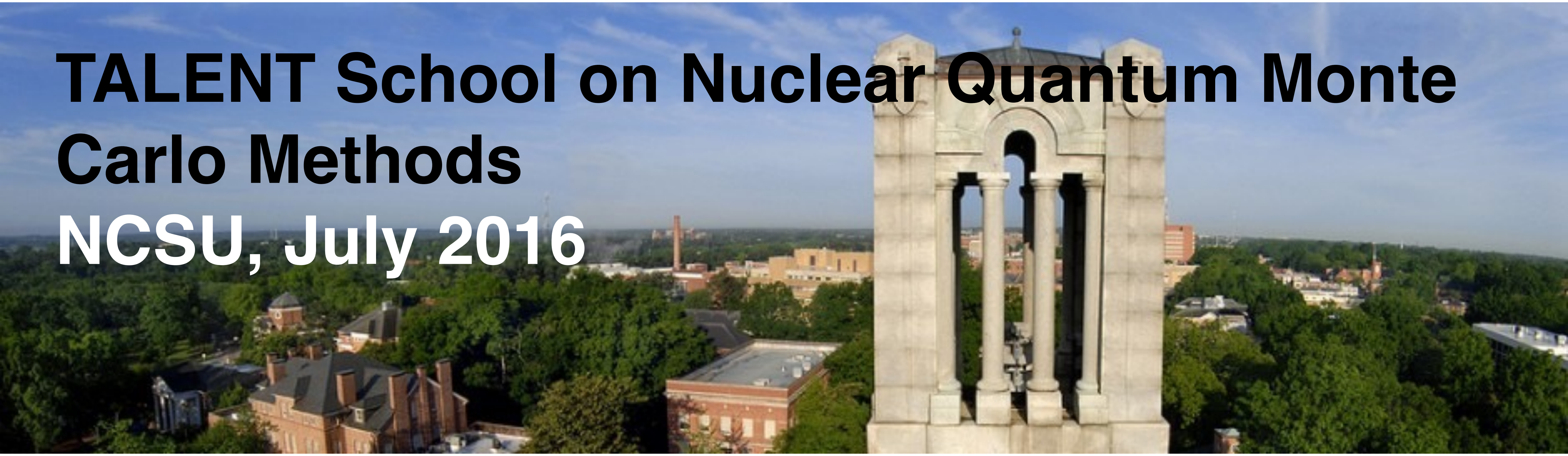
Finite-temperature lattice methods
Lecture 3.

Joaquín E. Drut

University of North Carolina at Chapel Hill

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL



TALENT School on Nuclear Quantum Monte Carlo Methods

NCSU, July 2016

Goals

- **Lecture 1**:
  General motivation. Review of thermodynamics and statistical mechanics. Non-interacting quantum gases at finite temperature.

- **Lecture 2**:
  QRL1. Formalism at finite temperature: Trotter-Suzuki decompositions; Hubbard-Stratonovich transformations. From traces to determinants. Bosons and fermions. The sign problem.

- **Lecture 3**:
  QRL2. Computing expectation values (finally!). Particle number. Energy. Correlation functions: static and dynamic. Sampling techniques. Signal-to-noise issues and how to overcome them.

Goals

- **Lecture 4**:
  Quantum phase transitions and quantum information. Entanglement entropy. Reduced density matrix. Replica trick. Signal-to-noise issues.

- **Lecture 5**:
  QRL3. Finite systems. Particle-number projection. The virial expansion. Signal-to-noise issues. Trapped systems.

- **Lecture 6**:
  QRL5. Spacetime formulation. Finite-temperature perturbation theory on the lattice.

- **Lecture 7**:
  Applications to ultracold atoms in a variety of situations. Beyond equilibrium thermodynamics.

Quick review of Lecture 2

The problem: interacting vs. non-interacting

$$\hat{H} = \hat{T} + \hat{V} \qquad\qquad [\hat{H}, \hat{N}] = 0$$

$$[\hat{T}, \hat{V}] \neq 0$$

In the non-interacting case, the Hamiltonian is trivially diagonal in all N-particle subspaces.

In the interacting case, each N needs to be diagonalized independently, and the dimension grows exponentially.

so… we need to do something different…

Towards thermodynamics on the lattice

Objective

$$\langle \hat{\mathcal{O}} \rangle = \frac{1}{\mathcal{Z}} \mathrm{Tr}\left[\hat{\mathcal{O}} e^{-\beta(\hat{H}-\mu\hat{N})} \right] \qquad \mathcal{Z} \equiv \mathrm{Tr}\left[e^{-\beta(\hat{H}-\mu\hat{N})} \right]$$

We can, at least formally, always write a generating functional:

$$\mathcal{Z}[j] = \mathrm{Tr}\left[e^{-\beta(\hat{H}-\mu\hat{N})+j\hat{\mathcal{O}}} \right]$$

such that

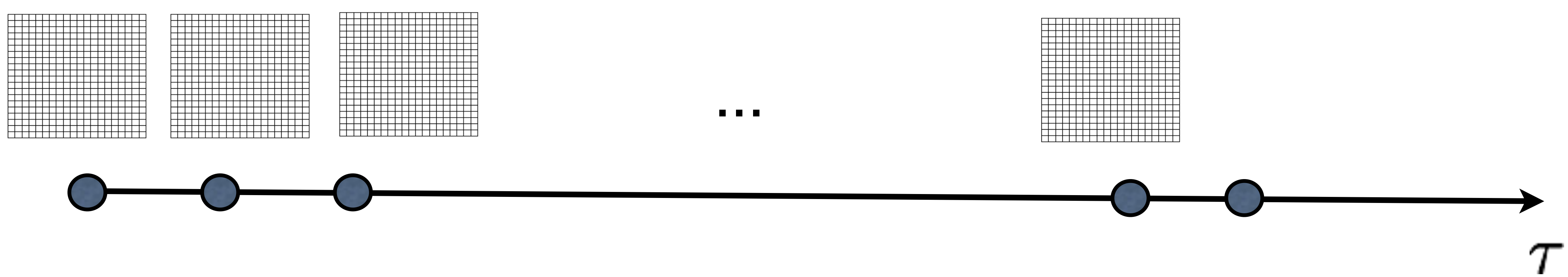
$$\langle \mathcal{O} \rangle = \left. \frac{\delta \log \mathcal{Z}[j]}{\delta j} \right|_{j=0}$$

→ It is useful to focus on the **partition function** $\mathcal{Z}$

Imaginary time and Trotter-Suzuki factorization

Discretize time

$$\exp(-\beta\hat{H}) = \left[\exp(-\tau\hat{H})\right]^{N_\tau}$$



Trotter-Suzuki factorization

$$\exp(-\tau\hat{H}) = \exp(-\tau\hat{T}/2)\,\exp(-\tau\hat{V})\,\exp(-\tau\hat{T}/2)$$

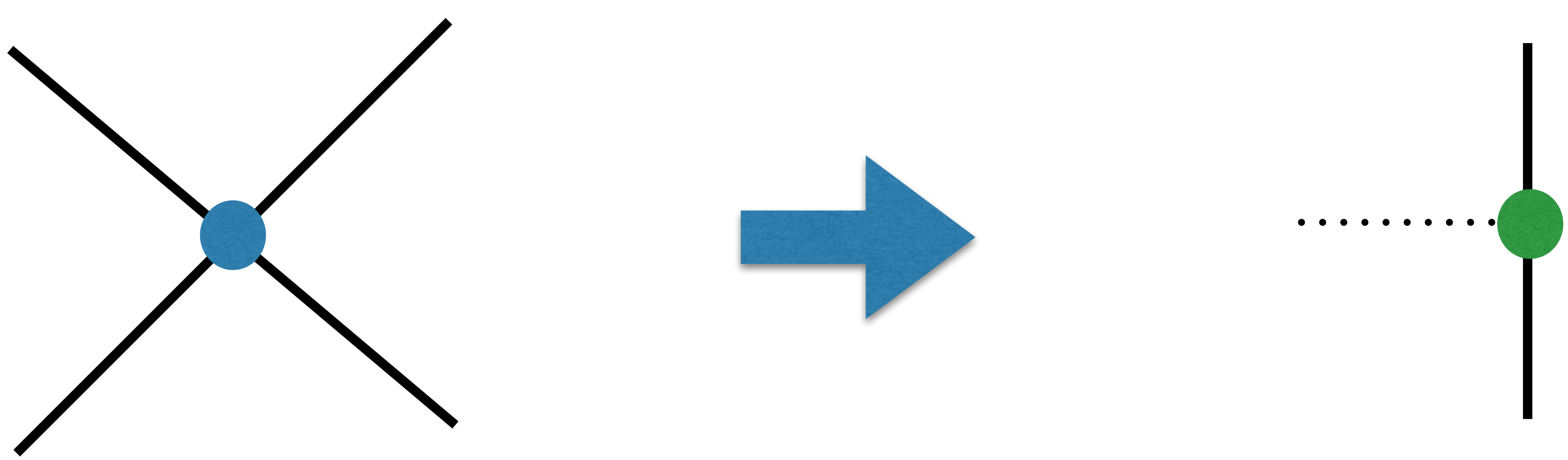
The potential energy factor is of course where all our problems are.

The Hubbard-Stratonovich transformation

Exponential of two-body operator becomes a field integral over all possible external fields:

$$\exp(-\tau\hat{V}) = \int \mathcal{D}\sigma \exp(-\tau\hat{V}_{\text{ext}}[\sigma])$$

Diagrammatically…



(The field integral "builds" the other half)

Going back to the transfer matrix...

... putting everything back together, we obtain

$$\mathcal{Z} \equiv \mathrm{Tr}\left[e^{-\beta(\hat{H}-\mu\hat{N})}\right]$$

$$\mathcal{Z} = \mathrm{Tr}\left[\int \prod_i d\sigma_i \; \mathcal{T}[\sigma]\right] = \int \mathcal{D}\sigma \; \mathrm{Tr}\, \mathcal{T}[\sigma] \qquad \mathcal{D}\sigma \equiv \prod_i d\sigma_i$$

where $\mathcal{T}[\sigma] = \mathcal{T}_{\uparrow}[\sigma]\mathcal{T}_{\downarrow}[\sigma]$

$$\mathcal{T}_s[\sigma] = \hat{U}_1\hat{U}_2\hat{U}_3\ldots\hat{U}_{N_\tau}$$

$$\hat{U}_t = \exp(-\tau\hat{T}/2)\exp(-\tau\hat{V}_{\mathrm{ext}}[\sigma])\exp(-\tau\hat{T}/2)$$

$$\mathrm{Tr}\,\mathcal{T}[\sigma] = \det M_{\uparrow}M_{\downarrow}$$

$$= \det(1+W_{\uparrow})\det(1+W_{\downarrow})$$

$$W_s = U_1U_2U_3\ldots U_{N_\tau}$$

After the dust settled…

… we managed to write

$$\mathcal{Z} = \int \mathcal{D}\sigma \; \mathcal{P}[\sigma] \qquad \text{where} \qquad \mathcal{P}[\sigma] = \det(1 + W_\uparrow) \det(1 + W_\downarrow)$$

$$W_s = U_1 U_2 U_3 \ldots U_{N_\tau}$$

If we put a source, take a derivative of the log, and set the source to zero, we will always end up with something of the form

$$\langle O \rangle = \left. \frac{\delta \log \mathcal{Z}[j]}{\delta j} \right|_{j=0} = \frac{1}{\mathcal{Z}} \left. \frac{\delta \mathcal{Z}[j]}{\delta j} \right|_{j=0} = \frac{1}{\mathcal{Z}} \int \mathcal{D}\sigma \; \mathcal{P}[\sigma] O[\sigma]$$

OK, but what do we do with this now?

Computing observables

Computing expectation values

- We managed to get things to this point:

$$\langle O \rangle = \left. \frac{\delta \log \mathcal{Z}[j]}{\delta j} \right|_{j=0} = \frac{1}{\mathcal{Z}} \left. \frac{\delta \mathcal{Z}[j]}{\delta j} \right|_{j=0} = \frac{1}{\mathcal{Z}} \int \mathcal{D}\sigma \; \mathcal{P}[\sigma] O[\sigma]$$

where $\mathcal{Z} = \int \mathcal{D}\sigma \; \mathcal{P}[\sigma]$

Computing expectation values

- We managed to get things to this point:

$$\langle O \rangle = \frac{\delta \log \mathcal{Z}[j]}{\delta j}\bigg|_{j=0} = \frac{1}{\mathcal{Z}} \frac{\delta \mathcal{Z}[j]}{\delta j}\bigg|_{j=0} = \frac{1}{\mathcal{Z}} \int \mathcal{D}\sigma \; \mathcal{P}[\sigma] O[\sigma]$$

where $\mathcal{Z} = \int \mathcal{D}\sigma \; \mathcal{P}[\sigma]$

- If we are given samples of the field σ that obey the probability measure, then:

$$\langle O \rangle = \frac{1}{N_\sigma} \sum_{\{\sigma\}} O[\sigma]$$

with an uncertainty of order $\mathcal{O}\left(\frac{1}{\sqrt{N_\sigma}}\right)$

Computing expectation values

- Simplest observables we discussed:

$$\langle \hat{N} \rangle = \frac{\partial(-\beta\Omega)}{\partial(\beta\mu)} = \frac{\partial(\ln \mathcal{Z})}{\partial(\beta\mu)}$$

$$\langle \hat{H} \rangle = -\frac{\partial(\ln \mathcal{Z})}{\partial\beta}$$

but…

$$\mathcal{Z} = \int \mathcal{D}\sigma \det M_{\uparrow}[\sigma] M_{\downarrow}[\sigma]$$

How do we differentiate a determinant?

Computing expectation values

- Taking derivatives of determinants

$$\det A(x) = \exp(\operatorname{tr} \ln A(x))$$

Exercise 1: Use the above to obtain expressions
for the derivatives of the determinant w.r.t. x

Computing expectation values

- More complicated observables

$$\langle \hat{a}_i^\dagger \hat{a}_j \rangle = \frac{\partial \ln \mathcal{Z}[\lambda]}{\partial \lambda}$$

(assume spin-up operators)

$$\mathcal{Z}[\lambda] = \mathrm{Tr}\left[e^{-\beta\left(\hat{H}-\mu\hat{N}\right)} e^{\lambda \hat{a}_i^\dagger \hat{a}_j} \right] = \int \mathcal{D}\sigma \, \det M_\uparrow[\sigma,\lambda] \, \det M_\downarrow[\sigma]$$

Exercise 2: What is the form of the matrices?

Exercise 3: How do we obtain even more complicated observables?

Exercise 4: How would you obtain $\langle \hat{a}_j \hat{a}_i^\dagger \rangle$?

Sampling techniques

Algorithms

- We have identified a probability measure for our problem. What we need is to design a way to sample it.

Algorithms

- We have identified a probability measure for our problem. What we need is to design a way to sample it.
- This is a whole chapter by itself for a good reason: Sampling algorithms need to be efficient.

Algorithms

- We have identified a probability measure for our problem. What we need is to design a way to sample it.
- This is a whole chapter by itself for a good reason: Sampling algorithms need to be efficient.
- Many of the algorithms currently in use are not very efficient. But the problems one can treat with them are correspondingly very limited!

Algorithms

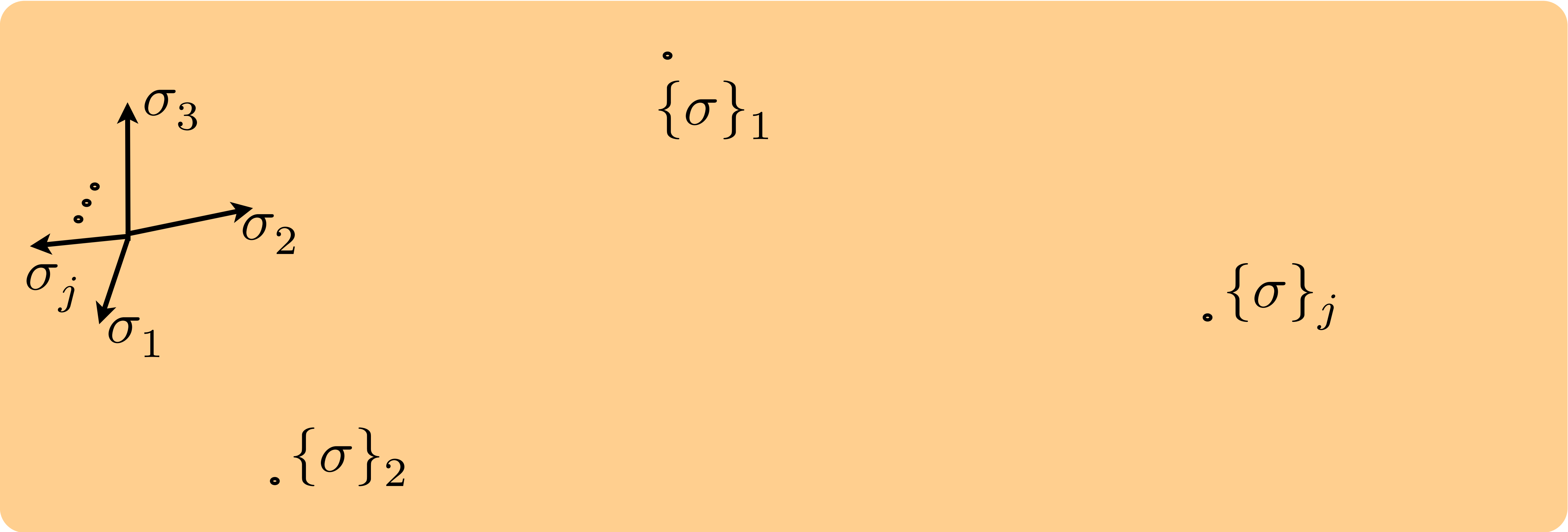
- We have identified a probability measure for our problem. What we need is to design a way to sample it.
- This is a whole chapter by itself for a good reason: Sampling algorithms need to be efficient.
- Many of the algorithms currently in use are not very efficient. But the problems one can treat with them are correspondingly very limited!
- In particular, the whole Lattice QCD effort would not be possible without efficient sampling algorithms.

So, how do we proceed?

Algorithms - ideally

- We would like to obtain samples of our probability distribution in the same way as we obtain random numbers from a generator, and (perhaps, why not) with the same good properties of decorrelation, reproducibility, etc.

- In our immense space of possible configurations, our probability represents the likelihood that the system will be here or there...



Algorithms - in practice

In practice it is extremely difficult to sample directly from such a complicated probability distribution.

Given a configuration $\{\sigma\}_j$ we can compute its probability *relative* to another configuration (we do not know the normalization!), and that is almost all we can do.

Algorithms - in practice

In practice it is extremely difficult to sample directly from such a complicated probability distribution.

Given a configuration $\{\sigma\}_j$ we can compute its probability *relative* to another configuration (we do not know the normalization!), and that is almost all we can do.

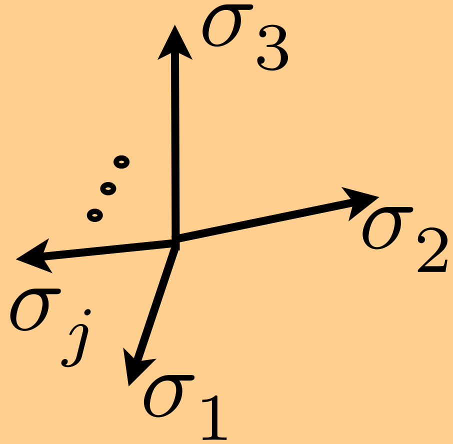
The work of Metropolis (et al). devised a way to use such relative probabilities to generate a chain of configurations that approximate $\mathcal{P}[\sigma]$

Algorithms - in practice

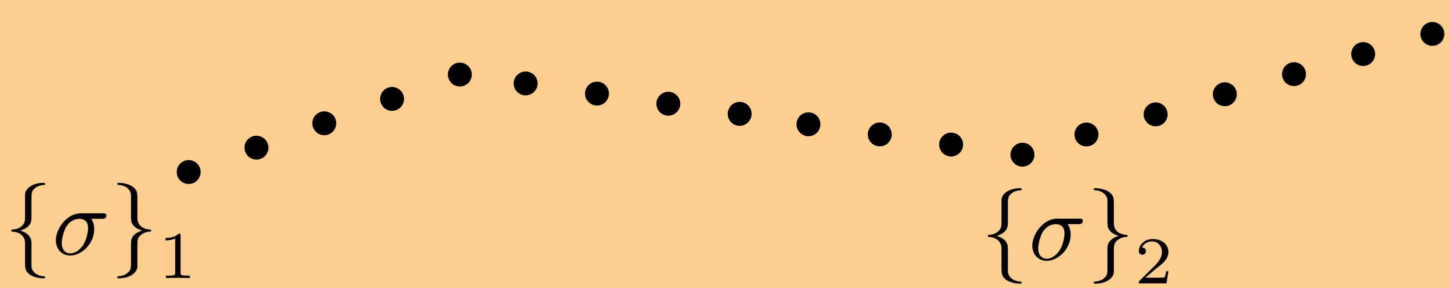
In practice it is extremely difficult to sample directly from such a complicated probability distribution.

Given a configuration $\{\sigma\}_j$ we can compute its probability *relative* to another configuration (we do not know the normalization!), and that is almost all we can do.

The work of Metropolis (et al). devised a way to use such relative probabilities to generate a chain of configurations that approximate $\mathcal{P}[\sigma]$



This is known today as the **Metropolis algorithm**



The Metropolis Algorithm

- Form a (Markov) chain of configurations:
  - **Start** from some arbitrary initial configuration σ_0

The Metropolis Algorithm

- Form a (Markov) chain of configurations:
  - **Start** from some arbitrary initial configuration σ_0
  - **Propose** a new one σ_1

The Metropolis Algorithm

- Form a (Markov) chain of configurations:
  - **Start** from some arbitrary initial configuration σ_0
  - **Propose** a new one σ_1
  - **Accept or reject** the new one with probability

$$p = \min\{1, \mathcal{P}[\sigma_1]/\mathcal{P}[\sigma_0]\}$$

The Metropolis Algorithm

- Form a (Markov) chain of configurations:
  - **Start** from some arbitrary initial configuration σ_0
  - **Propose** a new one σ_1
  - **Accept or reject** the new one with probability

$$p = \min\{1, \mathcal{P}[\sigma_1]/\mathcal{P}[\sigma_0]\}$$

If **accepted**, σ_1 becomes the new current configuration

If **rejected**, go back to the beginning.

The Metropolis Algorithm

- Form a (Markov) chain of configurations:
  - **Start** from some arbitrary initial configuration σ_0
  - **Propose** a new one σ_1
  - **Accept or reject** the new one with probability

$$p = \min\{1, \mathcal{P}[\sigma_1]/\mathcal{P}[\sigma_0]\}$$

If **accepted**, σ_1 becomes the new current configuration

If **rejected**, go back to the beginning.

$\sigma_0 \quad \sigma_0 \quad \sigma_1 \quad \sigma_2 \quad \ldots$

The Metropolis Algorithm

We have therefore replaced the problem of finding the system in a particular state with the problem of simulating a process where the system moves from one state to another.

The Metropolis Algorithm

We have therefore replaced the problem of finding the system in a particular state with the problem of simulating a process where the system moves from one state to another.

This works because the Metropolis algorithm satisfies two sufficient (though not necessary) conditions:

- **Detailed balance**

  “Moving forwards is as probable as moving backwards”

The Metropolis Algorithm

We have therefore replaced the problem of finding the system in a particular state with the problem of simulating a process where the system moves from one state to another.

This works because the Metropolis algorithm satisfies two sufficient (though not necessary) conditions:

- **Detailed balance**

  "Moving forwards is as probable as moving backwards"

- **Ergodicity**

  "If you wait long enough you will reach every configuration"

Updating strategies

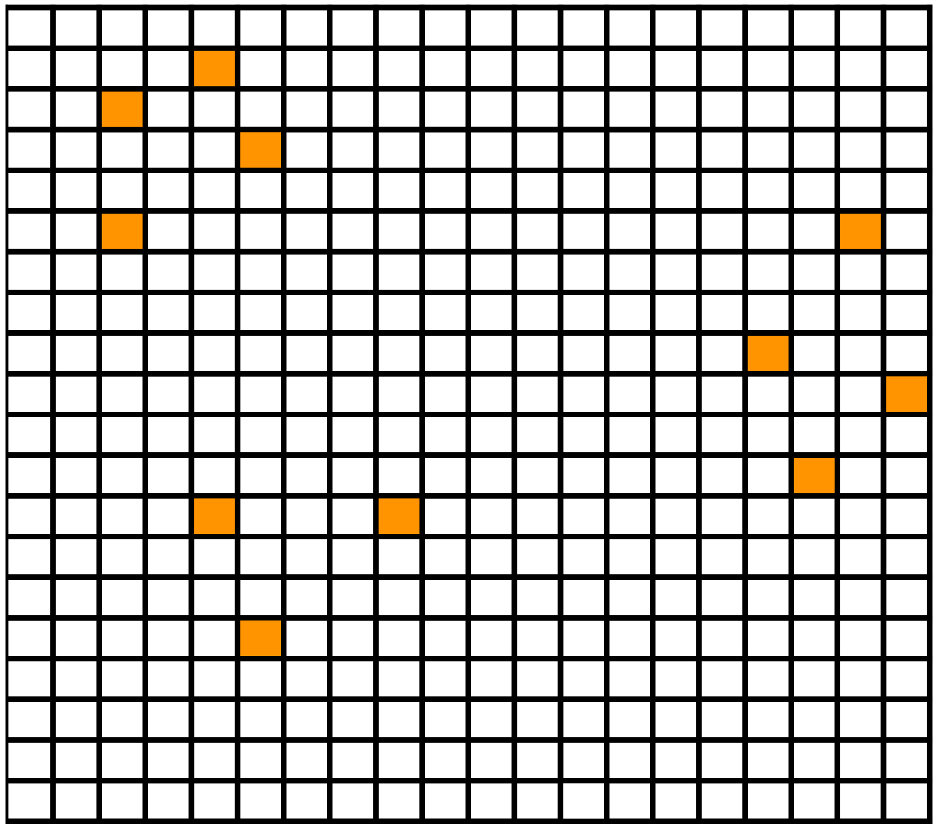
Now, how do we propose a new configuration to the accept/reject step?

There are, in principle, many ways to do this:

- One-by-one (random or ordered)
- Clusters (spread or compact)
- Globally

Updating strategies

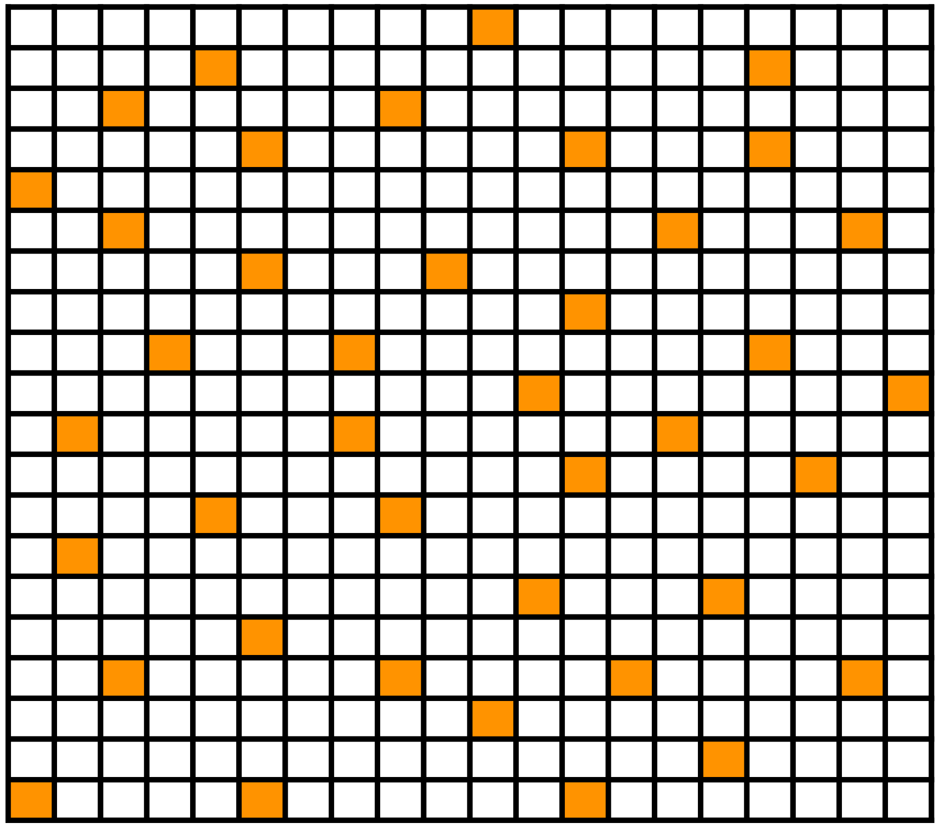
Local updates



Easy to implement

Bad decorrelation properties

Global updates

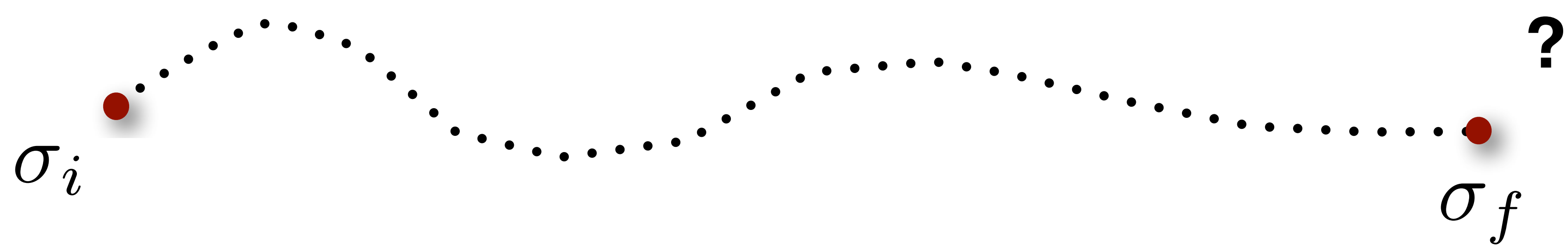


More sophisticated
(harder to implement)

Excellent decorrelation!

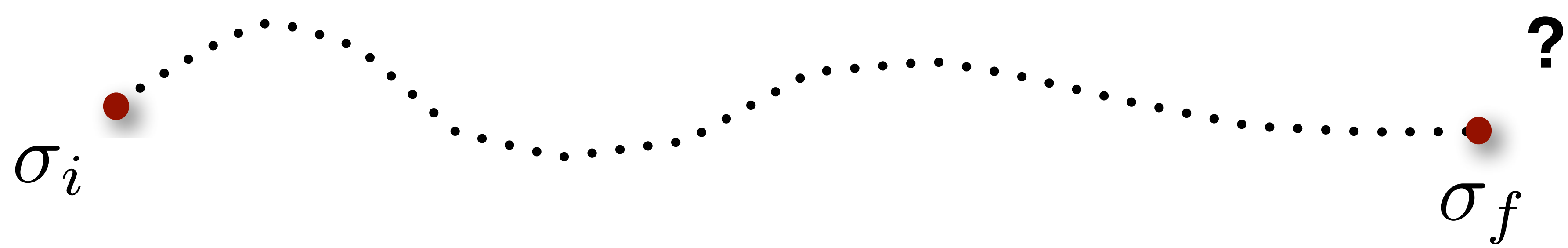
Hybrid Monte Carlo

Problem: How do we change σ_i as much as possible without obtaining an extremely improbable configuration?



Hybrid Monte Carlo

Problem: How do we change σ_i as much as possible without obtaining an extremely improbable configuration?



$$\mathcal{P}[\sigma] = e^{-S_{\text{eff}}}$$

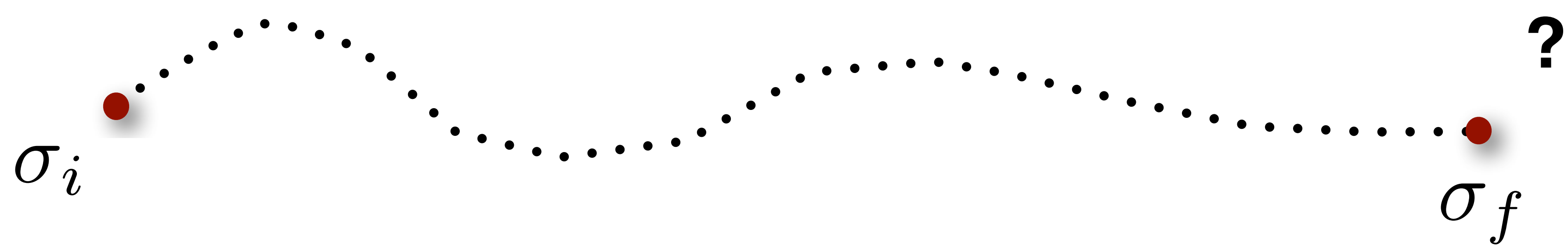
$$\mathcal{Z} = \int \mathcal{D}\sigma e^{-S_{\text{eff}}} \rightarrow \int \mathcal{D}\sigma \mathcal{D}\pi e^{-\mathcal{H}[\sigma,\pi]}$$

Doesn't affect the problem!

$$\mathcal{H}[\sigma, \pi] = \sum_{\mathbf{n},\tau} \pi^2_{\mathbf{n},\tau} + S_{\text{eff}}[\sigma]$$

Hybrid Monte Carlo

Problem: How do we change σ_i as much as possible without obtaining an extremely improbable configuration?



$$\mathcal{P}[\sigma] = e^{-S_{\text{eff}}}$$

$$\mathcal{Z} = \int \mathcal{D}\sigma e^{-S_{\text{eff}}} \to \int \mathcal{D}\sigma \mathcal{D}\pi e^{-\mathcal{H}[\sigma,\pi]}$$

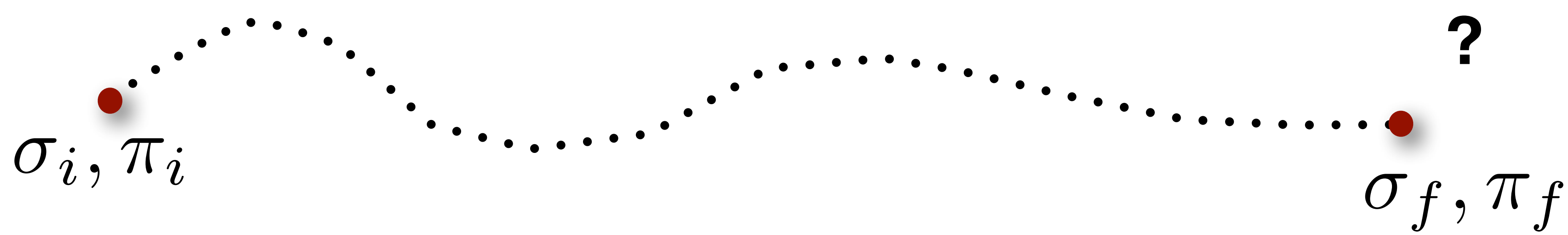
$$\mathcal{H}[\sigma,\pi] = \sum_{\mathbf{n},\tau} \pi^2_{\mathbf{n},\tau} + S_{\text{eff}}[\sigma]$$

Define classical EOM

$$\begin{aligned} \dot{\sigma} &= \frac{\delta \mathcal{H}}{\delta \pi} = \pi \\ \dot{\pi} &= -\frac{\delta \mathcal{H}}{\delta \sigma} = F[\sigma, \varphi] \end{aligned}$$

Hybrid Monte Carlo

Problem: How do we change σ_i as much as possible without obtaining an extremely improbable configuration?



Pick a random gaussian “momentum”

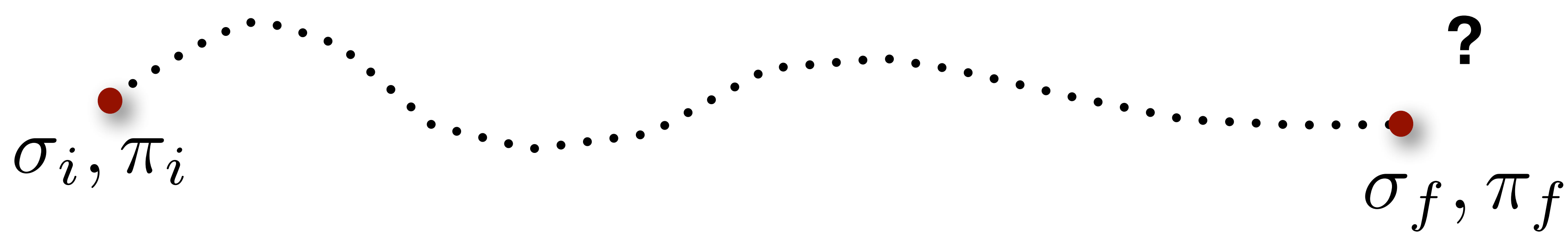
Evolve “classically”

$$
\begin{aligned}
\dot{\sigma} &= \frac{\delta \mathcal{H}}{\delta \pi} = \pi \\
\dot{\pi} &= -\frac{\delta \mathcal{H}}{\delta \sigma} = F[\sigma, \varphi]
\end{aligned}
$$

Use a Metropolis accept/reject step

Hybrid Monte Carlo

Problem: How do we change σ_i as much as possible without obtaining an extremely improbable configuration?



Pick a random gaussian "momentum"

Evolve "classically"

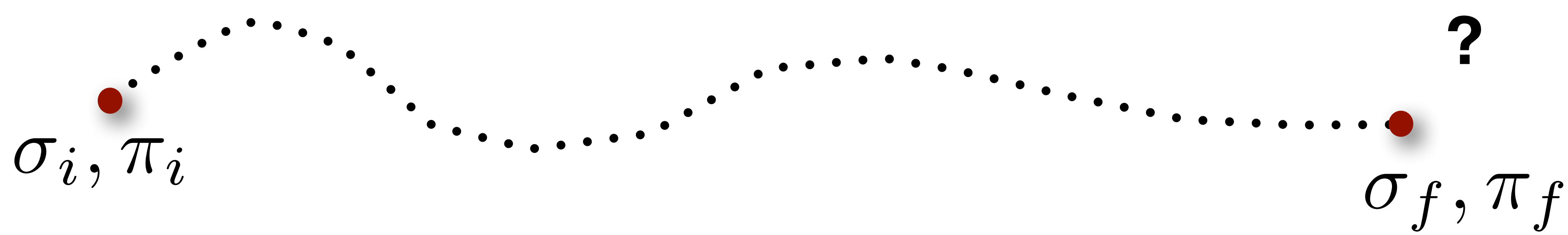
$$\begin{aligned}\dot{\sigma} &= \frac{\delta \mathcal{H}}{\delta \pi} = \pi \\ \dot{\pi} &= -\frac{\delta \mathcal{H}}{\delta \sigma} = F[\sigma, \varphi]\end{aligned}$$

Use a Metropolis accept/reject step

Key points: This classical evolution is global - it touches every point!

Hybrid Monte Carlo

Problem: How do we change σ_i as much as possible without obtaining an extremely improbable configuration?



Pick a random gaussian "momentum"

Evolve "classically"

$$\begin{aligned}\dot{\sigma} &= \frac{\delta \mathcal{H}}{\delta \pi} = \pi \\ \dot{\pi} &= -\frac{\delta \mathcal{H}}{\delta \sigma} = F[\sigma, \varphi]\end{aligned}$$

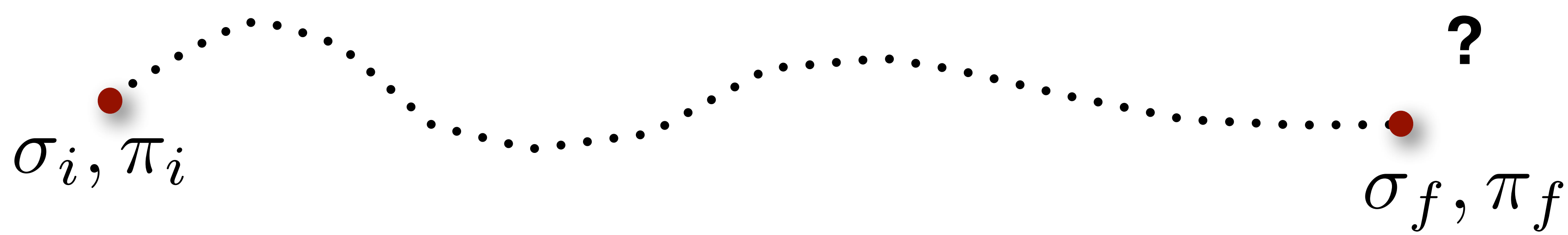
Use a Metropolis accept/reject step

Key points: This classical evolution is global - it touches every point!

Using the accept/reject step ensure we have the right problem

Hybrid Monte Carlo

Problem: How do we change σ_i as much as possible without obtaining an extremely improbable configuration?



Pick a random gaussian "momentum"

Evolve "classically"

$$
\begin{aligned}
\dot{\sigma} &= \frac{\delta \mathcal{H}}{\delta \pi} = \pi \\
\dot{\pi} &= -\frac{\delta \mathcal{H}}{\delta \sigma} = F[\sigma, \varphi]
\end{aligned}
$$

Use a Metropolis accept/reject step

Key points: This classical evolution is global - it touches every point!

Using the accept/reject step ensure we have the right problem

"Energy" is conserved, so acceptance is high!

How efficient are these algorithms?

Determinantal Monte Carlo

- Scaling of computational cost

$$\underbrace{L^3}_{\text{Full basis}} \times \underbrace{N_\tau}_{\text{Time evolution}} \times \underbrace{L^3 \times \log L^3}_{\text{FFT}} \times \underbrace{L^3 \times N_\tau}_{\text{Full update (sweep) via local updates}}$$

It doesn't matter how big your computer is...

A more efficient way

- Scaling of computational cost

$$\underbrace{L^3}_{\text{Full basis}} \times \underbrace{N_\tau}_{\text{Time evolution}} \times \underbrace{L^3 \times \log L^3}_{\text{FFT}} \times \underbrace{\cancel{L^3 \times N_\tau}}_{\substack{\text{Full update (sweep)} \\ \text{via local updates}}}$$

Use **Hybrid Monte Carlo**!

Global updates of the auxiliary field via Molecular Dynamics

A more efficient way

- Scaling of computational cost

$$\cancel{L^3} \times N_\tau \times \underline{L^3 \times \log L^3} \times \underline{\cancel{L^3 \times N_\tau}}$$

Full basis

Time evolution

FFT

Full update (sweep)
via local updates

Use **pseudofermions**!

Stochastic calculation
of the determinant

Use **Hybrid Monte Carlo**!

Global updates of the
auxiliary field via
Molecular Dynamics

A more efficient way

- Scaling of computational cost

$$\cancel{L^3} \times N_\tau \times \underline{L^3 \times \log L^3} \times \underline{\cancel{L^3 \times N_\tau}}$$

Full basis

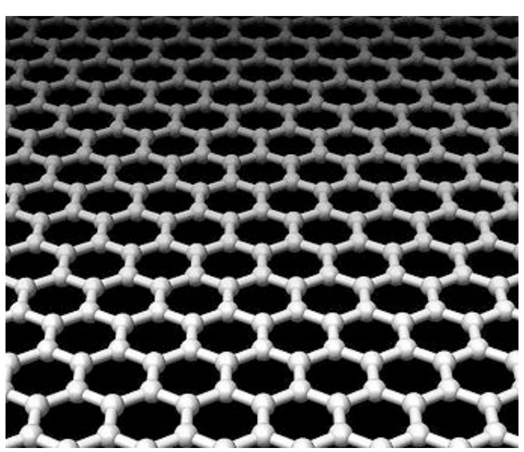
Time evolution

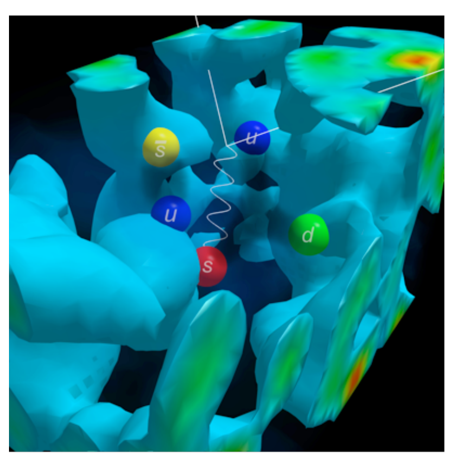
FFT

Full update (sweep)
via local updates

Use **pseudofermions**!

Use **Hybrid Monte Carlo**!



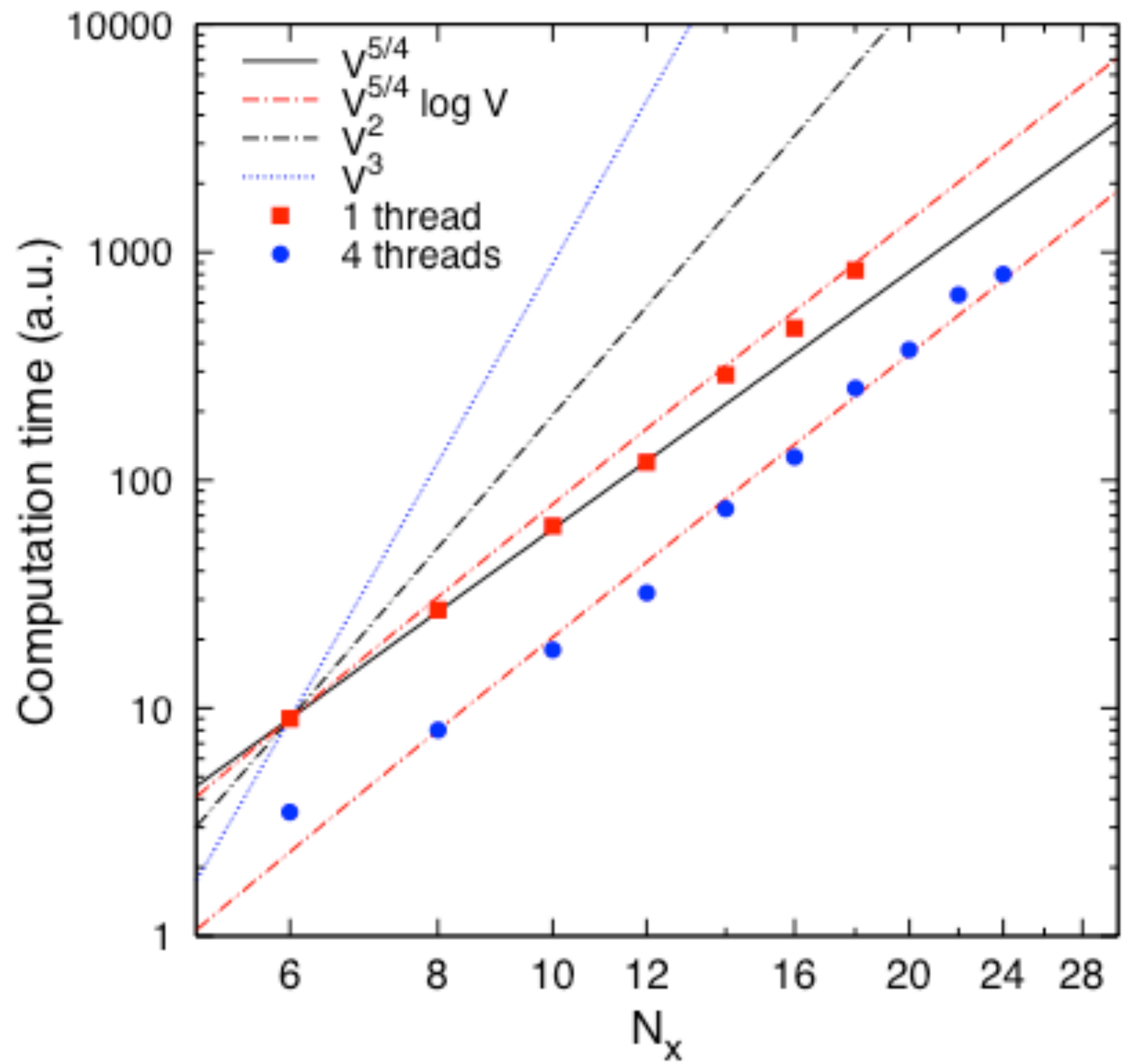


This is how state-of-the-art lattice QCD is done! (and graphene!)

Drut & Lähde, Phys. Rev. Lett. **102**, 026802 (2009)

Scaling tests

- Determinantal MC ~ V^3
- Determinantal MC with "worm" updates ~ V^2
- Polynomial HMC ~ $V^{5/4}$



A "new" algorithm

- Scaling of computational cost

$$L^3 \times N_\tau \times \underline{L^3 \times \log L^3} \times \underline{\cancel{L^3 \times N_\tau}}$$

Full basis

Time evolution

FFT

Full update (sweep)
via local updates

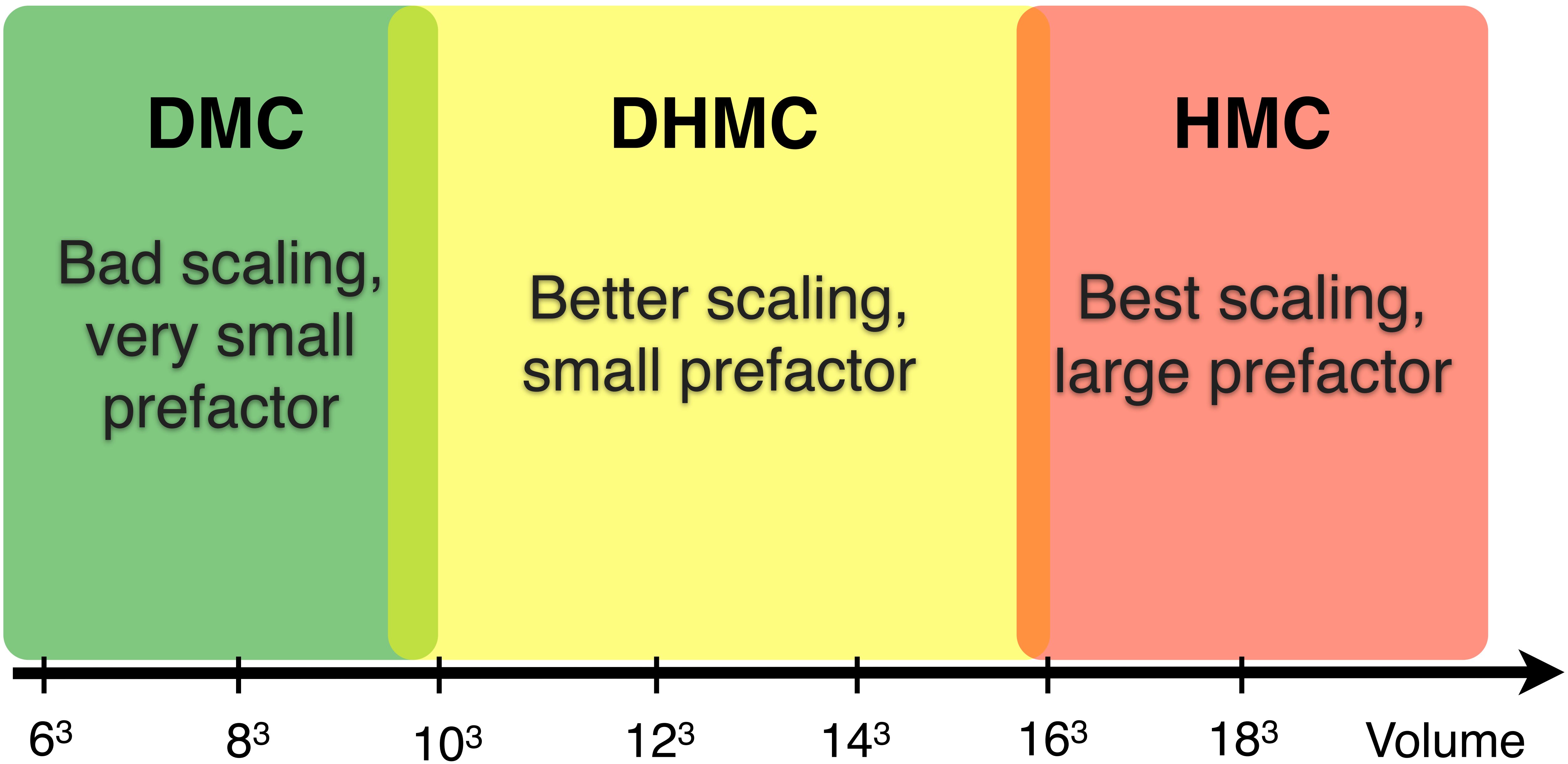
Still compute the
determinant exactly!

Use **Hybrid Monte Carlo**!

Global updates of the
auxiliary field via
Molecular Dynamics

In practice...

... prefactors matter!



Summary

- Calculating simple observables requires inverting matrices.

- Calculating more complicated observables requires inverting matrices more than once, which is numerically challenging.
  Statistically, this amounts to asking for (complicated) moments of the quantum probability distribution, which will require many samples, i.e. these observables are typically noisy!

- Sampling techniques range from the simple (local Metropolis) to the extremely advanced (sophisticated forms of HMC)

- Scaling is important. Pre-factors matter.