# Parallel Fractional Step kinetic Monte Carlo algorithms

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

[1]Department of Applied Mathematics, University of Crete
[2]Department of Mathematics and Statistics, University of Massachusetts
[3]Department of Mathematical Sciences, University of Delaware

February 14, 2013

# Outline

# Motivation

- Simulation of models that describe Many Particle Systems
  - Statistical physics (Ising type models)
  - Computational chemistry (chemical reactions)
  - Material Science
- Difficulties
  - Hard to simulate for large systems
  - Standards algorithms are not easy parallelizable

# Goals

- Development of a new mathematical framework for constructing **parallel algorithms** for lattice kinetic Monte Carlo (kMC) simulations, **that covers pre-existing methods**.

- Prove **convergence** of the algorithms

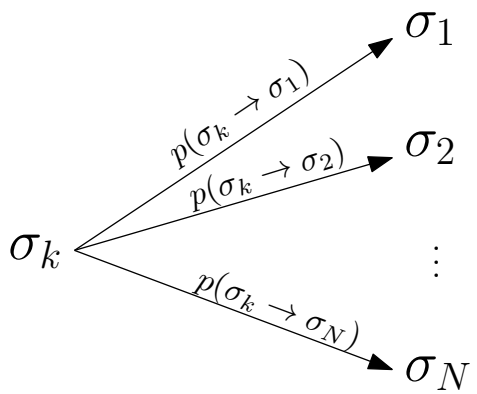
very short **Introduction to Stochastic Systems**

# General Stochastic Systems

- Define the set of all possible states that the system can be

$$\mathcal{S} = \{\sigma_1, \ldots, \sigma_N\}$$

- from every state there is a probability of moving to another state that depends only on the current state (**Markov assumption**)




- define the **Transition Probability Matrix**

$$P_{ij} = p(\sigma_i \to \sigma_j)$$

# General Stochastic Systems - Discrete time

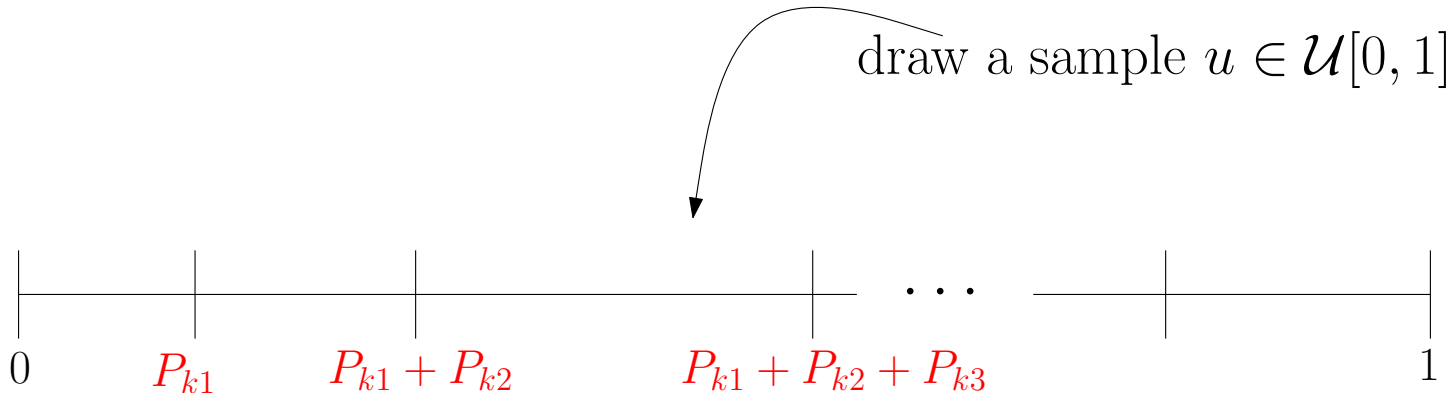
- In **discrete time** Stochastic Systems, the process jumps at equal time steps.

# General Stochastic Systems - Discrete time

- In **discrete time** Stochastic Systems, the process jumps at equal time steps.
- **Question:** given a state $\sigma_k$, how do I choose the next state?

# General Stochastic Systems - Discrete time

- In **discrete time** Stochastic Systems, the process jumps at equal time steps.
- **Question:** given a state $\sigma_k$, how do I choose the next state?




- In this example the new state is $\sigma_3$.

# General Stochastic Systems - Continuous time

- In **continuous time** Stochastic Markovian Systems we usually describe the **transition rates**, $c(\sigma_i \to \sigma_j)$, instead of the the transition probabilities.
- The transition probability is,

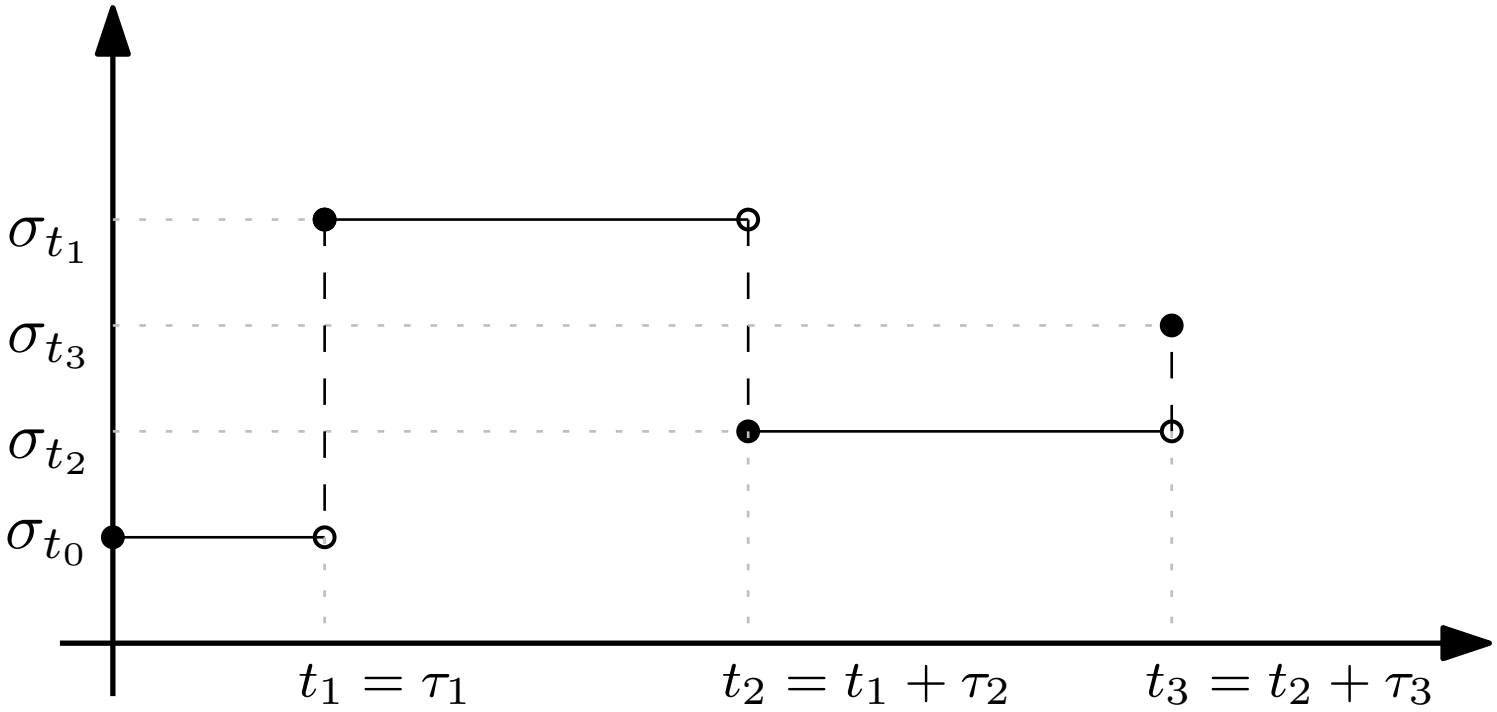
$$P_{ij} = \frac{c(\sigma_i \to \sigma_j)}{c_0(\sigma_i)}, \qquad c_0(\sigma_i) = \sum_j c(\sigma_i \to \sigma_j)$$

- But we need one more information: **how long does the system stays at the current state** $\sigma_i$ ?

$$\tau_i \sim \mathcal{E}\big(c_0(\sigma_i)\big)$$

This is why we call $c_0(\sigma_i)$ the **clock** at state $\sigma_i$.

# General Stochastic Systems - Continuous time




- What we already described, is known as the **Gillespie algorithm** or Stochastic Simulation Algorithm (**SSA**)

# General Stochastic Systems - Observables

- In order to get valuable information for the evolution of a system we define **observable functions** (or test functions)

$$f : \mathcal{S} \rightarrow \mathbb{R}$$

- $f$ must also be a bounded function on $\mathcal{S}$

$$f \in C_b(\mathcal{S})$$

# General Stochastic Systems - Observables


**Figure:** *One path of a stochastic system*

# General Stochastic Systems - Observables

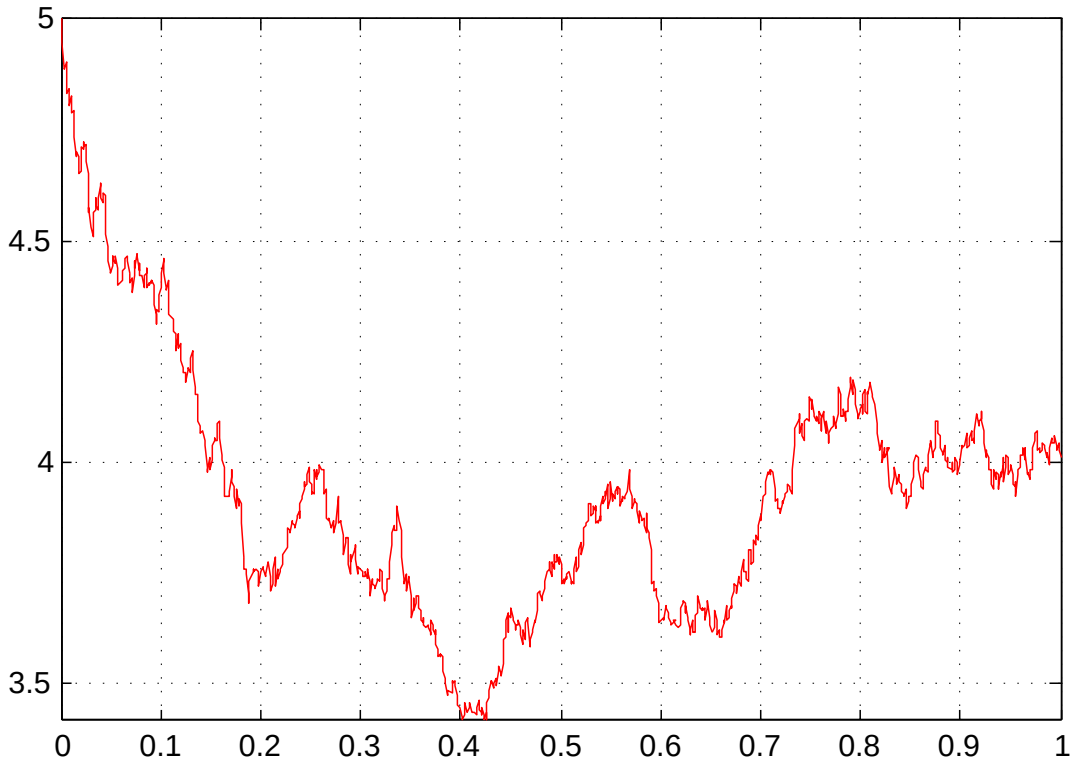



**Figure:** *Another path of a stochastic system*

# General Stochastic Systems - Observables

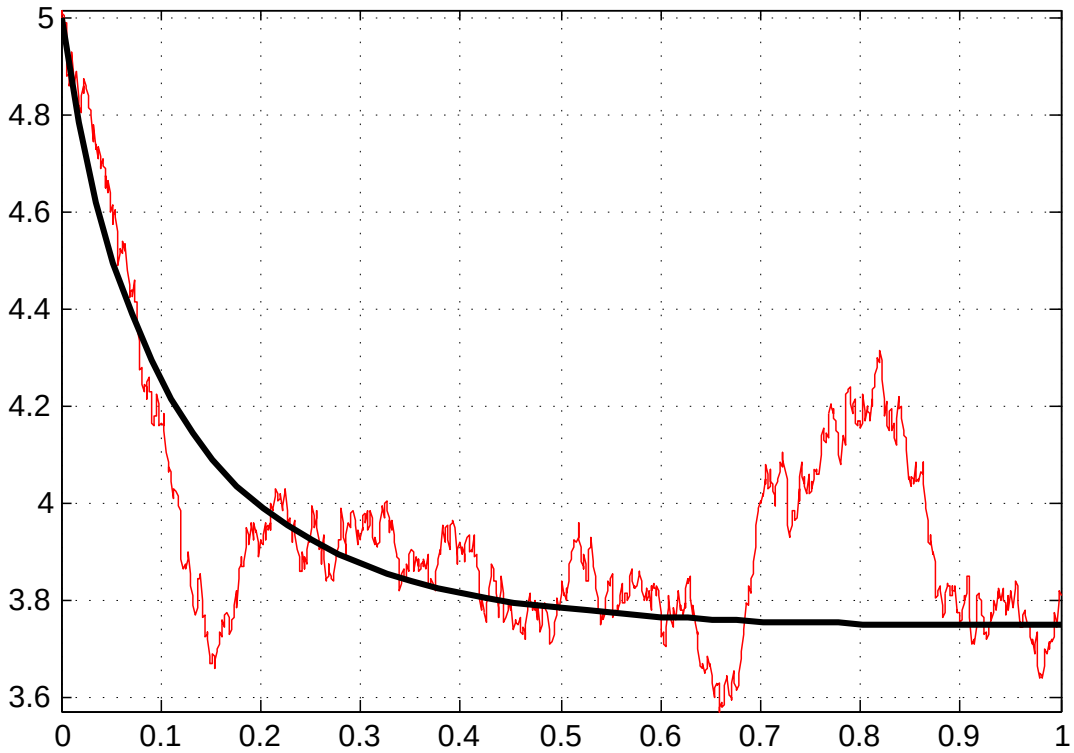



**Figure:** *Another path of a stochastic system and that* **fluctuates around the mean**.

# General Stochastic Systems - Mean Values

- The **mean value** of all realizations is defined as,

$$u(t, \sigma_0) := \mathbb{E}_{\sigma_0}[f(\sigma_t)]$$

- and satisfies the ODE

$$\frac{d}{dt} u(t, \sigma_0) = \mathcal{L} u(t, \sigma_0), \qquad u(0, \sigma_0) = f(\sigma_0).$$

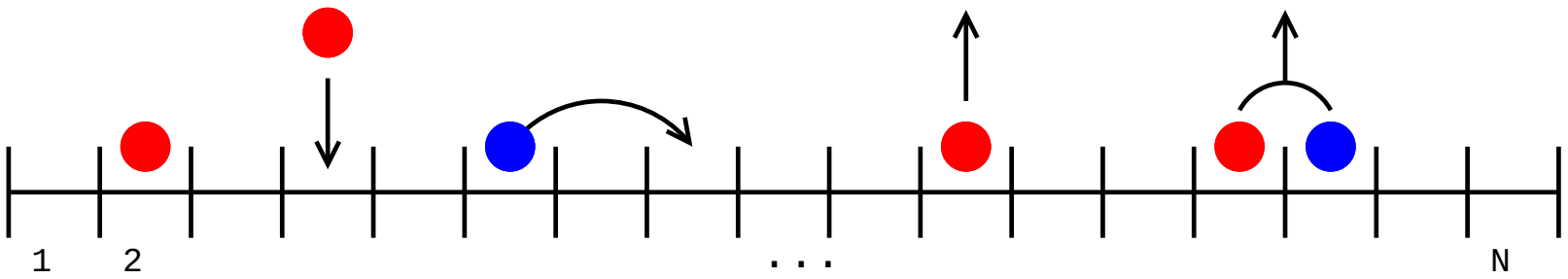
- **Generator** of the process

$$\mathcal{L} f(\sigma) = \sum_{\sigma' \in \mathcal{S}} c(\sigma \to \sigma') \big[ f(\sigma') - f(\sigma) \big]$$

# Lattice models - Description

Our setup is as follows:

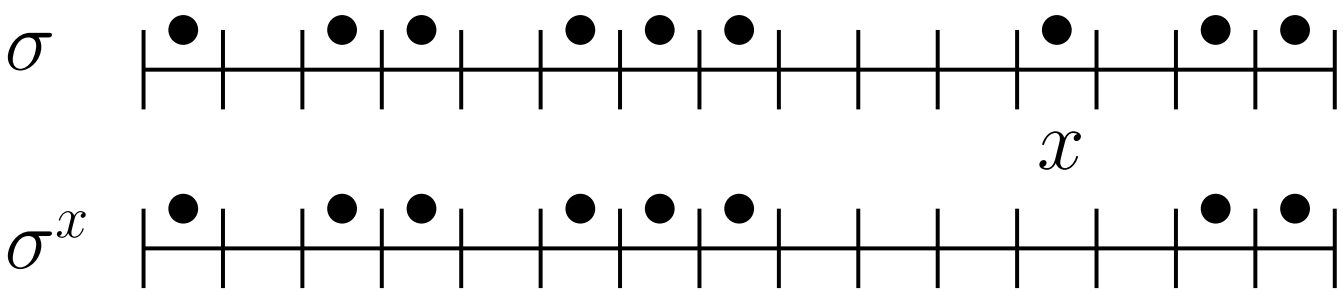
- a **lattice** $\Lambda$ with $N$ sites
- the **states** $\sigma : \Lambda_N \to \mathbb{R}$
  - $\sigma(x) = 0$ site x is vacant
  - $\sigma(x) = k \in \{1, 2, \ldots, K\}$ site x is occupied by species k
- particles can be adsorbed, desorbed, diffuse or react with certain rates



# Lattice models - Description

- A new configuration, $\sigma^x, \forall x \in \Lambda$, is defined as,

$$\sigma^x(y) := \begin{cases} \sigma^x(x) \neq \sigma(x), & y = x \\ \sigma^x(x) = \sigma(y), & y \neq x \end{cases}$$

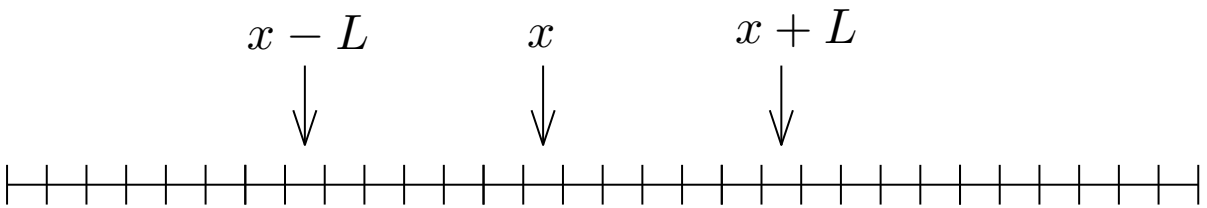


# Lattice models - Description

- The **rates** are

$$c(\sigma \to \sigma') := \begin{cases} c(x,\sigma), & \text{if } \sigma' = \sigma^x,\ x \in \Lambda \\ 0, & \text{if } \sigma' \neq \sigma^x,\ \forall x \in \Lambda \end{cases}$$

- If the rate function, $c(x,\sigma)$ depends on $x-L,\ldots,x+L$, then we call $L$ the **Length of Interaction**.



# Lattice models - Generator

- The **generator of a lattice system** is,

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

# Lattice models - Generator

- The **generator of a lattice system** is,

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

- compare to the general generator

$$\mathcal{L}f(\sigma) = \sum_{\sigma' \in \mathcal{S}} c(\sigma \to \sigma')\big[f(\sigma') - f(\sigma)\big]$$

**Fractional Step Method**

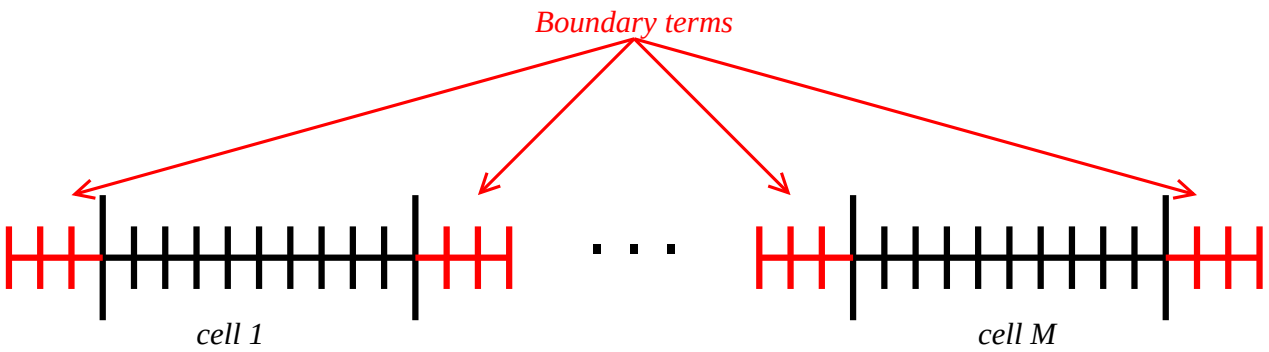
OR

**Operator Splitting Method**

# Lattice Decomposition

Decompose the lattice as follows,

- split the lattice into **M cells** each of them containing $\mathbf{q} = \frac{\mathbf{N}}{\mathbf{M}}$ sites
- The idea is to choose $M$ such that $q > 2L + 1$.




- In a mathematical notation

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

# Operator Decomposition

- The operator $\mathcal{L}$ can be decomposed as follows

$$\begin{aligned}\mathcal{L}f(\sigma) &= \sum_{k=1}^{M} \overbrace{\sum_{x \in C_k} c(x,\sigma)(f(\sigma^x) - f(\sigma))}^{\mathcal{L}_k} \\ &= \overbrace{\sum_{k \text{ odd}} \mathcal{L}_k f(\sigma)}^{\mathcal{L}^{\text{odd}}} + \overbrace{\sum_{k \text{ even}} \mathcal{L}_k f(\sigma)}^{\mathcal{L}^{\text{even}}}\end{aligned}$$

- We managed to split the operator in two to disjoint parts,

$$\mathcal{L} = \mathcal{L}^{\text{odd}} + \mathcal{L}^{\text{even}}$$

# Operator Decomposition

- The evolution of equation for the mean observable becomes,

$$\frac{d}{dt}u(t,\sigma_0) = \mathcal{L}^{\text{odd}}u(t,\sigma_0) + \mathcal{L}^{\text{even}}u(t,\sigma_0)$$

- and the solution can be written as

$$u(t,\sigma_0) = e^{t\mathcal{L}}f(\sigma_0) = e^{t(\mathcal{L}^{\text{odd}}+\mathcal{L}^{\text{even}})}f(\sigma_0).$$

- Note that in general

$$e^{t(\mathcal{L}^{\text{odd}}+\mathcal{L}^{\text{even}})} \neq e^{t\mathcal{L}^{\text{odd}}}e^{t\mathcal{L}^{\text{even}}}.$$

# Approximation Schemes

- Lie splitting,

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

# Approximation Schemes

- 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,

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

# Approximation Schemes

- 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,

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

- Random splitting (SPPARKS, Sandia National Lab.),

$$u(\Delta t) = e^{\Delta t \mathcal{L}} u(0) \approx e^{\Delta t \mathcal{L}^{1}} e^{\Delta t \mathcal{L}^{2}} u(0)$$

where $\mathcal{L}^1$ and $\mathcal{L}^2$ are either $\mathcal{L}^{\text{odd}}$ or $\mathcal{L}^{\text{even}}$ with equal probability.

# Approximation Schemes

- Trotter Product formula for semigroups (Proc. AMS 1958)

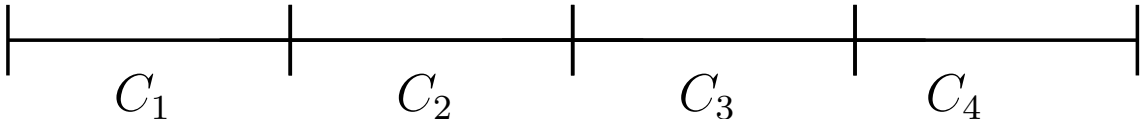
$$\lim_{\Delta t \to 0} \left[ e^{\Delta t A} e^{\Delta t B} \right]^{\frac{t}{\Delta t}} = e^{t(A+B)}.$$

- For the Lie splitting scheme

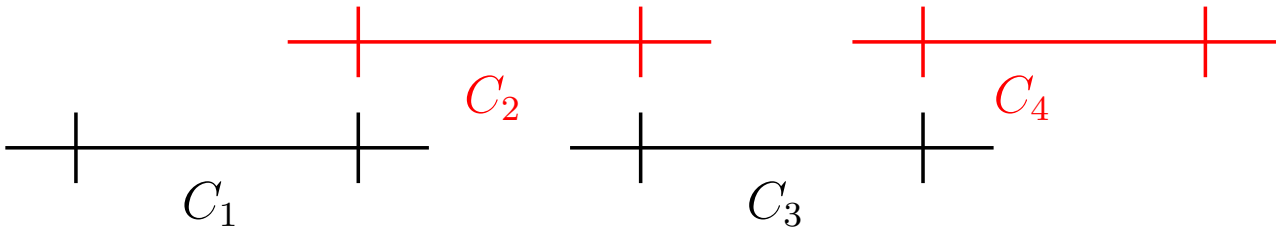
$$e^{t\mathcal{L}} \approx \left[ e^{\frac{t}{N}\mathcal{L}^{\text{odd}}} e^{\frac{t}{N}\mathcal{L}^{\text{even}}} \right]^{N}.$$

# Example: Lie Splitting

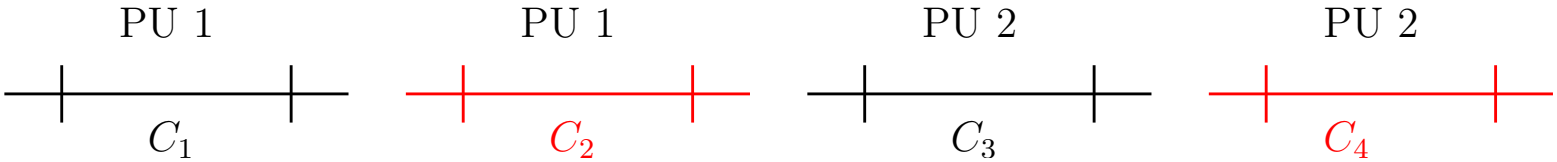
- split the lattice




into sublattices

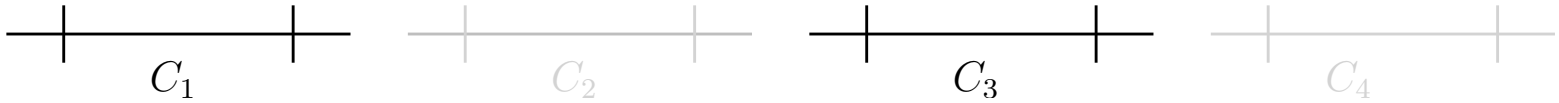



and distribute odd and even sublattices into different process units (PU)



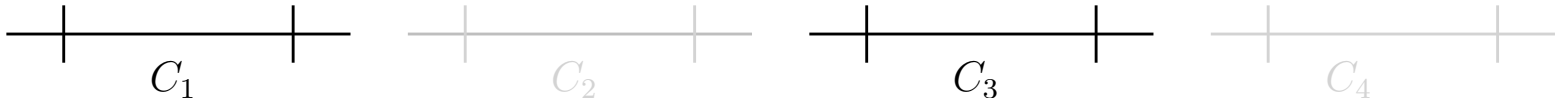
# Example: Lie Splitting

1. apply the SSA algorithm, up to time $\Delta t$, on the **odd cells**

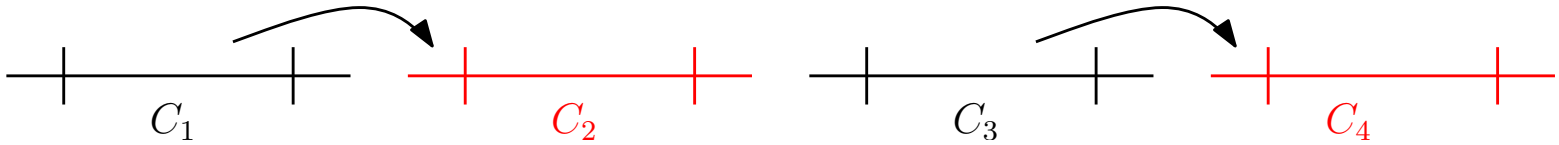


# Example: Lie Splitting

1. apply the SSA algorithm, up to time $\Delta t$, on the **odd cells**

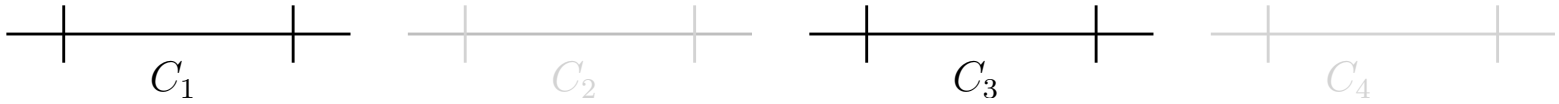



2. and **then communicate** the boundary information

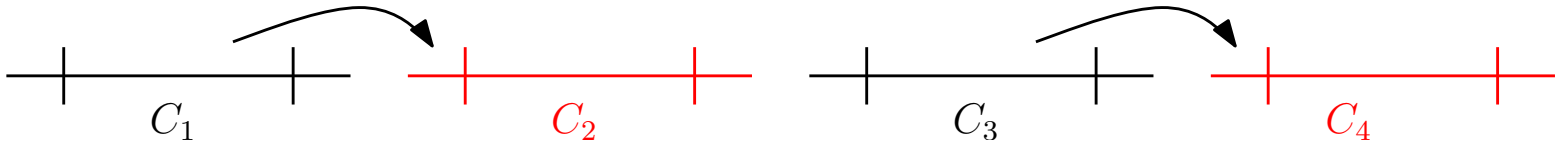


# Example: Lie Splitting

1. apply the SSA algorithm, up to time $\Delta t$, on the **odd cells**



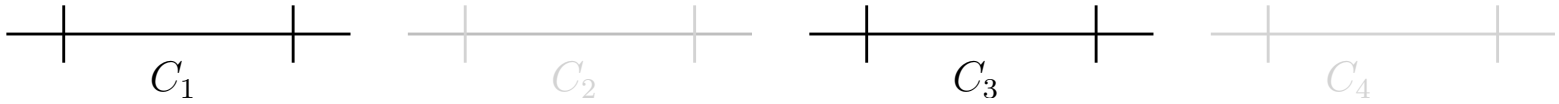

2. and **then communicate** the boundary information



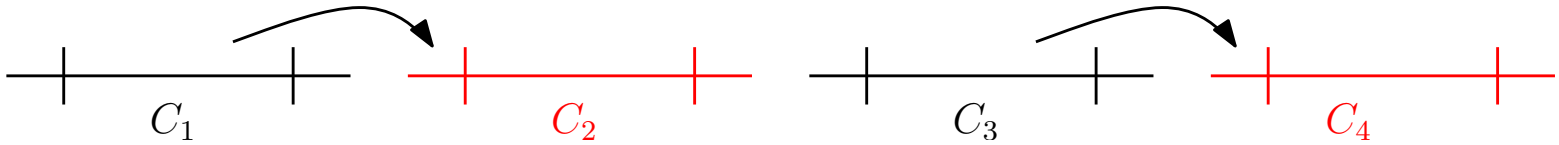

3. apply the SSA algorithm, up to time $\Delta t$, on the **even cells**

# Example: Lie Splitting

1. apply the SSA algorithm, up to time $\Delta t$, on the **odd cells**

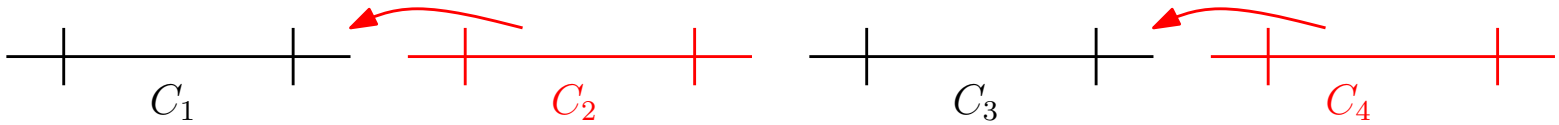



2. and **then communicate** the boundary information



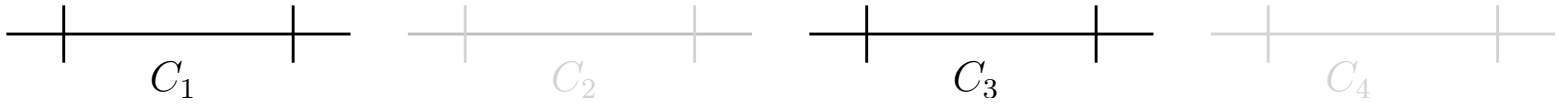

3. apply the SSA algorithm, up to time $\Delta t$, on the **even cells**


4. and **then communicate** the boundary information

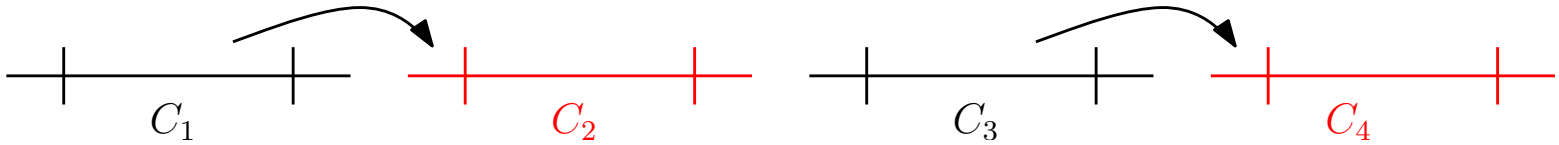


# Example: Lie Splitting

1. apply the SSA algorithm, up to time $\Delta t$, on the **odd cells**

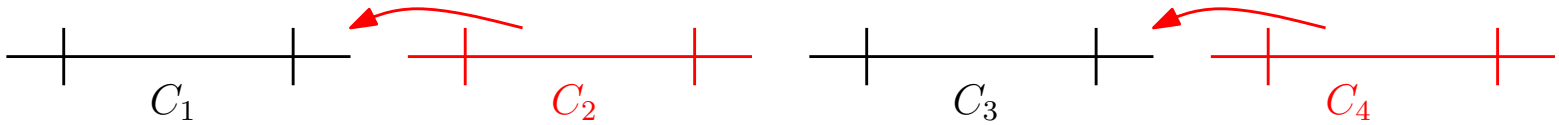



2. and **then communicate** the boundary information




3. apply the SSA algorithm, up to time $\Delta t$, on the **even cells**


4. and **then communicate** the boundary information




5. repeat steps 1-4 until time $T = n\Delta t$

# Error Analysis - Local Error

- Define $\mathcal{L}_1 = \mathcal{L}^{\text{odd}}$ and $\mathcal{L}_2 = \mathcal{L}^{\text{even}}$ and the **commutator** or **Lie Bracket**

$$[\mathcal{L}_1, \mathcal{L}_2] = \mathcal{L}_1\mathcal{L}_2 - \mathcal{L}_2\mathcal{L}_1.$$

- Let $P_L(t)$ and $P_S(t)$ be the semigroups, associated with the Lie and Strand splitting respectively,

$$u(\Delta t) \approx P_L(\Delta t)u_0 = e^{\Delta t\mathcal{L}_1} e^{\Delta t\mathcal{L}_2} u_0$$

$$u(\Delta t) \approx P_S(\Delta t)u_0 = e^{\frac{\Delta t}{2}\mathcal{L}_1} e^{\Delta t\mathcal{L}_2} e^{\frac{\Delta t}{2}\mathcal{L}_1} u_0$$

# Error Analysis - Local Error - Lie

- The **local error** for the Lie scheme is bounded by

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

where

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

# Error Analysis - Local Error - Strang

- for the Strang scheme,

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

where

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

# Error Analysis - Local Error - Random

- for the Random scheme

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

# Error Analysis - Global Error

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

$$\|P_L(t_n)u_0 - u(t_n)\| \leq C_1 \max_{k=0,\ldots,n} \|[\mathcal{L}_1, \mathcal{L}_2]u(t_k)\|\Delta t + C_2\tilde{\mathcal{R}}_L(u)\Delta t^2$$

where

$$\tilde{\mathcal{R}}_L(u) = \max_{k=0,\ldots,n} \mathcal{R}_L(u(t_k))$$

# Error Analysis - Global Error - Lie

- Our purpose is to simulate large systems (size of the lattice $|\Lambda| \sim 10^8$)
- We are interested in investigating how the terms

$$\|[\mathcal{L}_1, \mathcal{L}_2]u(t_k)\|$$

and

$$\tilde{\mathcal{R}}_L\big(u(t_k)\big) = c_2 \sum_{|m|=3} \|\mathcal{L}_1^{m_1}\mathcal{L}_2^{m_2}u(t_k)\|$$

behave as $N \to \infty$.

# Error Analysis - Macroscopic Observables

## Definition (Discrete derivatives)

For $x \in \Lambda$ we write,

$$\delta_x f(\sigma) = f(\sigma^x) - f(\sigma)$$

for the first order discrete derivative.
For $\mathbf{x} = (x_1, \ldots, x_m) \in \Lambda^m$ we introduce the notation

$$\delta_{\mathbf{x}} f(\sigma) = \delta_{x_1} \ldots \delta_{x_m} f(\sigma) = \delta_{x_1 \ldots x_m} f(\sigma)\,,$$

and we will refer to it as the *discrete derivative* of $f$ with respect to $\mathbf{x}$. For example if $\mathbf{x} = (x, y)$ then

$$\delta_{xy} f(\sigma) = \delta_x \delta_y f(\sigma) = f(\sigma^{xy}) - f(\sigma^x) - f(\sigma^y) + f(\sigma)\,.$$

# Error Analysis - Macroscopic Observables

## Definition (Macroscopic Observables)

Let $\mathbf{x} = (x_1, \ldots, x_m) \in \Lambda^m$ and $f \in C_b(\mathcal{S})$. Then we define the norm

$$\|f\|_m = \sum_{x_1 \in \Lambda} \cdots \sum_{x_m \in \Lambda} \|\delta_{\mathbf{x}} f\|_\infty ,$$

and the function space

$$C^m(\mathcal{S}) := \{f \in C_b(\mathcal{S}) \,|\, \sum_{k=1}^{m} \|f\|_k \leq C_f \text{ and } C_f \text{ is independent of } N\}$$

We will refer to elements of $C^m(\mathcal{S})$ as *macroscopic observables*.

# Error Analysis - Main Theorem

## Theorem

*(a) Let* $u(t)$ *be the solution of* $\frac{d}{dt}u(t,\sigma_0) = \mathcal{L}u(t,\sigma_0)$ *with* $u(0) = f \in C^3(\mathcal{S})$ *is the observable. Then for the global error estimate for the Lie scheme,*

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

*holds that*

$$\|[\mathcal{L}_1,\mathcal{L}_2]u(t_k)\| < CML^{d+1}\,,$$

*and*

$$\mathcal{R}_L(u) < \tilde{C}\,,$$

*where both constants* $C$ *and* $\tilde{C}$ *are independent of the dimension of the lattice* $\Lambda$ *and* $d$ *is the dimension of the lattice* $\Lambda$*.*

# Error Analysis - Main Theorem

## Theorem

*(b) Many macroscopic observables are not just in $C^3(\mathcal{S})$ but also satisfy a local bound such as,*

$$\max_{z\in\Lambda} \|\delta_z u(0,\cdot)\|_\infty + \max_{x,y\in\Lambda} \|\delta_{xy} u(0,\cdot)\| + \max_{x,y,z\in\Lambda} \|\delta_{xyz} u(0,\cdot)\| \leq \frac{C}{N},$$

*Then the bounds for the commutators become*

$$\|[\mathcal{L}_1, \mathcal{L}_2]u(t,\cdot)\|_\infty \leq C\,\frac{L^{d+1}}{q}.$$

# Error Analysis - Macroscopic Observables

- **Question:** are the constraints on the initial data too tight?
- **Answer:** All known and useful observable functions, (e.g. coverage, spatial correlations, variance, surface roughness ) fall into these categories.

# Error Analysis - Order of Convergence

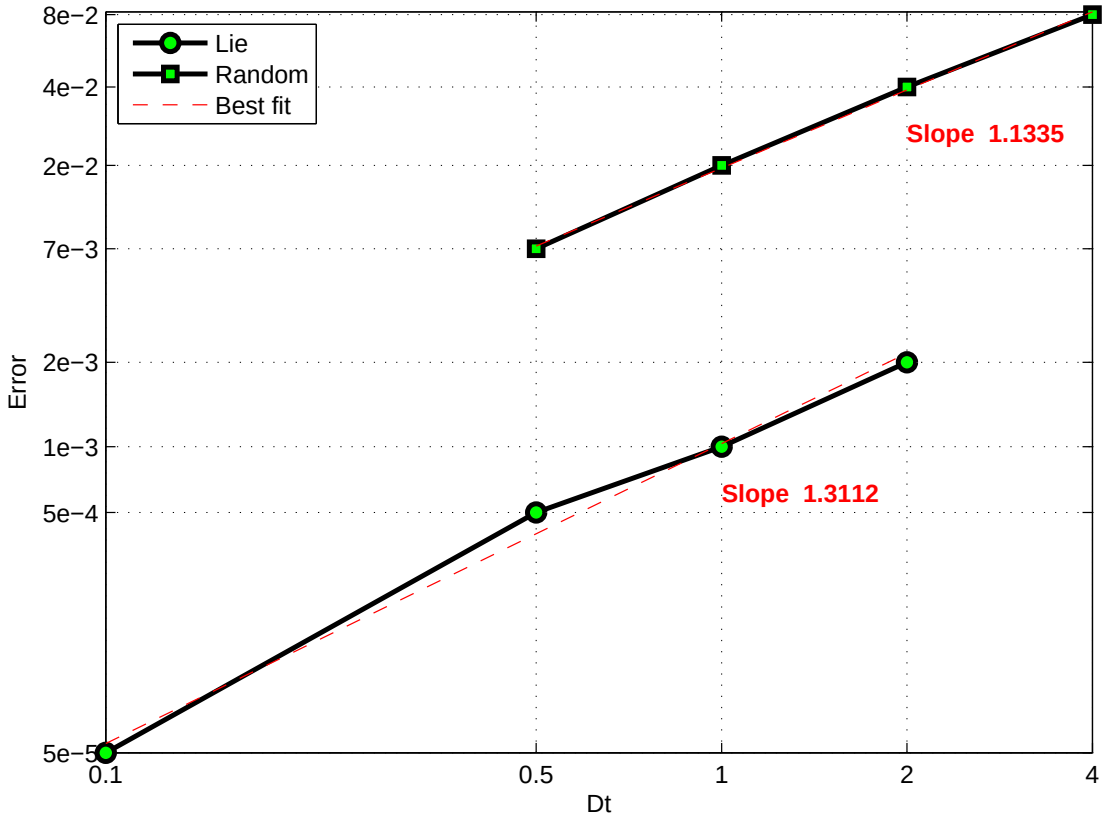



**Figure:** *Order of convergence for the Lie and the Random scheme. Note that the Random scheme has larger error for the same value of* $\Delta t$.

# Error Analysis - Error VS sublattice size

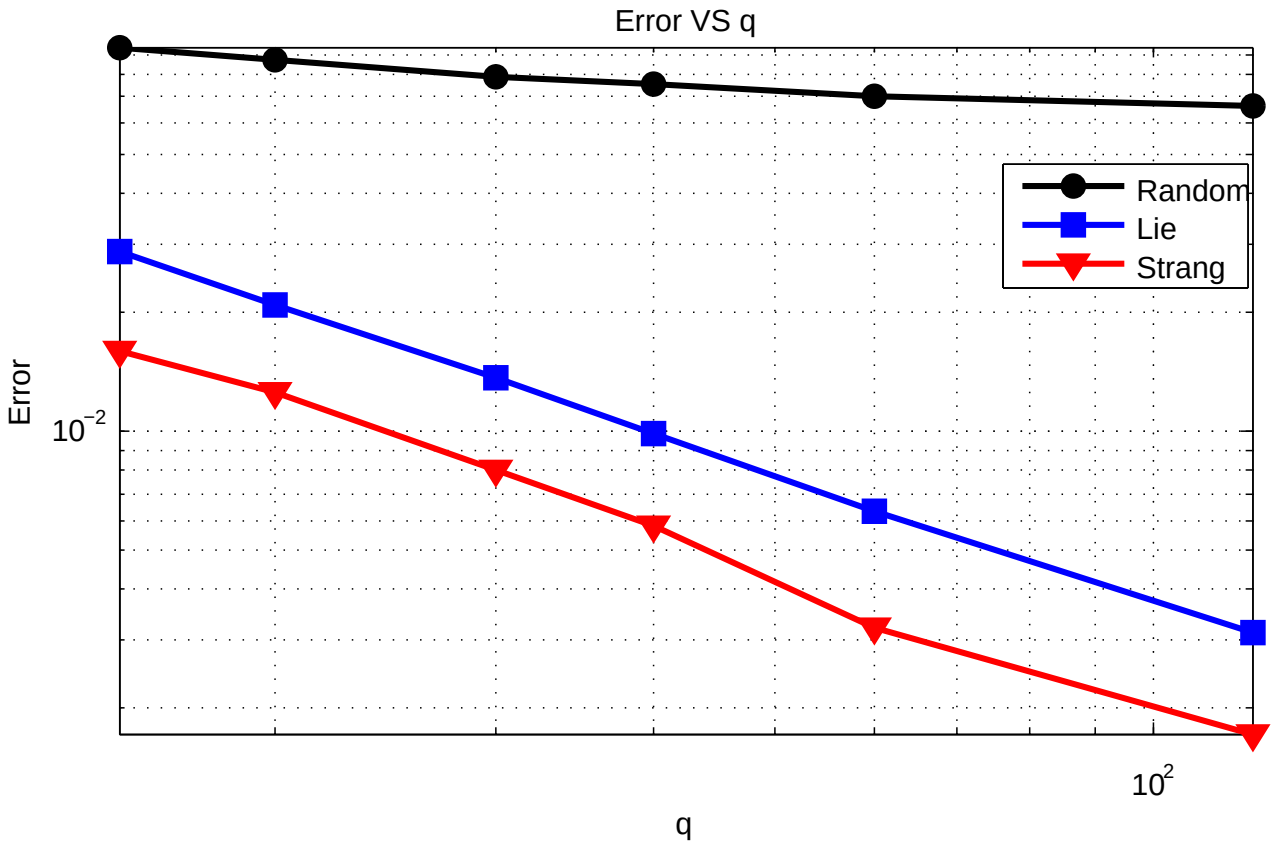



**Figure:** *For the Lie and the Strang schemes the error scales with the size of the sublattice.*

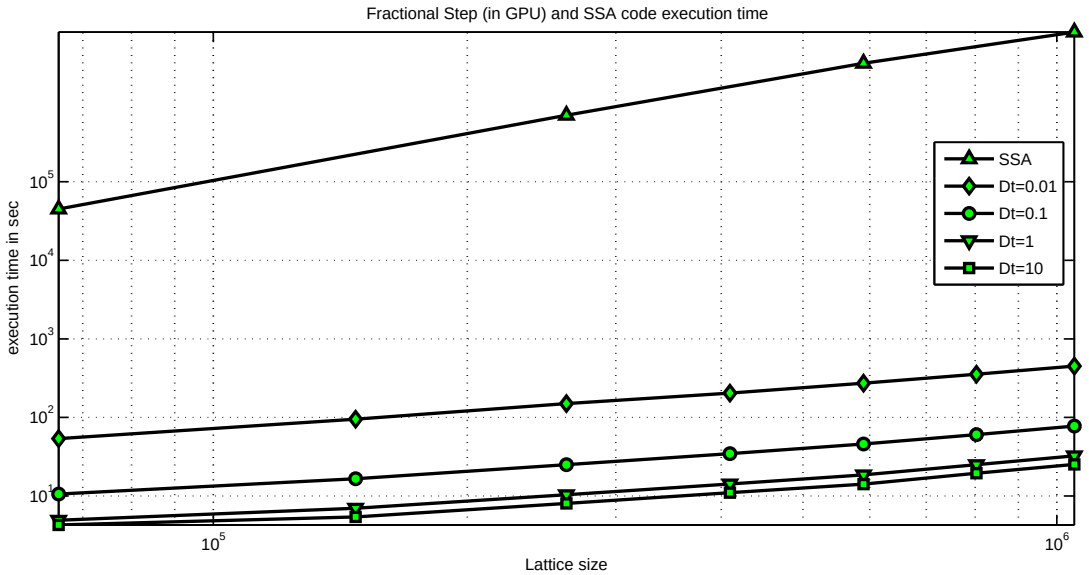
# Implementation and Application

# Implementation

- GPGPU
- Implemented in CUDA and OpenCl
- cluster of GPUs (CUDA + MPI)

# Implementation - Weak Scaling

- The following results were obtained in a **Fermi nVidia GPU**.
- Fractional-step alg. is about $\mathbf{10^5}$ **times faster** than the SSA.



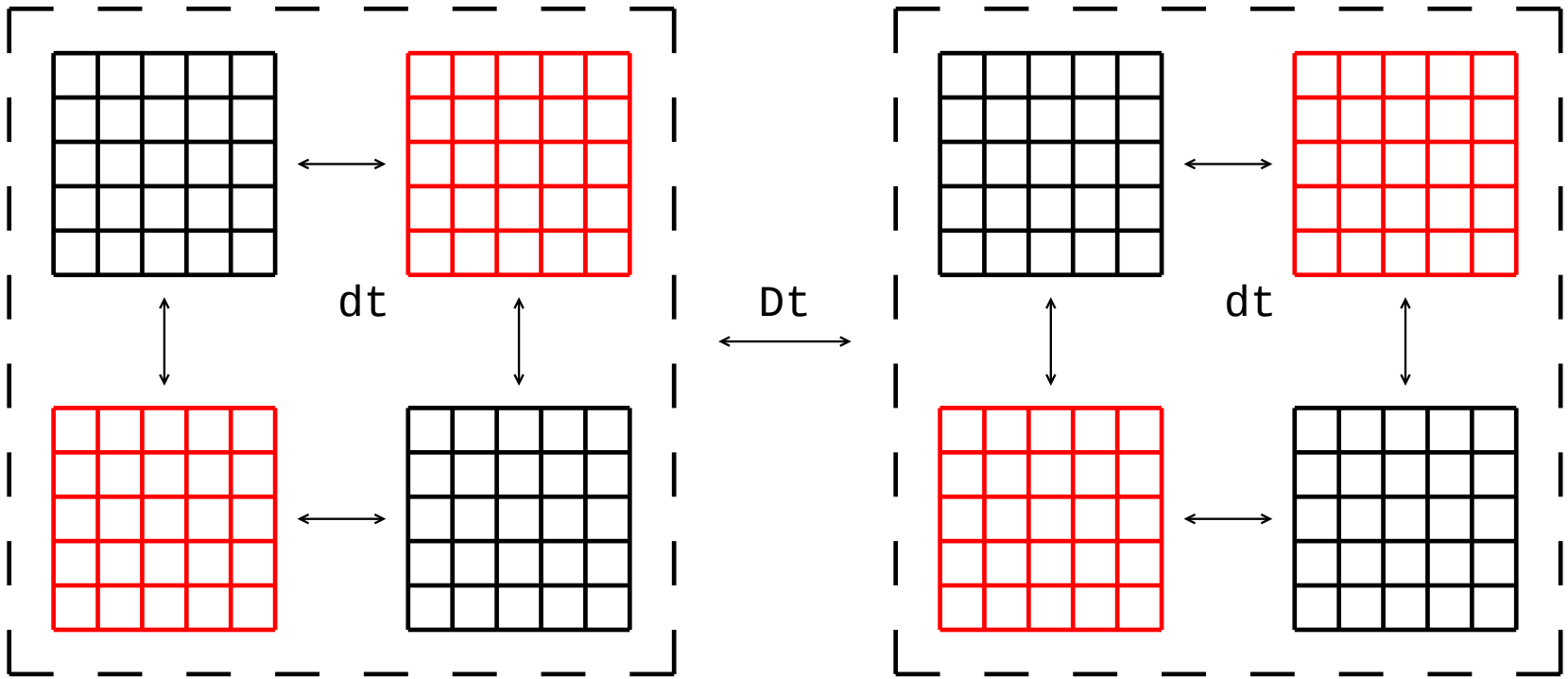

- Note that : **Large $\Delta$t $\Rightarrow$ large error** but also **less communication**.

# Application - Catalytic Reactions

In order to run 2 dimensional large systems ($N = 10^8$),

- split the lattice into large parts and distribute them into a cluster of CPUs
- the sublattice in every CPU node runs on a GPU
- apply the Fractional-step algorithm on the "CPU-lattices" using $\Delta t$ much larger that the $\delta t$ used in the "GPU-lattices"
- error may be driven by the large $\Delta t$ but this selection ensures **less communication** between CPUs.

# Application - Catalytic Reactions



# Application - Catalytic Reactions

The following simulation is a $\mathbf{CO_2}$ **oxidation model** on a catalytic surface with the following reactions

$$\mathrm{CO}\downarrow, \quad \mathrm{O_2}\downarrow, \quad \mathrm{CO} + \mathrm{O} \rightarrow \mathrm{CO_2}\uparrow$$