Parallel kinetic Monte Carlo simulations: algorithms and numerical analysis

M. A. Katsoulakis[1], P. Plecháč[2], and D. G. Vlachos[3]

[1]Mathematics and Statistics, University of Massachusetts at Amherst, [2]Mathematical Sciences, University of Delaware, [3]Chemical Engineering, University of Delaware

Project: MULTISCALE MATHEMATICS FOR BIOMASS CONVERSION TO RENEWABLE HYDROGEN

DE-SC0001340

Overreaching computational thrusts

Catalytic kinetics for renewables on novel structures

- Kinetic Monte Carlo methods for complex kinetics
- Hybrid DFT/KMC methods
- Coarse-grained MC methods
- GPU/multicore acceleration of MC simulations

Accelerating kinetic Monte Carlo simulations

- Mathematical framework for **parallel algorithms** in lattice kinetic Monte Carlo (kMC) simulations, **that covers pre-existing methods**.
- **Easy to implement** on GPUs or multicore processors.
- **Numerical and statistical consistency** rigorously justified
- Rigorous **error analysis** and convergence of parallel approximation schemes..
- **Load balancing** and communication protocols
- Efficient and reliable **parallel** (pseudo)-random number generators

Mathematical tools and algorithms

Modeling setup:

- Continuous time Markov jump process on $\Lambda \subset \mathbb{Z}^d$
- Chemical species/particle configuration: $\sigma : \Lambda_N \to \mathbb{R}$
- Lattice processes: adsorption, desorption, diffusion, chemical reactions
- Multi-site updates $\sigma^{x,\omega}$ for most systems

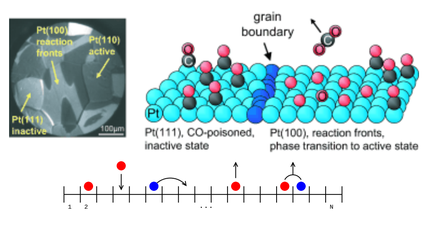


Figure 1: *Oxidation on Pt(111), Suchorski et al.(2010).*

Generator of the continuous time Markov process $\{\sigma_t\}$

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

Evolution of observables $u(\sigma,t) = \mathbb{E}[f(\sigma_t)]$, $f : \Sigma \to \mathbb{R}$:

$$\partial_t u(\sigma,t) = \mathcal{L}u(\sigma,t)$$

Simulation engine

SSA (Stochastic Simulation Algorithm) or **n-fold BKL**

- Exact simulation of $\{\sigma_t\}$
- Complexity $\mathcal{O}(N^2)$ or $\mathcal{O}(N \log N)$, for short-range interactions $\mathcal{O}(1)$.
- asynchronous parallel versions inefficient

New approach: **Partially Asynchronous kinetic Monte Carlo**[1]

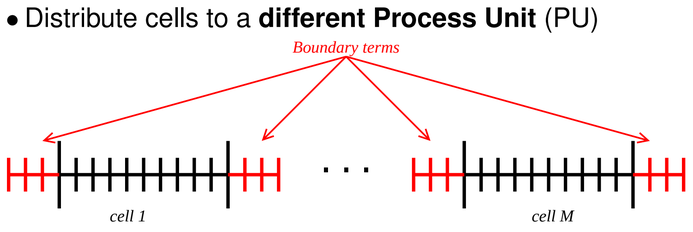
Partially asynchronous parallel algorithm

Lattice decomposition

- Lattice decomposed into M **cells** each of them containing $q = \frac{N}{M}$ sites

$$\Lambda = \bigcup_{k=1}^{M} C_k, \quad C_i \cap C_j = \emptyset,\ i \neq j$$

- Distribute cells to a **different Process Unit** (PU)



Operator decomposition

- The operator $\mathcal{L}$ is decomposed using the lattice decomposition

$$\mathcal{L}f(\sigma) = \sum_{k=1}^{M} \underbrace{\sum_{x\in C_k} c(x,\sigma)(f(\sigma^x) - f(\sigma))}_{\mathcal{L}_k} = \underbrace{\sum_{k\ \text{odd}} \mathcal{L}_k f(\sigma)}_{\mathcal{L}^{\text{odd}}} + \underbrace{\sum_{k\ \text{even}} \mathcal{L}_k f(\sigma)}_{\mathcal{L}^{\text{even}}}$$

- Backward Kolmogorov equation for $u(t,\sigma) = \mathbb{E}_\sigma f(\sigma_t)$ becomes

$$\begin{cases} \partial_t \boldsymbol{u}(t) = \mathcal{L}^{\text{odd}}\boldsymbol{u}(t) + \mathcal{L}^{\text{even}}\boldsymbol{u}(t) \\ \boldsymbol{u}(0,\sigma) = \boldsymbol{f}(\sigma) \end{cases} \tag{1}$$

Approximation of backward Kolmogorov equation

- Short range interactions: the processes $\sim \mathcal{L}_k$ are independent and can be simulated exactly on separate processors:

$$e^{\mathcal{L}\Delta t} \approx e^{\mathcal{L}^O \Delta t} e^{\mathcal{L}^E \Delta t} = \prod_{k:\,\text{odd}} \overbrace{e^{\mathcal{L}_k \Delta t}}^{\text{k-th}} \prod_{k:\,\text{even}} \overbrace{e^{\mathcal{L}_k \Delta t}}^{\text{processor}}$$

- **Lie splitting**,

$$u(\Delta t) = e^{\Delta t \mathcal{L}} u(0) \approx e^{\Delta t \mathcal{L}^{\text{even}}} e^{\Delta t \mathcal{L}^{\text{odd}}} u(0)$$

- **Strang splitting**,

$$u(\Delta t) = e^{\Delta t \mathcal{L}} u(0) \approx e^{\Delta t \mathcal{L}^{\text{even}}} e^{\frac{\Delta t}{2} \mathcal{L}^{\text{odd}}} e^{\Delta t \mathcal{L}^{\text{even}}} u(0)$$

- **Random splitting** [4],

$$u(\Delta t) = e^{\Delta t \mathcal{L}} u(0) \approx e^{\Delta t \mathcal{L}^{\xi_1}} e^{\Delta t \mathcal{L}^{\xi_2}} u(0)$$

$\mathbb{P}(\mathcal{L}^\xi = \mathcal{L}^{\text{odd}}) = p$ and $\mathbb{P}(\mathcal{L}^\xi = \mathcal{L}^{\text{even}}) = 1 - p$

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

$(\tau_k - \tau_{k-1})/n = \Delta t$, and $\mathbb{P}(\xi_k = O) = \mathbb{P}(\xi_k = E) = 1/2$

Random Trotter theorem, [3]:

$$\bar{\mathcal{L}}g = \int \mathcal{L}_\xi g \mu(d\xi)\,, \quad \lim_{t\to\infty} \frac{1}{t} \int_0^t g(X(s))\,ds = \int g(\xi)\,\mu(d\xi)$$

- Processor Communication Schedule – ξ_t

Unifying mathematical tools

- Operator spatial decomposition.
- Approximation of the Markov semigroup.
- Random Trotter Theorem, [3].
- Mathematical framework, includes previous parallel methods, e.g., [4].

Algorithm 1: *Example: Lie splitting*

1. Run on **odd cells** the SSA algorithm: execute events with random times δt up to time Δt.
2. Synchronize the boundary overlapping regions.
3. Run on **even cells** the SSA algorithm: execute events with random times δt up to time Δt.
4. Synchronize the boundary overlapping regions.
5. Repeat until time $T = n\Delta t$

Error analysis

Local Error

- The local error for the Lie scheme is bounded by

$$\|\mathcal{S}_L(\Delta t)u_0 - u(\Delta t)\| \le c_1 \|[\mathcal{L}_1, \mathcal{L}_2]u_0\| \Delta t^2 + R_L(u_0)\Delta t^3$$

and

$$R_L(u) = c_2 \sum_{|m|=3} \|\mathcal{L}_1^{m_1}\mathcal{L}_2^{m_2} u\|$$

and u solves Eq. (1)

- The local error for the Strang scheme,

$$\|\mathcal{S}_S(\Delta t)u_0 - u(\Delta t)\| \le c_3 \|[\mathcal{L}_1,[\mathcal{L}_1,\mathcal{L}_2]]u_0 - 2[\mathcal{L}_2,[\mathcal{L}_2,\mathcal{L}_1]]u_0\| \Delta t^3 + R_S(u_0)\Delta t^4$$

where

$$R_S(u) = c_4 \sum_{|m|=4} \|\mathcal{L}_1^{m_1}\mathcal{L}_2^{m_2}\mathcal{L}_1^{m_3} u\|$$

- For the Random scheme

$$\text{Mean Local Error} \sim c_5 \|(\mathcal{L}_1 - \mathcal{L}_2)^2 u_0\| \Delta t^2$$

Global Error

- The global error for the Lie scheme at $t_n = n\Delta t$:

$$\|\mathcal{S}_L(t_n)u_0 - u(t_n)\| \le C_1 \max_{k=0,\ldots,n} \|[\mathcal{L}_1,\mathcal{L}_2]u(t_k)\| \Delta t + \mathcal{O}(\Delta t^2) \tag{2}$$

where

$$\mathcal{O}(\Delta t^2) \sim R_L(u(t_i)),\ i = 0,\ldots,n$$

- When the commutator, as well as the remainder term, is independent of the system size N?

Mesoscopic Observables for Extended Systems

- Let f be a mesoscopic observable that satisfies

$$\sum_{x\in\Lambda} |f(\sigma^x) - f(\sigma)| \le C$$

(e.g., mean coverage, spatial correlations)

- Bernstein-type estimates, [2] yield the global error bound (2) for the solution $u(\sigma,t)$ of Eq. (1).
- $[\mathcal{L}_1,\mathcal{L}_2]u(t)$ and $R_L(u(t))$ are **independent** from the size N of the system.
- Error for mesoscopic observables:

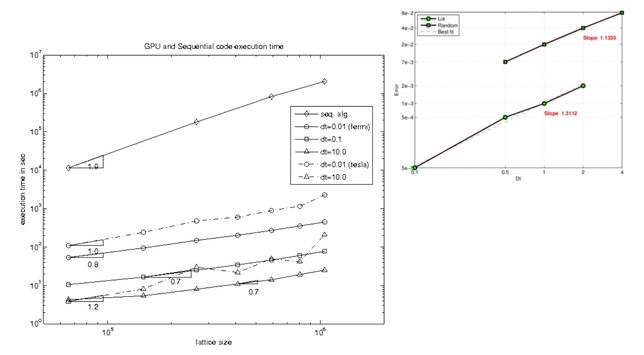
Deterministic scheme (Lie) $\|[\mathcal{L}_1,\mathcal{L}_2]f(\sigma)\| \sim \mathcal{O}\left(\frac{1}{q}\right)$

Randomized scheme $\|(\mathcal{L}_1 - \mathcal{L}_2)^2 f(\sigma)\| \sim \mathcal{O}(1)$

- For a given **error tolerance**:

$$\Delta t_L \sim \mathcal{O}(q)\Delta t_R$$

Application performance



Weak scalling: nVidia-Fermi, SSA kernel, serial complexity $\mathcal{O}(\mathbf{N}^2)$

- Processor Communication Schedule:
  Larger Δt $\Rightarrow$ larger error but **less communication**.

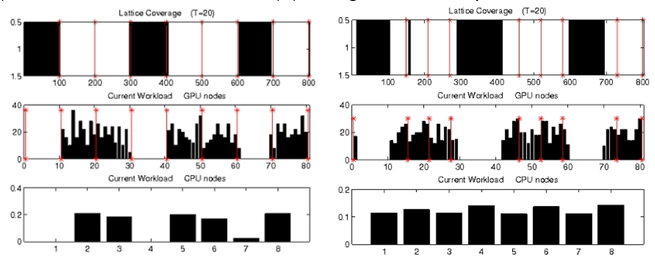
Load balancing

Treated as Monge-Kantorovich transport problem:

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

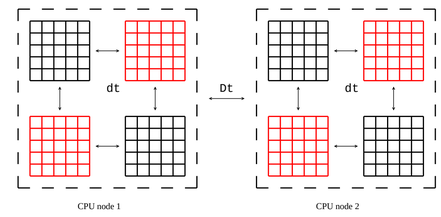
Mass transport $W_{n\Delta t}$ to a uniform distribution of jumps.

Example: (a) Workload imbalance in 1D unimolecular reaction system; (b) Workload redistribution in (a) using mass transport.



Hierarchical decomposition on multiple GPUs

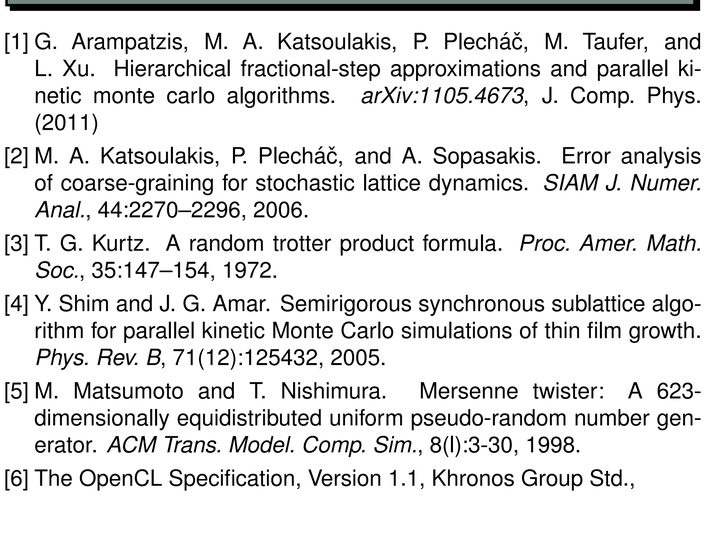
Example: 2D system $\approx 1\mu\text{m}^2$ – $N = 10^8$ – 16 GPUs



Applications in catalysis modeling

Simulation of **CO oxidation ZGB model** on a catalytic surface with the following reactions

$$\text{CO} \downarrow, \quad \text{O}_2 \downarrow, \quad \text{CO} + \text{O} \to \text{CO}_2 \uparrow$$



Implementation and software development

General purpose parallel kMC environment in OPENCL, [6], **G. Arampatzis and A. Athanasopoulos**

- efficient and portable parallel (pseudo)-random number generator
  Parallel version of MT, [5], named DCMT, ported to OpenCL
  available from `http:www.math.udel.edu/~plechac`
- portable parallel kMC simulator in OPENCL

References

[1] G. Arampatzis, M. A. Katsoulakis, P. Plecháč, M. Taufer, and L. Xu. Hierarchical fractional-step approximations and parallel kinetic monte carlo algorithms. *arXiv:1105.4673*, J. Comp. Phys. (2011)

[2] M. A. Katsoulakis, P. Plecháč, and A. Sopasakis. Error analysis of coarse-graining for stochastic lattice dynamics. *SIAM J. Numer. Anal.*, 44:2270–2296, 2006.

[3] T. G. Kurtz. A random trotter product formula. *Proc. Amer. Math. Soc.*, 35:147–154, 1972.

[4] Y. Shim and J. G. Amar. Semirigorous synchronous sublattice algorithm for parallel kinetic Monte Carlo simulations of thin film growth. *Phys. Rev. B*, 71(12):125432, 2005.

[5] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comp. Sim.*, 8(I):3-30, 1998.

[6] The OpenCL Specification, Version 1.1, Khronos Group Std.,