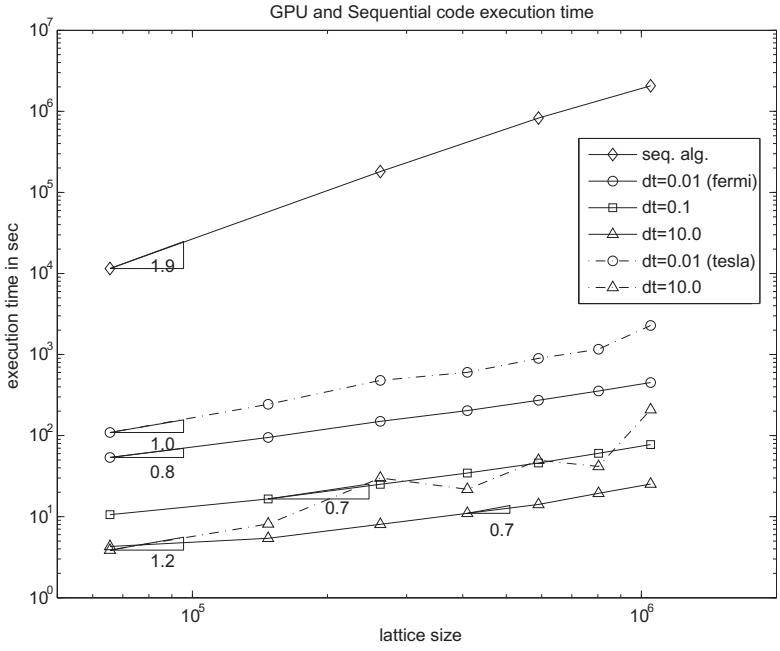



**Fig. 2.** Execution time of the fractional step KMC for lattices of different sizes. The comparison with the sequential algorithm (top curve) is based on the same SSA KMC implementation which, however, does not have the optimal complexity of the BKL algorithm. The simpler implementation of the SSA algorithm was used. The simple implementation has the complexity $\mathcal{O}(N^2)$, where $N$ is the total number of lattice sites. This complexity is reflected in the indicated scaling (the slope in the log–log plot). Note that the due to partitioning of the lattice in the fractional step algorithm the same KMC kernel will scale as $\mathcal{O}(N)$ only, which is in agreement with the observed slope in the plots.

role in the parallelization of our algorithm when multiple GPUs are required. As we discussed earlier, this scenario happens when the lattice size grows to the point that the lattice data structures no longer fit into a single GPU global memory. In turn, this threshold depends on the type of GPU used, e.g., for a NVIDIA's C2050 GPU the maximum lattice size is currently $8{,}182 \times 8{,}182$ cells. To simulate larger systems, we can decompose the domain into regular sub-domains and distribute both the sub-domain cells and associated computation among multiple GPUs, as discussed in (56). Boundary communication between two adjacent sub-domains are exchanged between GPUs, see Fig. 1(b), and supported by either MPI or OpenMP, depending on the fact that the GPUs are located on the same cluster node or across nodes. Thus, the multi-GPU parallel KMC algorithm is based on and benefits from the hierarchical structure of the fractional step KMC algorithms discussed in (56). At the same time, it can enable the scalability of our simulations to lattice sizes beyond the ones accessible with a single GPU e.g., $8{,}182 \times 8{,}182$ sites in a C2050 GPU. The study of performance and scalability of our multi-GPU algorithm and code for different lattice sizes and types of GPU clusters is beyond the scope of this paper.

## 6. Mass transport and dynamic workload balancing

Due to the spatially distributed nature of KMC simulations and the dependence of jump rates on local coverage, (2), fractional step algorithms may have an imbalance in the number of operations/jumps performed in each coarse cell $C_m$ in (9), as well as on the corresponding processors. In fact, formulas (2) and (3), and the very structure of the fractional step algorithms (10), allow us to define the *workload* $W_{n\Delta t}(\sigma) = W_{n\Delta t}(m;\sigma)$, $1 \leqslant m \leqslant M$ as

$$W_{n\Delta t}(m) = \#\text{jumps in } C_m \text{ during } [(n-1)\Delta t, n\Delta t], \tag{57}$$

when the configuration at time $(n-1)\Delta t$ is $\sigma$. We also renormalize $W_{n\Delta t}$ (and still denote it with the same symbol) in order to obtain a histogram, i.e., a probability density. Since different coarse cells $C_m$ in the fractional step algorithms such as (21) or (25) do not communicate during intervals of length $\Delta t$ the quantities (57) are easy to keep track on-the-fly during the simulations. The possibility of workload imbalance is depicted in Fig. 3, where many more jumps are performed in the processors corresponding to cells of low coverage, while the other processors remain idle.

In this section we introduce a probabilistic strategy to re-balance the workload $W_{n\Delta t}$ dynamically during the simulation based on the following idea from *mass transport* methods, e.g., [10]. One wants to transport the "imbalanced" density $W_{n\Delta t}$ into an almost uniform density over the number of processors used, in order to ensure that they remain as uniformly active as possible. The mass transport connection and terminology refers to the mapping of a given probability measure into a desirable probability measure. Typically, [10], this problem is posed as an optimization over a suitable cost functional and is known as the Monge–Kantorovich problem. In our context the cost functional could reflect constraints related to various parallel architectures.

We can formulate and implement this strategy in several different ways: probably the simplest approach, that serves mostly as an illustration, is to assume that we have a number of processors $P$, where $P \ll M$; during the interval $[(n-1)\Delta t, n\Delta t]$ a number of coarse cells $C_m$, $1 \leqslant m \leqslant M$, which are simulated independently in a fractional step algorithm, are allocated to each processor. By the end of the simulation time $n\Delta t$ the workload on all processors is described similarly to (57), by a histogram $R_{n\Delta t}(\sigma) = R_{n\Delta t}(l;\sigma)$, $1 \leqslant l \leqslant P$. One wants to map (57) onto a histogram $R_{n\Delta t}$ which is almost uniform in $1 \leqslant l \leqslant P$. One such function can be constructed by mapping the mass corresponding to each value of the cumulative distribution function (cdf) of (57), onto an equal mass on the uniform distribution over the $P$ processors. In another implementation of the mass transport method we can adjust the size of the coarse cells $C_m$ according to the workload

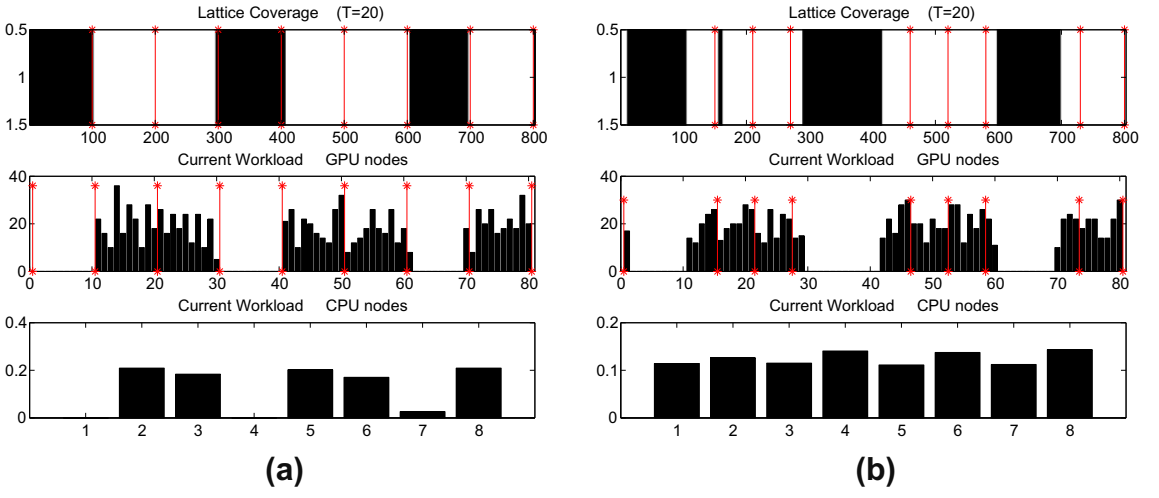



**Fig. 3.** (a) Workload imbalance in 1D unimolecular reaction system: the top figure depicts local coverage, the bottom figure workload distribution. (b) Workload redistribution in figure (a) using the mass transport for re-balancing.

redistribution strategy discussed earlier, see Fig. 3. This is effectively a one-dimensional example of an adsorption/desorption process where the mass transport procedure is carried out by mapping (57) into a new histogram $R_{n\Delta t}(\sigma) = R_{n\Delta t}(l;\sigma)$ corresponding to a new set of variable size coarse cells $C_l$, $1 \leqslant l \leqslant M'$. The cell size adjustment ensures the uniformity of the new histogram by defining $R_{n\Delta t}$ as a mapping of the cdf corresponding to (57).

The mass transport mappings discussed above are not expected to be carried out at every time step $n\Delta t$ in order to reduce computational and communication cost, but instead they should follow a rationally designed coarser-in-time schedule, in analogy to processor communication scheduling, e.g., (29).The overall implementation appears rather simple since here we demonstrated the methodology in a one-dimensional example. However, in higher dimensions, adjusting the size and shape of coarse cells $C_m$ can be much harder. Nevertheless the structure of re-balancing procedure can remain one-dimensional even in higher dimensional lattices if we pick a sub-lattice decomposition (15) into strips $C_m$. We note that the mapping we constructed using cdf's did not take into account the processor architecture and a suitable cost functional formulation for the mass transport to a uniform distribution, as in the Monge–Kantorovich problem, [10], may be more appropriate. We will revisit such issues in a future publication.

## 7. Parallel simulations: benchmarks and applications

Exactly solvable models of statistical mechanics provide a test bed for sampling algorithms applied to interacting particle systems. We present benchmarks for two important cases: (a) sampling of equilibrium distributions, i.e., long time behavior of the simulated Markov process, and (b) weak approximations of the dynamics. In the first set of tests we work with the classical Ising model on one and two dimensional lattices where spins interact through a nearest-neighbor potential. Thus the Hamiltonian of the system is

$$H(\sigma) = -\frac{K}{2}\sum_{x\in\Lambda_N}\sum_{|y-x|=1}\sigma(x)\sigma(y) + h\sum_{x\in\Lambda_N}\sigma(x),$$

where $K$ is a real parameter that defines the strength of the interaction and $h$ the external field. We work with the spin-flip Arrhenius dynamics with the rates defined in the nearest-neighbor set $\Omega_x = \{z \,|\, |z-x| = 1\}$ and the updates in $\mathcal{S}_x = \{0,1\}$.

$$c(x,\sigma) = c_1(1-\sigma(x)) + c_2\sigma(x)e^{-\beta U(x)}, \tag{58}$$

$$U(x) = K\sum_{y\in\Omega_x}\sigma(x+y) + h, \tag{59}$$

with $\beta$ is a given inverse temperature. The generator of (58) is a self-adjoint operator on the space $L^2(\mathcal{S},\mu_N)$ where $\mu_N(d\sigma) = Z^{-1}e^{-\beta H(\sigma)}\,d\sigma$ is the canonical Gibbs measure of the system at the constant inverse temperature $\beta$. Consequently the dynamics is reversible and the measure $\mu_t$ of the process $\{S_t\}_{t\geqslant 0}$ converges to the Gibbs measure $\mu_N$ as $t\to\infty$. Thus the dynamics (58) can be used for computing expected values $\mathbb{E}_{\mu_N}[f]$ by invoking ergodicity and averaging on a single trajectory

$$\mathbb{E}_{\mu_N}[f] \equiv \int_{\mathcal{S}} f(\sigma)\mu_N(d\sigma) = \lim_{T\to\infty}\frac{1}{T}\int_0^T f(S_t)\,dt.$$

In the simulations we estimate two observables:

$$\text{mean coverage}: \ \bar{c}_t = \frac{1}{|\Lambda_N|}\mathbb{E}\left[\sum_{x\in\Lambda_N}\sigma_t(x)\right],$$

$$\text{2-point correlation function}: \ \bar{\lambda}_t(x,y) = \mathbb{E}[\sigma_t(x)\sigma_t(x+y)].$$

Due to translational invariance the function $\bar{\lambda}_k(x,y)$ depends on the distance $|x-y|$ only. For exactly solvable one and two dimensional Ising models we have explicit formulas which we summarize here for the spins in $\Sigma = \{0,1\}$.

*1D Ising model:* The one-dimensional Ising model does not exhibit a phase transition and thus presents a simple benchmark for accuracy. Working with lattice gas models requires a simple transformation of the well-known exact solution, [5], which for the Hamiltonian of the system given on the periodic lattice

$$H(\sigma) = -K\sum_{x=1}^{N}\sigma(x)\sigma(x+1) + h\sum_{x=1}^{N}\sigma(x),$$

yields the equilibrium mean coverage and the 2-point correlation function

$$\bar{c}(h,\beta) = \frac{1}{2}\left(1 + \frac{\sinh(h')}{(\sinh^2(h') + e^{-4K'})^{1/2}}\right), \tag{60}$$

$$\bar{\lambda}(x,y) = \frac{1}{4}(1+e^{4K'}\sinh^2(h'))\left[\frac{e^{K'}\cosh(h') - e^{-K'}(1+e^{4K'}\sinh^2(h'))^{1/2}}{e^{K'}\cosh(h') + e^{-K'}(1+e^{4K'}\sinh^2(h'))^{1/2}}\right]^{(x-y)}, \quad y \geqslant x, \tag{61}$$

where

$$K' = \frac{1}{4}\beta K, \quad \text{and} \quad h' = \frac{1}{2}\beta(h - K). \tag{62}$$

Since the one-dimensional Ising model does not exhibit a phase transition it allows us to assess the accuracy of the approximation for the phase diagram calculation. The phase diagram depicting dependence of the coverage on the external field for different values of $\beta$ is shown in Fig. 4(a). In this simulation a rather conservative $\Delta t = 1.0$ was chosen. The statistical errors (confidence intervals) are below the resolution of the graph. As seen in the figure the isotherms for the average equilibrium coverage are thus obtained with a good accuracy. As a global observable the total coverage is less sensitive to statistical errors therefore we also monitor the 2-point correlation function and its agreement with the exact solution (61). The results for different values of $\beta$ in Fig. 4(b) demonstrate good accuracy.

*2D Ising model:* The phase transition that occurs in two-dimensional Ising model presents a more challenging test case. However, the celebrated exact solution due to Onsager for spins $\Sigma = \{-1, 1\}$, [30], in the case with the zero external field and further refinements yield closed formulas for the mean coverage and two point correlation functions. We restrict our tests to the isotropic case, i.e., on the two-dimensional periodic lattice we have the Hamiltonian

$$H(\sigma) = -K \sum_{x=(x_1,x_2)\in\Lambda_N} (\sigma(x_1,x_2)\sigma(x_1,x_2+1) + \sigma(x_1,x_2)\sigma(x_1+1,x_2)) + h\sum_{x\in\Lambda_N}\sigma(x).$$

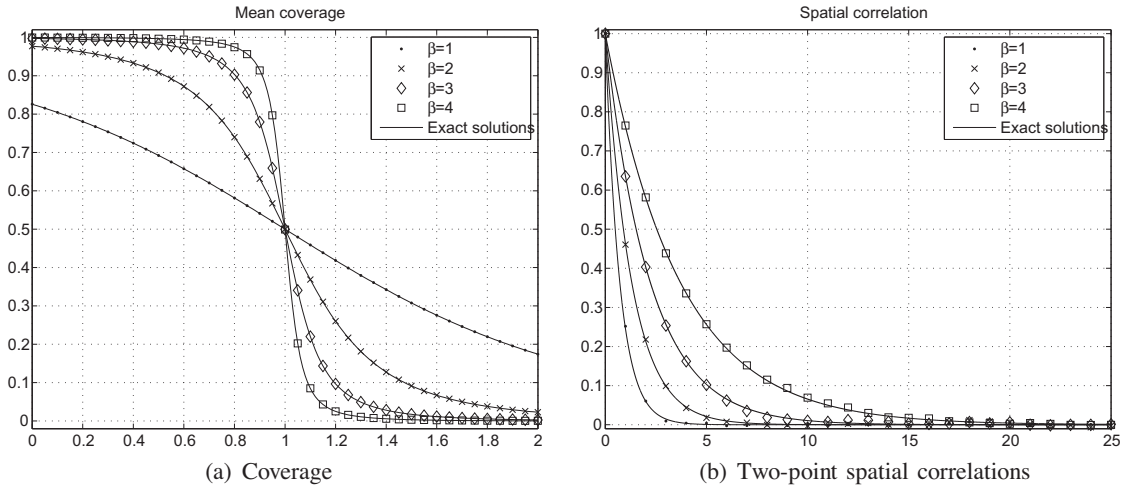
Transforming the exact solutions for the spins $\Sigma = \{0, 1\}$ we obtain the equivalent to the zero external field the value $h = 2K$ at which value the critical inverse temperature solves $\sinh(\frac{1}{2}\beta_c K) = 1$. The exact solution for the mean coverage has the form

$$\bar{c}(\beta) = \begin{cases} \frac{1}{2}(1 + [1 - (\sinh(\frac{1}{2}\beta K))^{-4}]^{1/8}), & \beta > \beta_c, \\ \frac{1}{2}, & \beta < \beta_c. \end{cases} \tag{63}$$

The exact solution for the 2-point correlation is available in [38], however, we use only the asymptotics in $|x - y|$, [5]. Introducing $\kappa = (\sinh(\frac{1}{2}\beta K))^{-2}$ we have

$$\bar{\lambda}(x,y) = \begin{cases} (1-\kappa^2)^{1/4} + \mathcal{O}(\kappa^{|x-y|}), & \beta > \beta_c, \\ \mathcal{O}(\kappa^{-|x-y|/2}), & \beta < \beta_c. \end{cases} \tag{64}$$

The phase diagram is computed at $h = 2$ which for $K = 1$ corresponds to the regime when the second-order phase transition occurs at the critical temperature $\sinh(\frac{1}{2}K\beta_c) = 1$. Sampling the coverage exhibits well-known difficulties close to the critical point $\beta_c$ which are not cured by the fractional step algorithm. Instead, we demonstrate in Fig. 5(a) that for wide range of choices $\Delta t$ the phase diagram is constructed accurately for $\beta$ outside a neighborhood of $\beta_c$. Close to the critical point the algorithm provides approximations that are in agreement with other Monte Carlo sampling approach. The finite-size effects are pronounced at the neighborhood of the critical point due to algebraic decay of correlations. Thus it is not expected that a good agreement with the infinite volume exact solution will be observed in the finite size simulations. Nonetheless, the presence of the second-order phase transition is indicated in the computed phase diagram. Furthermore, the proposed algorithm provides an efficient implementation that allows for simulations on large lattice. It is shown in Fig. 5(b) that algebraic decay




(a) Coverage (b) Two-point spatial correlations

**Fig. 4.** (a) Comparison of the exact solution (60) (solid line) for the total coverage $c_\beta(K, h)$, $K = 1$, with the mean coverage obtained in simulations on the one-dimensional lattice with $N = 2^{15}$ and $\Delta t = 1.0$. (b) Two-point spatial correlation function estimated at $h = 1$ on the same lattice and $\Delta t = 1.0$ compared to the exact solution.

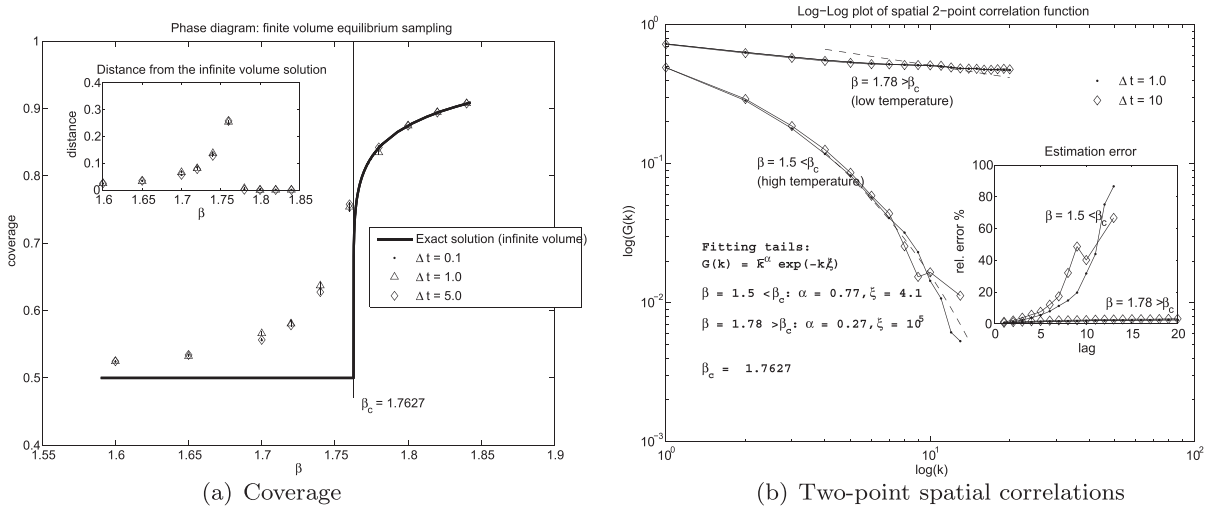



**Fig. 5.** (a) Comparison of the exact solution (63) (solid line) for the total coverage $c_\beta(K,h)$, $h=2$, with mean coverage obtained in simulations on the one-dimensional lattice with $N=128$ and various $\Delta t$'s. (b) Spatial two-point correlation function in the two-dimensional Ising model simulated on the lattice $N=512^2$ at a sub-critical temperature $\beta>\beta_c$ and supercritical regime $\beta<\beta_c$. The simulation confirms the behavior obtained from the infinite volume exact solution: at high temperatures the decay is exponential while at temperatures below the critical temperature the decay is algebraic. The dashed line represents the fitted function of the form $k^{-\alpha}e-k/\xi$.

of the 2-point correlation function is well approximated in the low-temperature (sub-critical) regime, while at super-critical temperatures the exponential decay is observed. Overall, we note that such long-time sampling of the simulated CTMC is a particularly challenging task since in principle, errors from any approximation may accumulate at long times and contaminate the simulation.

Studying approximation properties of the stochastic dynamics poses a more difficult task due to the lack of an exact solution for the evolution of observables. Certain guidance can be obtained from mean-field approximations, however, those do not give sufficiently good approximation for Ising model in low dimensions. Therefore we compare the evolution of the coverage obtained from the traditional SSA algorithm with approximations generated by the proposed fractional time step algorithm with different choices $\Delta t$. In Fig. 7(a) we compare the expected value and variance of the total coverage process $C_t=\frac{1}{|\Lambda_N|}\sum_{\Lambda_N}S_t(x)$. Furthermore, it is also shown that the auto-correlation function for the process $C_t$ is well-approximated and approximations converge as $\Delta t\to 0$, see Figs. 6(b) and 7(b).

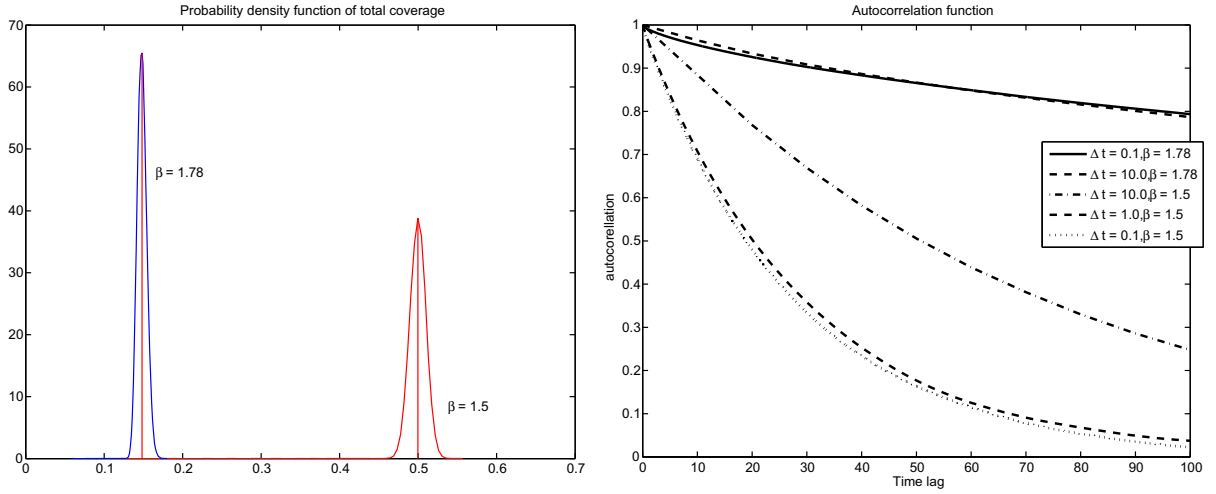
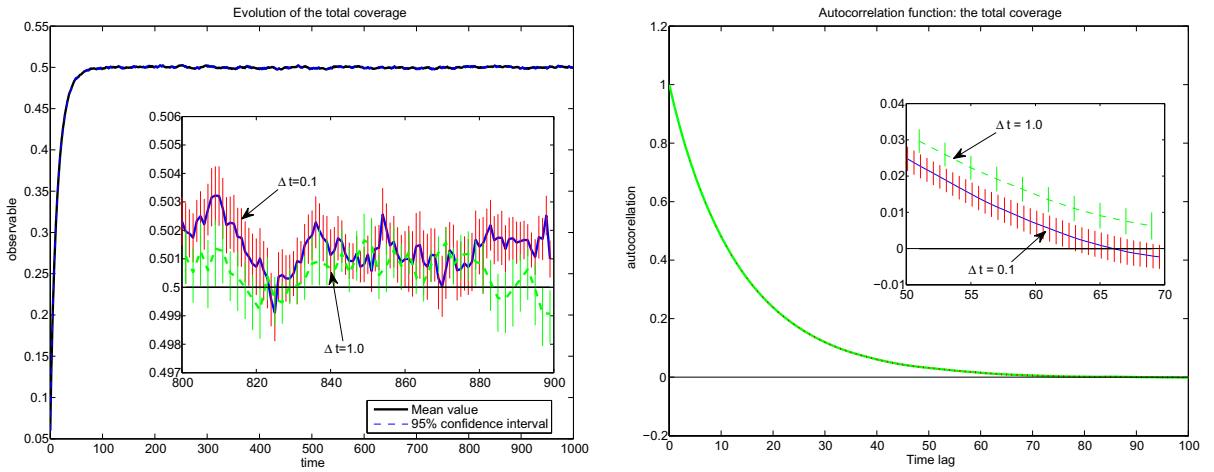



**Fig. 6.** (a) Estimated equilibrium distributions of the coverage process at the two temperatures simulated in Fig. 5(b). (b) Autocorrelation functions for the coverage process in the two-dimensional Ising model simulated at $\beta=1.5$ (high temperature above the critical temperature $\beta_c$ and at $\beta=1.78>\beta_c$ (low temperature), see parameters in Fig. 5(b).




(a) Sample path (b) Autocorrelation function

**Fig. 7.** (a) A sample path of the total coverage process $\{S_t\}$ simulated at $\Delta t = 1.0$ and $\Delta t = 0.1$ on the one-dimensional lattice with $N = 2^{15}$ and $\Delta t = 1.0$. (b) Autocorrelation function of the coverage process. The means were obtained from $M = 1000$ independent realizations of the process at $\beta = 4$ and $h = 1$. The inset shows error bars for the empirical mean estimator.

### 7.1. Examples from catalysis and reaction engineering

In order to demonstrate the applicability of the proposed parallelization methodology in systems exhibiting complex spatio-temporal morphologies at mesoscopic length scales, e.g., islands, spirals, rings, etc., we implement a KMC algorithm arising in the modeling of chemical reaction dynamics on a catalytic surface. Here we focus on CO oxidation, which is a prototypical example for molecular-level reaction–diffusion mechanisms between adsorbates on a surface. We note that molecular dynamics simulations have also been employed to understand micro-mechanisms on surfaces such as reaction paths [31]. However, reaction kinetics for mesoscale adsorbate structures cannot be simulated by using molecular dynamics because of spatio-temporal scale limitations of such methods, while KMC methods, have the ability to simulate much larger scales [24].

In KMC models for CO oxidation on a catalytic surface spatial resolution is a critical ingredient of the modeling since inhomogeneously adsorbed O and CO react on the catalytic surface only where the corresponding phases meet. Sophisticated KMC models for CO oxidation on catalytic surfaces, where kinetic parameters are estimated by *ab initio* density functional theory (DFT), [17], were recently developed in [32] and later in [28,22]. Such KMC models yield a remarkable agreement with experiments, see also the review articles [27,8].

Next we demonstrate the performance of parallel fractional step algorithms for KMC simulation to heterogeneous catalysis. We implement a simplified CO oxidation model known as the Ziff–Gulari–Barshad (ZGB) model, [40], which was one of the first attempts towards a spatially distributed KMC modeling in reaction systems. Although a simplified model compared to the *ab initio* KMC models described earlier, it incorporates the basic mechanisms for the dynamics of adsorbate structures during CO oxidation on catalytic surfaces: single site updates (adsorption/desorption) and multi-site updates (specifically, reactions with two sites being involved). The spins take values $\sigma(x) = 0$ denoting a vacant site $x \in \Lambda_N$, $\sigma(x) = -1$ for a molecule CO at $x$, and $\sigma(x) = 1$ representing a $O_2$ molecule. Depending on the local configurations of the nearest neighbors in $\Sigma_x = \{y \,|\, |y - x| = 1\}$ the events in Table 1 are executed. The rates of individual events depend on the states in $\Omega_x$ which are enumerated by $\omega = \{1, 2, 3, 4\}$ and are summarized in Table 1.

**Table 1**
An Event in $\Omega_x$, $x^{nn} \in \Omega_x$ is a randomly selected site from the nearest-neighbor set of $x$, and $r_2(x) = \frac{1}{4}(1 - \sigma(x)^2)v_0^x$, $r_3(x) = \frac{1}{8}\sigma(x)(1 + \sigma(x))v_{-1}^x$, $r_4(x) = \frac{1}{8}\sigma(x)(\sigma(x) - 1)v_1^x$, where $v_k^x$ is the number of nearest neighbors ($nn$) of $x$ that are equal to $k$.

| $\omega$ | Site | $\sigma(x)$ | $\sigma^x$ | Rate $c(x, \omega; \sigma)$ | Comment |
|---|---|---|---|---|---|
| 1 | Vacant | 0 | $0 \to 1$ | $k_1(1 - (\sigma(x))^2)$ | CO adsorb |
| 2 | Vacant | 0 | $0 \to -1$<br>$0 \to -1$, $x^{nn}$ | $(1 - k_1)r_2(x)$ | $O_2$ adsorb |
| 3 | CO | 1 | $1 \to 0$<br>$-1 \to 0$, $x^{nn}$ | $k_2 r_3(x)$ | CO + O and desorb |
| 4 | O | $-1$ | $-1 \to 0$<br>$1 \to 0$, $x^{nn}$ | $k_2 r_4(x)$ | CO + O and desorb |

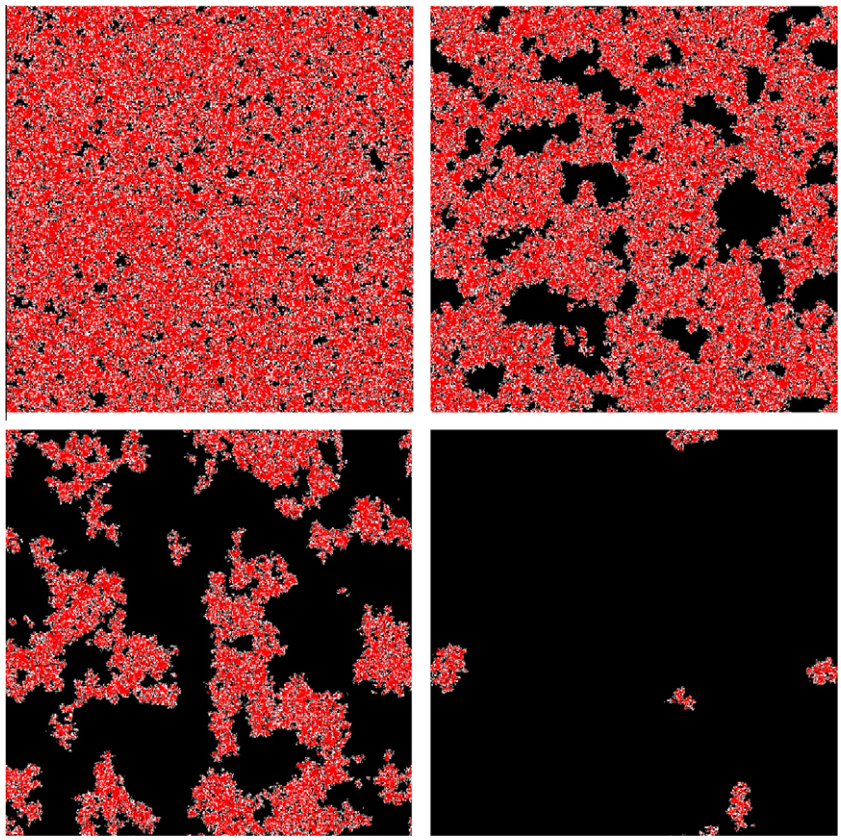


**Fig. 8.** Snapshot at different simulation times for the CO oxidation process, on a two-dimensional lattice $N = 1024^2$.

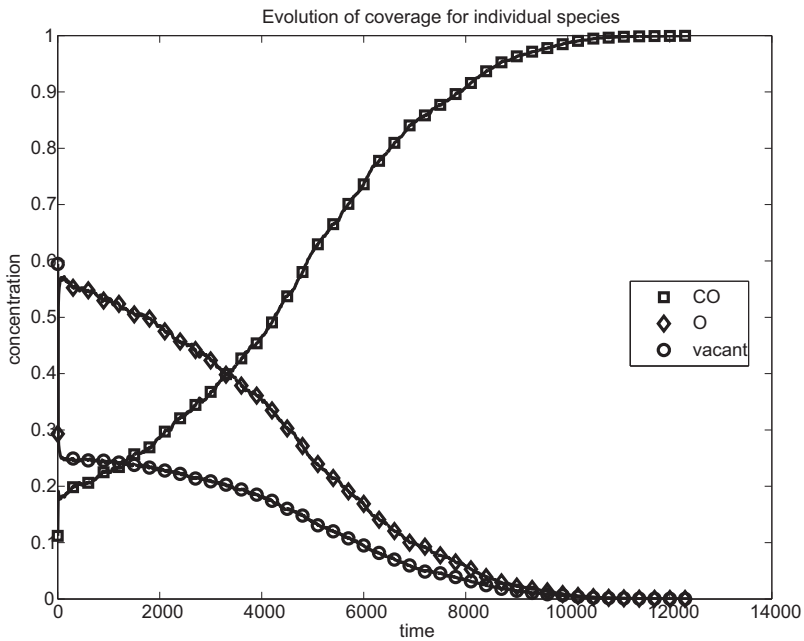



**Fig. 9.** Evolution of the mean coverage (46) for species in the oxidation process (CO, $O_2$, and vacant sites).

The execution times for lattices of different sizes are compared in Fig. 2, while a snapshot of the spatial morphology is depicted in Fig. 8. In Fig. 9 the evolution of the mean coverage of the species is presented as a function of time. Here we take as a reference the sequential KMC-BKL kernel. The same kernel is then used for the implementation on GPUs where we compare times for different choices of $\Delta t$. We remark that the KMC kernel is not optimized by techniques such as the BKL algorithm, [20], which is manifested in the scaling with respect to the size of the lattice $N$. However, the *same kernel* is used in the fractional step algorithm thus we present fair comparisons between serial and parallel solvers, noting that any optimized serial KMC algorithm can be used as a kernel in our fractional step framework. It is worth noting that by partitioning of the problem into the subproblems the $\mathcal{O}(N^2)$ complexity of the simple implementation for the SSA algorithm is reduced, which is also demonstrated in Fig. 2 where the slope of lines for simulations using GPUs suggest the reduced complexity of order $\mathcal{O}(N)$. Hence the proposed approach also offers a simple but efficient implementation of KMC simulators.

Finally, in our implementation (as well as in the original ZGB model) we did not implement the fast diffusion mechanism of O adsorbates on the surface, [22]. However, the scheme (25) can allow us to easily implement within our parallelization framework schemes with disparate time-scales which turn out to be important for the long-time adsorbate dynamics.

## 8. Conclusions

In this paper we proposed a new framework for constructing parallel algorithms for lattice KMC simulations. Our approach relies on a spatial decomposition of the Markov generator underlying the KMC algorithm, into a hierarchy of operators corresponding to processors' structure in the parallel architecture. Based on this operator decomposition, we can formulate fractional step approximation schemes by employing the Trotter product formula; these schemes allow us to run independently on each processor a serial KMC simulation on each fractional time-step window. Furthermore, the schemes incorporate the communication schedule between processors through the sequential application of the operators in the decomposition, as well as the time step employed in the particular fractional step scheme. Here we discussed deterministic schedules resulting from Lie- and Strang-type fractional step schemes, as well as random schedules derived by the Random Trotter Theorem, [19]. We demonstrated that the latter category includes the algorithm [35] as one particular example.

Some of the key features of the proposed framework and possible future directions include: The hierarchical structure can be easily derived and implemented for very general physiochemical processes modeled by lattice systems, allowing users to input as the KMC kernel their preferred serial algorithm. This flexibility and hierarchical structure allow for tailoring our framework to particular parallel architectures with complex memory and processor hierarchies, e.g., clusters of GPUs communicating, for instance, through an MPI protocol, and using the nested generator decomposition (56). Moreover, multiscale Trotter algorithms for systems with fast and slow processes are widely used in molecular dynamics, e.g., [13], and they can be recast along with the proposed methods into a spatio-temporal hierarchy of operators that allow computational tasks to be hierarchically decomposed in space/time. The numerical consistency of the proposed algorithms is rigorously justified by Trotter Theorems, [37,19] showing the convergence of our approximating schemes to the original serial KMC algorithm. Related numerical estimates are expected to provide insights on the design and the relative advantages of various communication schedules and architectures. We discussed work load balancing between processors through a re-balancing scheme based on probabilistic mass transport methods that is particularly well-suited for the proposed fractional step KMC methods. We carried out detailed benchmarking using analytically available exact solutions from statistical mechanics and applied the method to simulate complex spatially distributed molecular systems, such as reaction–diffusion processes on catalytic surfaces. Finally, we studied the performance and scalability of our algorithm (56) and the resulting code for different lattice sizes and types of GPUs.

Concluding we note that there are some interesting conceptual analogies between the parallelization and coarse-graining algorithms of KMC such as the coarse-grained Monte Carlo (CGMC) method e.g., [15,2]. In both methods we decompose the particle system in components communicating minimally, e.g., (10) and (21) or trivially as in coarse-graining methods, thus, local information is represented by collective (coarse) variables, or computed on separate processors within a parallel architecture. An early work towards parallelizing CGMC [15] in problems with locally well-mixed particle interactions is [39], while further progress towards understanding and exploiting the analogies and the complementarity of CGMC and parallel KMC has the potential to give efficient KMC algorithms capable of simulating complex systems at mesoscopic length scales.

## Acknowledgement

The research of M.A.K. was partially supported by the National Science Foundation under the grant NSF-DMS-071512, by the Office of Advanced Scientific Computing Research, U.S. Department of Energy under DE-SC0002339 and the European Commission FP7-REGPOT-2009-1 Award No. 245749. The research of P.P. was partially supported by the National Science Foundation under the grant NSF-DMS-0813893 and by the Office of Advanced Scientific Computing Research, U.S. Department of Energy under DE-SC0001340; the work was partly done at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. The research of G.A. was partially supported by the National Science Foundation under the grant NSF-DMS-071512 and NSF-CMMI-0835582. The research of M.T. was partially supported

by the AFOSR STTR Program, the Army Research Office under the grant 54723-CS, and the NVIDIA University Professor Partnership Program. The research of L.X. was supported by the National Science Foundation under the grant NSF-DMS-0813893.

## References

[1] G. Arampatzis, M.A. Katsoulakis, P. Plecháč, Error analysis for parallel kinetic Monte Carlo algorithms: accuracy and processor communication, 2012, preprint.
[2] S. Are, M.A. Katsoulakis, P. Plecháč, L. Rey-Bellet, Multibody interactions in coarse-graining schemes for extended systems, SIAM J. Sci. Comput. 31 (2008) 987–1015.
[3] S.M. Auerbach, Theory and simulation of jump dynamics, diffusion and phase equilibrium in nanopores, Int. Rev. Phys. Chem. 19 (2000).
[4] S.P.C. Battaile, M. Chandross, L. Holm, A. Thompson, V. Tikare, G. Wagner, E. Webb, X. Zhou, C.G. Cardona, A. Slepoy, Crossing the mesoscale no-man's land via parallel kinetic Monte Carlo, Sandia report, 2009.
[5] R.J. Baxter, Exactly Solved Models in Statistical Mechanics, third ed., Academic Press, 1989.
[6] A.B. Bortz, M.H. Kalos, J.L. Lebowitz, A new algorithm for Monte Carlo simulation of Ising spin systems, J. Comput. Phys. 17 (1975) 10–18.
[7] A. Chatterjee, D. Vlachos, An overview of spatial microscopic and accelerated kinetic Monte Carlo methods, J. Comput. Aided Mater. Des. 14 (2007) 253–308, http://dx.doi.org/10.1007/s10820-006-9042-9.
[8] C.H. Christensen, J.K. Norskov, A molecular view of heterogeneous catalysis, J. Chem. Phys. 128 (2008).
[9] S.G. Eick, A.G. Greenberg, B.D. Lubachevsky, A. Weiss, Synchronous relaxation for parallel simulations with applications to circuit-switched networks, ACM Trans. Model. Comput. Simul. 3 (1993) 287–314.
[10] L.C. Evans, Partial differential equations and Monge–Kantorovich mass transfer, in: Current Developments in Mathematics, 1997 (Cambridge, MA), International Press, Boston, MA, 1999, pp. 65–126.
[11] C.W. Gardiner, Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences, Springer, 2004.
[12] D.T. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions, J. Comput. Phys. 22 (1976) 403–434.
[13] E. Hairer, C. Lubich, G. Wanner, Geometric Numerical Integration, Structure-Preserving Algorithms for Ordinary Differential Equations, Springer Series in Computational Mathematics, second ed., vol. 31, Springer-Verlag, Berlin, 2006.
[14] P. Heidelberger, D.M. Nicol, Conservative parallel simulation of continuous time Markov chains using uniformization, IEEE Trans. Parallel Distrib. Syst. 4 (1993) 906–921.
[15] M. Katsoulakis, A. Majda, D. Vlachos, Coarse-grained stochastic processes for microscopic lattice systems, Proc. Nat. Acad. Sci 100 (2003) 782-782–782.
[16] M.A. Katsoulakis, P. Plecháč, A. Sopasakis, Error analysis of coarse-graining for stochastic lattice dynamics, SIAM J. Numer. Anal. 44 (2006) 2270–2296.
[17] W. Kohn, Nobel lecture: electronic structure of matter-wave functions and density functionals, Rev. Mod. Phys. 71 (1999) 1253–1266.
[18] G. Korniss, M.A. Novotny, P.A. Rikvold, Parallelization of a dynamic Monte Carlo algorithm: a partially rejection-free conservative approach, J. Comput. Phys. 153 (1999) 488–508.
[19] T.G. Kurtz, A random Trotter product formula, Proc. Am. Math. Soc. 35 (1972) 147–154.
[20] D.P. Landau, K. Binder, A Guide to Monte Carlo Simulations in Statistical Physics, Cambridge University Press, Cambridge, 2000.
[21] T.M. Liggett, Interacting Particle Systems, Grundlehren der Mathematischen Wissenschaften, vol. 276, Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1985.
[22] D.J. Liu, J.W. Evans, Atomistic and multiscale modeling of CO-oxidation on Pd(1 0 0) and Rh(1 0 0): from nanoscale fluctuations to mesoscale reaction fronts, Surf. Sci. 603 (2009) 1706–1716.
[23] B.D. Lubachevsky, Efficient parallel simulations of dynamic Ising spin systems, J. Comput. Phys. 75 (1988) 103–122.
[24] J.J. Lukkien, J.P.L. Segers, P.A.J. Hilbers, R.J. Gelten, A.P.J. Jansen, Efficient Monte Carlo methods for the simulation of catalytic surface reactions, Phys. Rev. E 58 (1998) 2598–2610.
[25] E. Martínez, J. Marian, M.H. Kalos, J.M. Perlado, Synchronous parallel kinetic Monte Carlo for continuum diffusion-reaction systems, J. Comput. Phys. 227 (2008) 3804–3823.
[26] M. Merrick, K.A. Fichthorn, Synchronous relaxation algorithm for parallel kinetic Monte Carlo simulations of thin film growth, Phys. Rev. E 75 (2007) 011606.
[27] H. Metiu, Preface to special topic: a survey of some new developments in heterogeneous catalysis, J. Chem. Phys. 128 (2008).
[28] M. Nagasaka, H. Kondoh, I. Nakai, T. Ohta, CO oxidation reaction on Pt(1 1 1) studied by the dynamic Monte Carlo method including lateral interactions of adsorbates, J. Chem. Phys. 126 (2007) 044704–044707.
[29] G. Nandipati, Y. Shim, J.G. Amar, A. Karim, A. Kara, T.S. Rahman, O. Trushin, Parallel kinetic Monte Carlo simulations of Ag(1 1 1) island coarsening using a large database, J. Phys. Condens. Matter 21 (2009) 084214.
[30] L. Onsager, Crystal statistics. I. A two-dimensional model with an order-disorder transition, Phys. Rev. 65 (1944) 117–149.
[31] M.C. Payne, M.P. Teter, D.C. Allan, T.A. Arias, J.D. Joannopoulos, Iterative minimization techniques for abinitio total-energy calculations - molecular-dynamics and conjugate gradients, Rev. Mod. Phys. 64 (1992) 1045–1097.
[32] K. Reuter, D. Frenkel, M. Scheffler, The steady state of heterogeneous catalysis, studied by first-principles statistical mechanics, Phys. Rev. Lett. 93 (2004).
[33] J. Sanders, E. Kandrot, CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley Professional, Cambridge, 2010.
[34] Y. Shim, J.G. Amar, Rigorous synchronous relaxation algorithm for parallel kinetic Monte Carlo simulations of thin film growth, Phys. Rev. B 71 (2005) 115436.
[35] Y. Shim, J.G. Amar, Semirigorous synchronous relaxation algorithm for parallel kinetic Monte Carlo simulations of thin film growth, Phys. Rev. B 71 (2005) 125432.
[36] G. Szabo, G. Fath, Evolutionary games on graphs, Phys. Rep. 446 (2007) 97–216.
[37] H.F. Trotter, On the product of semi-groups of operators, Proc. Am. Math. Soc. 10 (1959) 545–551.
[38] T.T. Wu, B.M. McCoy, C.A. Tracy, E. Barouch, Spin–spin correlation functions for the two-dimensional Ising model: exact theory in the scaling region, Phys. Rev. B 13 (1976) 316–374.
[39] L. Xu, M. Taufer, S. Collins, and D. G. Vlachos: Parallelization of Tau-Leap Coarse-Grained Monte Carlo Simulations on GPUs. In Proceedings of the IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS), April 2010, Atlanta, Georgia, USA.
[40] R.M. Ziff, E. Gulari, Y. Barshad, Kinetic phase transitions in an irreversible surface-reaction model, Phys. Rev. Lett. 56 (1986) 2553.