Introduction to Bosonic QMC: World line and Stochastic Green Function algorithms

G. G. Batrouni (Nice, Singapore, Beijing)

Outline

- Brief review of classical Monte Carlo
- World line algorithm
- Introduction to work-like algorithms

Preliminary Comments

- There is no “best” algorithm.
- Algorithms have advantages and disadvantages.
- Algorithms can be very well adapted to and optimized for a class of problems, but not work very well for other classes.
- The point is to choose an algorithm which is well adapted to your problem.
- Be pragmatic.

Here I will try to explain the basics building blocks of algorithms, but will not discuss important issues like auto-correlation, error bars etc. These are very interesting and important issues, but beyond our scope here.

Classical Monte Carlo

Our main goal is to calculate averages:

$$\langle \mathcal{A}[\{\phi\}] \rangle = \int D\phi \; \mathcal{A}[\{\phi\}] \, P[\{\phi\}]; \qquad P[\{\phi\}] = \frac{\mathrm{e}^{-\beta E[\{\phi\}]}}{Z}$$

P is the normalized (equilibrium) Boltzmann weight and Z the partition function. Complete numerical integration is impossible for large systems, even for a “simple” model like Ising where the number of configurations grows with the number of sites as 2^N.

Monte Carlo: Construct a Markov chain to generate the configurations such that in the stationary limit, configurations are given by the correct probability distribution: *P.* In a Markov chain, an event depends only on the event immediately before it.

Classical Monte Carlo

The evolution equation for *P* can be written as:

$$\frac{\mathrm{d}P[\phi,t]}{\mathrm{d}t} = \underbrace{\sum_{\{\phi'\}} P[\phi',t]W_{\phi'\to\phi}}_{\text{Rate of populating the configuration }\{\varphi\}} - \underbrace{\sum_{\{\phi'\}} P[\phi,t]W_{\phi\to\phi'}}_{\text{Rate of depopulating }\{\varphi\}}$$

- *P*[φ,t] is the probability density for the configuration {φ} at step t.
- $W_{\phi\to\phi'}$ is the transition rate from φ to φ'
- t is not necessarily real time.

In the stationary limit (**i.e. at equilibrium**):

$$\sum_{\{\phi'\}} P[\phi']W_{\phi'\to\phi} = \sum_{\{\phi'\}} P[\phi]W_{\phi\to\phi'}$$

Not easy to implement!

Classical Monte Carlo

Easier to implement the configuration-by-configuration condition:

$$P[\{\phi'\}]W_{\phi'\to\phi} = P[\{\phi\}]W_{\phi\to\phi'}$$

This condition is called ***detailed balance***, it is sufficient to reach the correct equilibrium, but ***not*** necessary!

Many ways to do this. I give two examples:

Glauber dynamics:

$$W_{\phi\to\phi'} = \frac{P[\{\phi'\}]}{P[\{\phi'\}] + P[\{\phi\}]}$$

Classical Monte Carlo

Check detailed balance:

$$P[\{\phi'\}]W_{\phi'\to\phi} = P[\{\phi\}]W_{\phi\to\phi'}$$

$$P[\{\phi'\}]W_{\phi'\to\phi} = \frac{P[\{\phi'\}]P[\{\phi\}]}{P[\{\phi\}] + P[\{\phi'\}]}$$

$$P[\{\phi\}]W_{\phi\to\phi'} = \frac{P[\{\phi\}]P[\{\phi'\}]}{P[\{\phi'\}] + P[\{\phi\}]}$$

We see that detailed balance is indeed satisfied. Of course we have:

$$P[\{\phi\}] = \frac{1}{Z}\mathrm{e}^{-\beta H[\{\phi\}]}$$

Z cancels out in the detailed balance condition: We do not need to know the partition function, we only need the exponential of the energy of the configuration. We have this!

Classical Monte Carlo

Another very popular choice is Metropolis dynamics:

$$\begin{aligned} W_{\phi_i \to \phi_f} &= 1, & \text{if } P[\{\phi_f\}] \geq P[\{\phi_i\}] \\ W_{\phi_i \to \phi_f} &= \frac{P[\{\phi_f\}]}{P[\{\phi_i\}]}, & \text{if } P[\{\phi_f\}] < P[\{\phi_i\}] \end{aligned}$$

Check detailed balance for transitions between two configurations φ and φ'. Two possibilities:

$P[\{\phi'\}] > P[\{\phi\}]$ or $P[\{\phi'\}] < P[\{\phi\}]$ Take the first one.

$$P[\{\phi\}] W_{\phi \to \phi'} = P[\{\phi\}]$$

$$P[\{\phi'\}] W_{\phi' \to \phi} = P[\{\phi'\}\} \frac{P[\{\phi\}]}{P[\{\phi'\}]} = P[\{\phi\}]$$

As before, we do not need Z. Choose the dynamics you want, iterate using the appropriate *W.*

Classical Monte Carlo

Example: Ising with Metropolis.

$$H[\{S\}] = -J\sum_{\langle i,j\rangle} S_i S_j, \quad S_\ell = \pm 1, \quad Z = \sum_{\{s\}} \mathrm{e}^{-\beta H[\{S\}]}$$

- Generate an arbitrary initial configuration of spins: $\{S\}_{\mathrm{init}}$
- Visit the spins in any order (sequentially or randomly).
- At each spin, try to change it: $S_i \to -S_i$
- Calculate the ratio: $\mathcal{R} = P[\{S'\}]/P[\{S\}]$
- $\mathcal{R} = \dfrac{\mathrm{e}^{-\beta J S_i \sum_j' S_j}}{\mathrm{e}^{\beta J S_i \sum_j' S_j}} = \mathrm{e}^{-2\beta J S_i \sum_j' S_j}$
- Accept the new value of the spin and move to the next one if: $\mathcal{R} \geq 1.$

Classical Monte Carlo

- Otherwise, generate a random number, *r*, uniformly distributed between 0 and 1. Accept the new value of the spin if: $r < \mathcal{R}$
- Otherwise reject the attempted change: Put back the value of the spin to its original value.
- Visit the next spin and repeat.
- When you visit **all** the spins, you have completed a *SWEEP*.
- Start another sweep. And so on.
- After many sweeps, the system will be at equilibrium and the spin configurations given by the correct Boltzmann weight.
- Continue generating new configurations now to calculate averages.
- To calculate physical quantities, visit the configuration and calculate.
- Average over all measurements. The average is arithmetic because the measurements have the correct weight built in.

Classical Monte Carlo

Important considerations:

- We need to make sure the system has thermalized, in other words P is stationary. Otherwise, the configurations are not drawn from equilibrium.
- The update scheme I described is local: In the Ising example I showed, we saw that only the close neighbors of a given spin participate in its evolution
- Away from a critical point, this is fine and evolution is efficient because correlation the length is short.
- But near a critical point, the correlation length is very long and one needs many more iterations for the effect of a flipped spin to be felt by all the spins correlated with it. This is called **critical slowing down**.
- In some cases it is possible to fight it by constructing “cluster algorithms”, e.g. the Wolff algorithm.

Quantum Monte Carlo

Apply the Metropolis method to quantum systems:

$$Z = \mathrm{Tr}\ \mathrm{e}^{-\beta\hat{H}}$$

But the partition function here is a trace over an operator: How can we apply Metropolis to something like this? Recall that for Metropolis, we changed c-numbers and probabilities; how can one “change operators”?

We change matrix elements of operators!

The trace of an operator is invariant, we can calculate it in any representation, |ψ>:

$$\begin{aligned} Z &= \sum \langle \Psi_1 | \mathrm{e}^{-\beta H} | \Psi_1 \rangle \\ &= \sum \langle \Psi_1 | \underbrace{\mathrm{e}^{-\Delta\tau H} \mathrm{e}^{-\Delta\tau H} \mathrm{e}^{-\Delta\tau H} \ldots \mathrm{e}^{-\Delta\tau H}}_{\text{L exponentials: } \beta = L\Delta\tau} | \Psi_1 \rangle \end{aligned}$$

L is chosen such that Δτ times the largest energy scale in *H* is very small.

Quantum Monte Carlo

So:

$$
\begin{aligned}
Z &= \sum \langle \Psi_1 | \mathrm{e}^{-\Delta\tau H} \mathrm{e}^{-\Delta\tau H} \mathrm{e}^{-\Delta\tau H} \ldots \mathrm{e}^{-\Delta\tau H} | \Psi_1 \rangle \\
&= \sum \langle \Psi_1 | \mathrm{e}^{-\Delta\tau H} | \Psi_L \rangle \langle \Psi_L | \mathrm{e}^{-\Delta\tau H} | \Psi_{L-1} \rangle \langle \Psi_{L-1} | \mathrm{e}^{-\Delta\tau H} | \Psi_{L-2} \rangle \langle \Psi_{L-2} | \ldots \\
&\quad \ldots\ldots | \Psi_2 \rangle \langle \Psi_2 | \mathrm{e}^{-\Delta\tau H} | \Psi_1 \rangle
\end{aligned}
$$

Used: $1 = \sum |\Psi\rangle\langle\Psi|$

- The partition function is now a sum over a product of matrix elements, i.e. c-numbers!
- Time evolution operator: e^{-iHt}
- Matrix elements formally look like (imaginary) time evolution between the two successive states: $|\Psi_n\rangle \to |\Psi_{n+1}\rangle$
- If the product of the matrix elements is positive, it can be used as a Boltzmann weight and the Metropolis algorithm can be used.
- If the product is negative or, worse, complex, we are in trouble!
- Sometimes, we can avoid this with a good choice of states. But **NOT ALWAYS!**

Quantum Monte Carlo: World Line Algorithm

Refreneces:

- ``Monte Carlo simulations of one-dimensional fermion systems'', J. E. Hirsch, R. L. Sugar, D. J. Scalapino, and R. Blankenbecler Phys. Rev. B26 (1982) 5033.
- ``World-line quantum Monte Carlo algorithm for a one-dimensional Bose model'', G. G. Batrouni et R. T. Scalettar, Phys. Rev. B46 (1992) 9051.

This is a relatively easy algorithm to understand intuitively and to implement in code. You can easily calculate all diagonal quantities (we will see what this means) but not non-diagonal ones, such as Green functions.

A large class of Hamiltonians can be written in the form:

$$H = \mathsf{K} + \mathsf{U}$$

Quantum Monte Carlo: World Line Algorithm

Where K is nondiagonal, for example:

$$\mathsf{K} = -t \sum_{\langle ij \rangle} (\hat{a}_i^\dagger \hat{a}_j + \hat{a}_j^\dagger \hat{a}_i)$$

And U is diagonal, for example:

$$\mathsf{U} = V_0 \sum_i \hat{n}_i (\hat{n}_i - 1) + V_1 \sum_{\langle ij \rangle} \hat{n}_i \hat{n}_j$$

$$\hat{n}_i = \hat{a}_i^\dagger \hat{a}_i \,, \qquad [\hat{a}_i \,, \hat{a}_j^\dagger] = \delta_{i,j}$$

This type of model will be our focus.

Quantum Monte Carlo: World Line Algorithm

To fix ideas, we will take the one-dimensional Bose-Hubbard model:

$$H = \underbrace{-t\sum_i (\hat{a}_i^\dagger \hat{a}_{i+1} + \hat{a}_{i+1}^\dagger \hat{a}_i)}_{\mathsf{K}} + \underbrace{V_0 \sum_i \hat{n}_i(\hat{n}_i - 1) + V_1 \sum_i \hat{n}_i \hat{n}_{i+1} + \ldots}_{\mathsf{U}}$$

Write:

$$\left.\begin{aligned} H_1 &= -t \sum_{i\,\mathrm{odd}} (\mathsf{a}_i^\dagger \mathsf{a}_{i+1} + \mathsf{a}_{i+1}^\dagger \mathsf{a}_i) + \frac{1}{2}\mathsf{U} \\ H_2 &= -t \sum_{i\,\mathrm{even}} (\mathsf{a}_i^\dagger \mathsf{a}_{i+1} + \mathsf{a}_{i+1}^\dagger \mathsf{a}_i) + \frac{1}{2}\mathsf{U} \end{aligned}\right\} \quad H = H_1 + H_2$$

H_1 connects odd site to even site after it, and H_2 connects even site to odd site after it. This allows us to write the first approximation:

$$\mathrm{e}^{-\Delta\tau H} \approx \mathrm{e}^{-\Delta\tau H_2}\mathrm{e}^{-\Delta\tau H_1} + \mathcal{O}\{(\Delta\tau)^2[H_1, H_2]\}$$

where we used the Baker-Hausdorff formula

$$\mathrm{e}^{\mathsf{A}}\mathrm{e}^{\mathsf{B}} = \mathrm{e}^{\mathsf{A}+\mathsf{B}+\frac{1}{2}[\mathsf{A},\mathsf{B}]+\ldots}$$

(Checkerboard decomposition)

Quantum Monte Carlo: World Line Algorithm

Then, the partition function becomes:

$$\begin{aligned} Z &= \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-\Delta\tau H} e^{-\Delta\tau H} e^{-\Delta\tau H} \ldots e^{-\Delta\tau H} e^{-\Delta\tau H} \quad L_\tau \text{ exponentials} \\ &\approx \mathrm{Tr}\, e^{-\Delta\tau H_2} e^{-\Delta\tau H_1} e^{-\Delta\tau H_2} e^{-\Delta\tau H_1} \ldots \\ &\quad \ldots e^{-\Delta\tau H_2} e^{-\Delta\tau H_1} \ldots e^{-\Delta\tau H_2} e^{-\Delta\tau H_1} \quad 2L_\tau \text{ exponentials} \end{aligned}$$

Now consider the occupation number states:

$$\begin{aligned} a_i^\dagger |n_1, n_2, \ldots, n_i, \ldots, n_S\rangle &= \sqrt{n_i + 1}\, |n_1, n_2, \ldots, n_i + 1, \ldots, n_S\rangle \\ a_i |n_1, n_2, \ldots, n_i, \ldots, n_S\rangle &= \sqrt{n_i}\, |n_1, n_2, \ldots, n_i - 1, \ldots, n_S\rangle \\ n_i |n_1, n_2, \ldots, n_i, \ldots, n_S\rangle &= n_i\, |n_1, n_2, \ldots, n_i, \ldots, n_S\rangle \end{aligned}$$

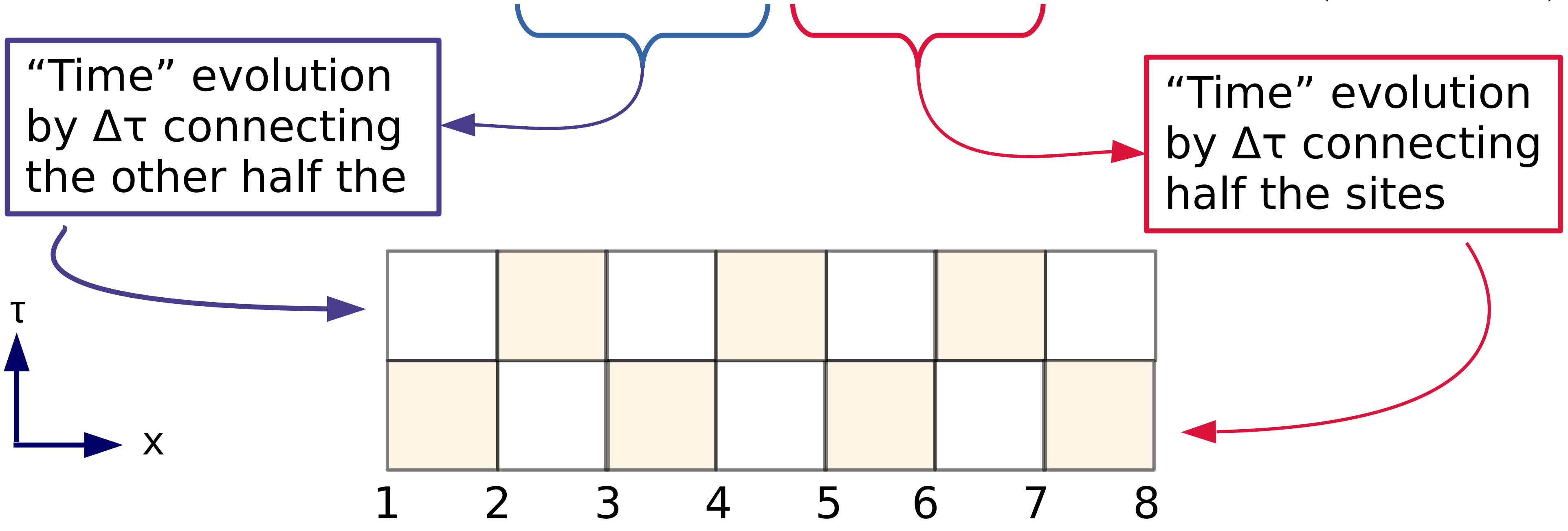
The subscripts label the sites on the 1-dimensional lattice. To evaluate the trace, introduce between each pair of exponentials the identity:

$$I = \sum_{\{n\}} |\mathbf{n}\rangle\langle\mathbf{n}| = \sum_{n_1, n_2, \ldots, n_i \ldots n_S} |n_1, n_2, \ldots, n_i, \ldots, n_S\rangle\langle n_1, n_2, \ldots, n_i, \ldots, n_S|$$

Quantum Monte Carlo: World Line Algorithm

$$Z = \sum_{\{\mathbf{n}\}} \langle \mathbf{n}^1 | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2L_\tau} \rangle \langle \mathbf{n}^{2L_\tau} | \mathrm{e}^{-\Delta\tau H_1} | \mathbf{n}^{2L_\tau - 1} \rangle \ldots$$

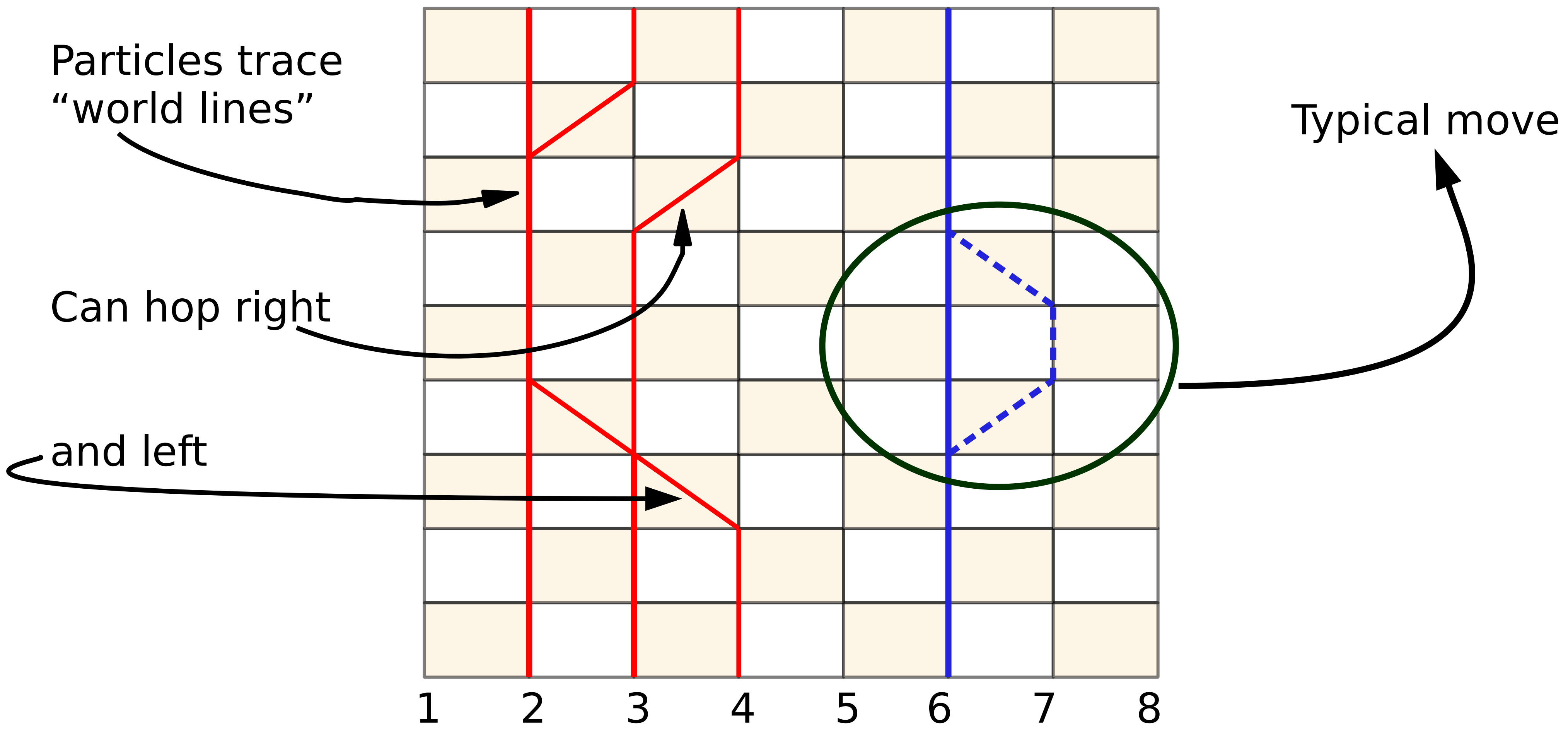
$$\ldots \underbrace{\langle \mathbf{n}^3 | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^2 \rangle} \underbrace{\langle \mathbf{n}^2 | \mathrm{e}^{-\Delta\tau H_1} | \mathbf{n}^1 \rangle} \qquad (\beta = \Delta\tau L_\tau)$$


"Time" evolution by Δτ connecting the other half the
"Time" evolution by Δτ connecting half the sites
τ
x
1
2
3
4
5
6
7
8

Sites on shaded squares are connected.

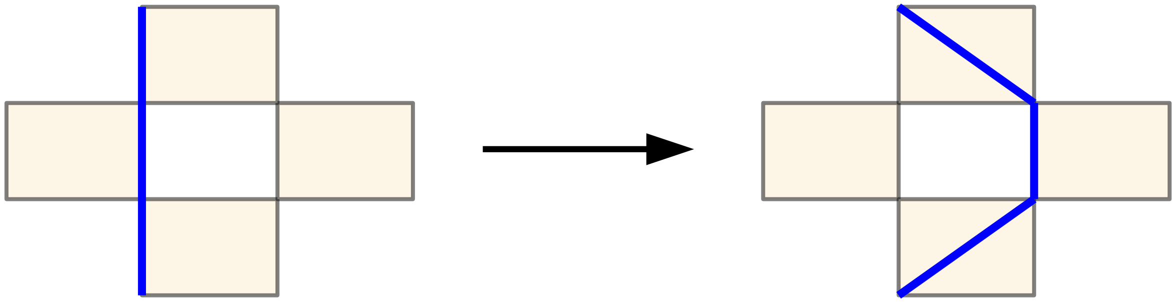
Two consecutive matrix elements advance the entire system by one imaginary time step: All sites will be connected.

Quantum Monte Carlo: World Line Algorithm



Quantum Monte Carlo: World Line Algorithm

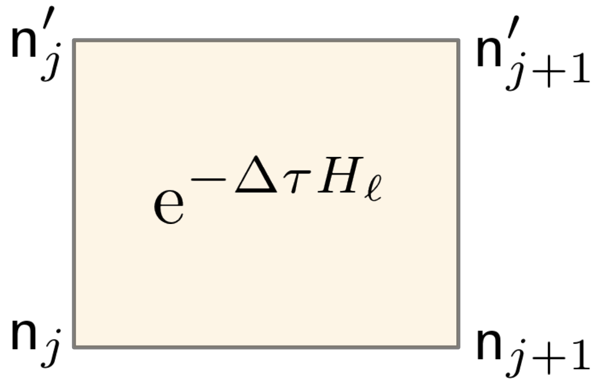
Consider a typical move:



In a typical move, only four matrix elements change. **They can be evaluated independently**. That is why we broke up the Hamiltonian as we did.

- Visit a shaded square.
- Randomly choose: Try a left jump or a right jump.
- Check if the choice is possible.
- Not possible: move to next square.
- Possible: Evaluate the product of the 4 elements before the move.
- Then: Evaluate the product of the 4 elements after the move.
- Accept/reject using the Metropolis criterion.
- Move to next shaded square.
- And so on.
- The problem is now reduced to a product of independent 4-site matrix elements which can be evaluated exactly.

Quantum Monte Carlo: World Line Algorithm



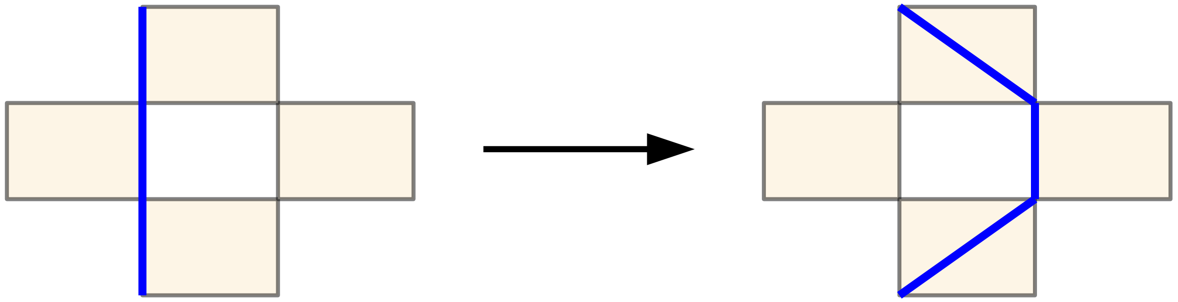
$$\begin{aligned}
\langle n'_j, n'_{j+1} | e^{-\Delta\tau H_\ell} | n_j, n_{j+1} \rangle &= \langle n'_j, n'_{j+1} | e^{-\Delta\tau (K_\ell + \frac{1}{2} U_\ell)} | n_j, n_{j+1} \rangle \\
&\approx \langle n'_j, n'_{j+1} | e^{-\Delta\tau \frac{1}{4} U_\ell} e^{-\Delta\tau K_\ell} e^{-\Delta\tau \frac{1}{4} U_\ell} | n_j, n_{j+1} \rangle \\
&\approx e^{-\Delta\tau U'/4} e^{-\Delta\tau U/4} \langle n'_j, n'_{j+1} | e^{-\Delta\tau K_\ell} | n_j, n_{j+1} \rangle
\end{aligned}$$

$$U = \frac{V_0}{2} \left[n_j (n_j - 1) + n_{j+1} (n_{j+1} - 1) \right]$$

The entire update sweep is comprised of evaluations of such matrix elements.

Quantum Monte Carlo: World Line Algorithm

We consider the typical move:



$$
\begin{aligned}
\Pi_i &= \langle 0,0|e^{-\Delta\tau H_1}|0,0\rangle\langle 1,0|e^{-\Delta\tau H_2}|1,0\rangle \\
&\quad \langle 0,1|e^{-\Delta\tau H_1}|0,1\rangle\langle 1,0|e^{-\Delta\tau H_2}|1,0\rangle \\
\\
&= \langle 0,0|e^{-\Delta\tau \mathsf{K}_1}|0,0\rangle\langle 1,0|e^{-\Delta\tau \mathsf{K}_2}|1,0\rangle \\
&\quad \langle 0,1|e^{-\Delta\tau \mathsf{K}_1}|0,1\rangle\langle 1,0|e^{-\Delta\tau \mathsf{K}_2}|1,0\rangle
\end{aligned}
$$

Quantum Monte Carlo: World Line Algorithm

The matrix elements can be evaluated exactly numerically. But we can also expand the exponential:

$$\begin{aligned}
\langle 0,1|\mathrm{e}^{-\Delta\tau \mathsf{K}_i}|0,1\rangle &\approx \langle 0,1|(1-\Delta\tau\mathsf{K}_i)|0,1\rangle \\
&= 1-\Delta\tau\langle 0,1|(a_i^\dagger a_{i+1}+a_{i+1}^\dagger a_i)|0,1\rangle = 1 \\
\langle 0,1|\mathrm{e}^{-\Delta\tau \mathsf{K}_i}|1,0\rangle &\approx \langle 0,1|(1-\Delta\tau\mathsf{K}_i)|1,0\rangle \\
&= \langle 0,1|1,0\rangle + t\Delta\tau\langle 0,1|(a_i^\dagger a_{i+1}+a_{i+1}^\dagger a_i)|1,0\rangle \\
&= t\Delta\tau
\end{aligned}$$

So, we have for our typical move:

$$\Pi_i = 1 \qquad\qquad \Pi_f = (t\Delta\tau)^2$$

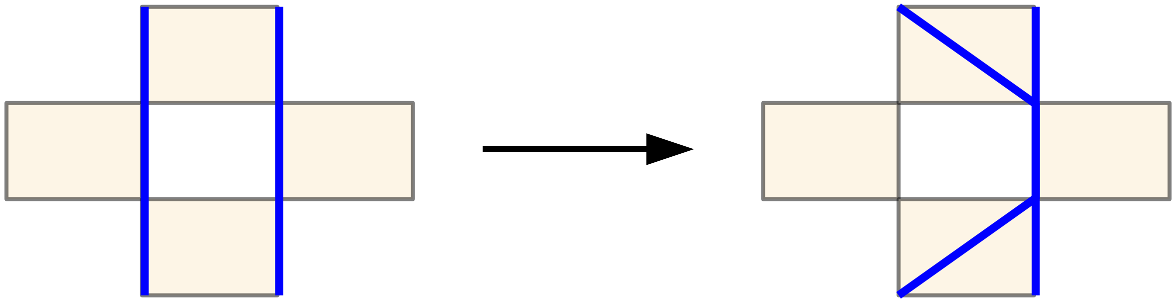
$$\frac{\Pi_f}{\Pi_i} = (t\Delta\tau)^2 < 1$$

Generate random number from uniform distribution [0,1)

Accept the new configuration if: $r \leq \dfrac{\Pi_f}{\Pi_i}$

Quantum Monte Carlo: World Line Algorithm

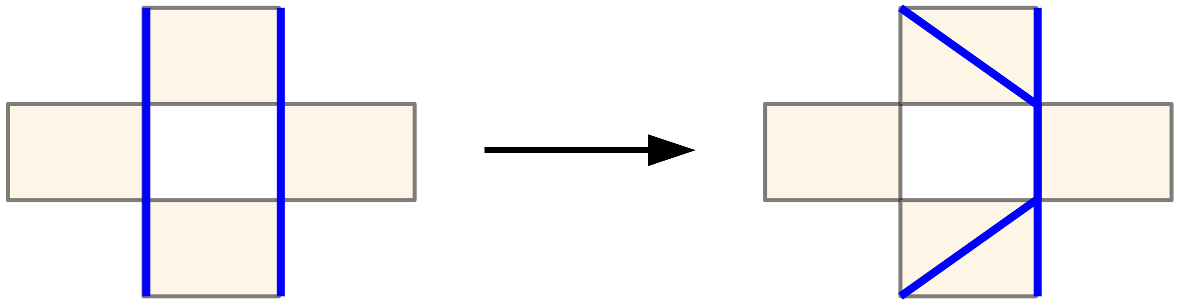
Another example:



$$
\begin{aligned}
\Pi_i &= \langle 1,0|\mathrm{e}^{-\Delta\tau H_1}|1,0\rangle\langle 1,1|\mathrm{e}^{-\Delta\tau H_2}|1,1\rangle \\
&\quad \langle 0,1|\mathrm{e}^{-\Delta\tau H_1}|0,1\rangle\langle 1,1|\mathrm{e}^{-\Delta\tau H_2}|1,1\rangle \\
&= \langle 1,0|\mathrm{e}^{-\Delta\tau \mathsf{K}_1}|1,0\rangle\langle 1,1|\mathrm{e}^{-\Delta\tau \mathsf{K}_2}|1,1\rangle \\
&\quad \langle 0,1|\mathrm{e}^{-\Delta\tau \mathsf{K}_1}|0,1\rangle\langle 1,1|\mathrm{e}^{-\Delta\tau \mathsf{K}_2}|1,1\rangle \\
&= 1 \quad \text{(Calculated exactly as before)}
\end{aligned}
$$

Quantum Monte Carlo: World Line Algorithm

Another example:



$$\begin{aligned}
\Pi_f &= \langle 2,0|\mathrm{e}^{-\Delta\tau H_1}|2,0\rangle\langle 1,1|\mathrm{e}^{-\Delta\tau H_2}|0,2\rangle \\
&\quad \langle 0,0|\mathrm{e}^{-\Delta\tau H_1}|0,0\rangle\langle 0,2|\mathrm{e}^{-\Delta\tau H_2}|1,1\rangle \\
&= \mathrm{e}^{-\frac{\Delta\tau V_0}{4}}\mathrm{e}^{-\frac{\Delta\tau V_0}{4}}\langle 2,0|\mathrm{e}^{-\Delta\tau \mathsf{K}_1}|2,0\rangle\langle 1,1|\mathrm{e}^{-\Delta\tau \mathsf{K}_2}|0,2\rangle\mathrm{e}^{-\frac{\Delta\tau V_0}{4}} \\
&\quad \langle 0,0|\mathrm{e}^{-\Delta\tau \mathsf{K}_1}|0,0\rangle\langle 0,2|\mathrm{e}^{-\Delta\tau \mathsf{K}_2}|1,1\rangle\mathrm{e}^{-\frac{\Delta\tau V_0}{4}} \\
&= \mathrm{e}^{-\Delta\tau V_0}\langle 2,0|\mathrm{e}^{-\Delta\tau \mathsf{K}_1}|2,0\rangle\langle 1,1|\mathrm{e}^{-\Delta\tau \mathsf{K}_2}|0,2\rangle \\
&\quad \langle 0,0|\mathrm{e}^{-\Delta\tau \mathsf{K}_1}|0,0\rangle\langle 0,2|\mathrm{e}^{-\Delta\tau \mathsf{K}_2}|1,1\rangle
\end{aligned}$$

Quantum Monte Carlo:
World Line Algorithm

The matrix elements can be evaluated as before:

$$\begin{aligned}\langle 2,0|\mathrm{e}^{-\Delta\tau \mathsf{K}_i}|1,1\rangle &\approx \langle 2,0|(1-\Delta\tau\mathsf{K}_i)|1,1\rangle \\ &= \langle 2,0|1,1\rangle + t\Delta\tau\langle 2,0|(a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i)|1,1\rangle \\ &= t\Delta\tau\langle 2,0|a_i^\dagger a_{i+1}|1,1\rangle \\ &= t\sqrt{2}\Delta\tau\end{aligned}$$

So, we have for our typical move:

$\Pi_i = 1$ $\qquad\qquad$ $\Pi_f = 2\mathrm{e}^{-\Delta\tau U}(t\Delta\tau)^2$

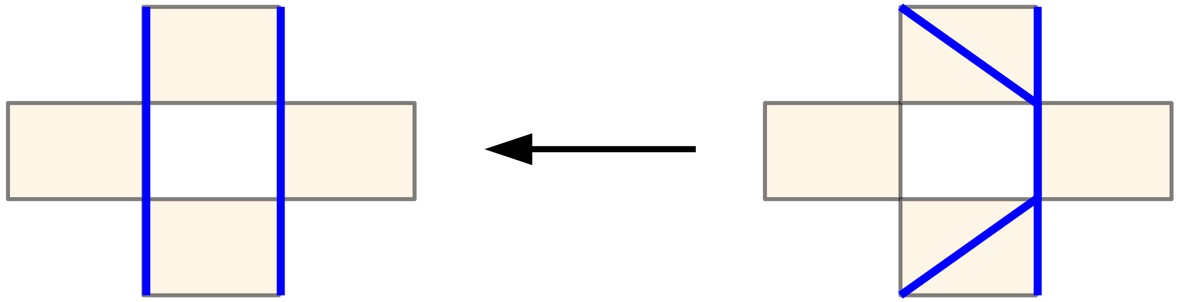
$$\frac{\Pi_f}{\Pi_i} = 2\mathrm{e}^{-\Delta\tau U}(t\Delta\tau)^2 < 1$$

Generate random number from uniform distribution [0,1)

Accept the new configuration if: $r \leq \dfrac{\Pi_f}{\Pi_i}$

Quantum Monte Carlo: World Line Algorithm

On the other hand:



Then:

$$\Pi_i = 2e^{-\Delta\tau U}(t\Delta\tau)^2 \qquad \Pi_f = 1$$

$$\frac{\Pi_f}{\Pi_i} = \left[2e^{-\Delta\tau U}(t\Delta\tau)^2\right]^{-1} > 1$$

Accept!

So, we now know how to make the Monte Carlo moves. We do many sweeps until the system thermalizes. Then we need to measure physical quantities and average them.

Quantum Monte Carlo: World Line Algorithm

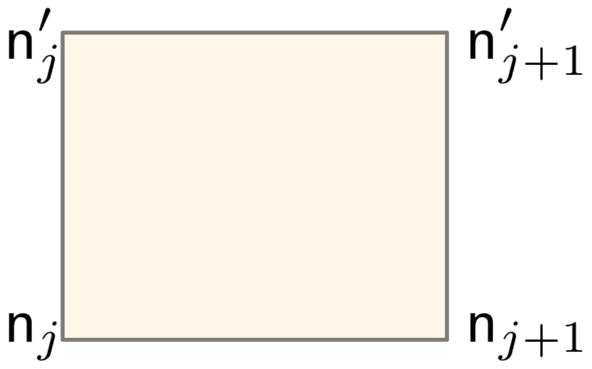
Why split H into H_1 and H_2? Recall, in my examples, I used expressions like: $e^{-\Delta\tau K_i} \approx 1 - \Delta\tau(a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i)$

to calculate the matrix elements of the 4-site problem. But in fact, for this 4-site problem, the matrix element of the **exponential** is calculated exactly:

- Calculate the matrix elements of K, assuming a maximum multiple occupancy on a site (say 6),
- Fourier transform (diagonalize),
- Exponentiate.
- Invert Fourier transform.

But to see how quickly this becomes complicated, go to second order:

$$\begin{aligned} e^{-\Delta\tau K_i} \quad \approx \quad & 1 - \Delta\tau(a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i) \\ & + \frac{(\Delta\tau)^2}{2}\left(a_i^\dagger a_{i+1} a_i^\dagger a_{i+1} + a_i^\dagger a_{i+1} a_{i+1}^\dagger a_i\right. \\ & \left. + a_{i+1}^\dagger a_i a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i a_{i+1}^\dagger a_i\right) \end{aligned}$$



Quantum Monte Carlo: World Line Algorithm

The term: $a_i^\dagger a_{i+1} a_i^\dagger a_{i+1}$ destroys 2 particles on *i+1* and creates 2 on *i*

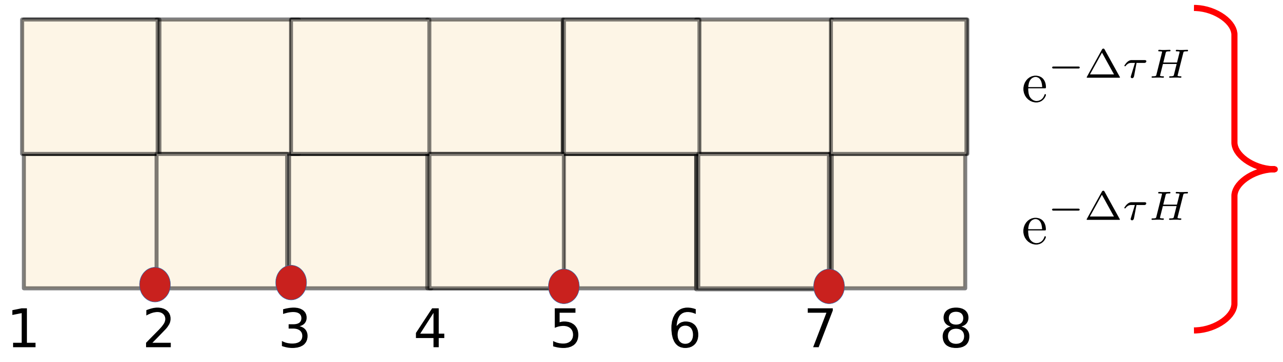
The term: $a_{i+1}^\dagger a_i a_i^\dagger a_{i+1}$ hops a particle from *i+1* to *i* and back to *i+1.*

The higher the power, the more such terms are included. In fact we need to keep all of them because the sum up to finite contributions.

That is why we need to calculate the transition matrix exactly, and that is not hard for a 4-site problem.

What if we do not split H into H_1 and H_2?

Quantum Monte Carlo: World Line Algorithm



We need matrix elements of both Steps because both change.

How will we evolve the particles from the first slice to the next?

- Suggest sequential moves for one particle at a time and apply Metropolis? We can but the matrix is much larger than before. Why not just do the 4-site problem?
- Suggest a move for all particles? The acceptance rate will be very low. Think of Ising: We only change one site at a time because if we change many dependent sites, the acceptance will not be high.
- This suggests that it is more efficient to do the checkerboard splitting like I showed before.

Quantum Monte Carlo: World Line Algorithm

Measurements:

$$
\begin{aligned}
\langle \mathsf{O} \rangle &\equiv \frac{1}{Z}\mathrm{Tr}\left(\mathsf{O}\mathrm{e}^{-\beta H}\right) \\
&= \frac{1}{Z}\sum_{\{\mathbf{n}\}} \langle \mathbf{n}^1|\mathrm{e}^{-\Delta\tau H_2}|\mathbf{n}^{2L}\rangle\langle \mathbf{n}^{2L}|\mathrm{e}^{-\Delta\tau H_1}|\mathbf{n}^{2L-1}\rangle\ldots \\
&\quad \ldots\langle \mathbf{n}^{2i+1}|\mathrm{e}^{-\Delta\tau H_2}\mathsf{O}|\mathbf{n}^{2i}\rangle\langle \mathbf{n}^{2i}|\mathrm{e}^{-\Delta\tau H_1}|\mathbf{n}^{2i-1}\rangle\ldots \\
&\quad \ldots\langle \mathbf{n}^{3}|\mathrm{e}^{-\Delta\tau H_2}|\mathbf{n}^{2}\rangle\langle \mathbf{n}^{2}|\mathrm{e}^{-\Delta\tau H_1}|\mathbf{n}^{1}\rangle \\
&= \frac{1}{Z}\sum_{\{\mathbf{n}\}} \langle \mathbf{n}^1|\mathrm{e}^{-\Delta\tau H_2}|\mathbf{n}^{2L}\rangle\langle \mathbf{n}^{2L}|\mathrm{e}^{-\Delta\tau H_1}|\mathbf{n}^{2L-1}\rangle\ldots \\
&\quad \ldots\langle \mathbf{n}^{2i+1}|\mathrm{e}^{-\Delta\tau H_2}\mathsf{O}|\mathbf{n}^{2i}\rangle \frac{\langle \mathbf{n}^{2i+1}|\mathrm{e}^{-\Delta\tau H_2}|\mathbf{n}^{2i}\rangle}{\langle \mathbf{n}^{2i+1}|\mathrm{e}^{-\Delta\tau H_2}|\mathbf{n}^{2i}\rangle}\langle \mathbf{n}^{2i}|\mathrm{e}^{-\Delta\tau H_1}|\mathbf{n}^{2i-1}\rangle\ldots \\
&\quad \ldots\langle \mathbf{n}^{3}|\mathrm{e}^{-\Delta\tau H_2}|\mathbf{n}^{2}\rangle\langle \mathbf{n}^{2}|\mathrm{e}^{-\Delta\tau H_1}|\mathbf{n}^{1}\rangle
\end{aligned}
$$

Quantum Monte Carlo: World Line Algorithm

Measurements:

$$
\begin{aligned}
\langle \mathsf{O} \rangle &= \frac{1}{Z} \sum_{\{\mathbf{n}\}} \langle \mathbf{n}^1 | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2L} \rangle \langle \mathbf{n}^{2L} | \mathrm{e}^{-\Delta\tau H_1} | \mathbf{n}^{2L-1} \rangle \ldots \\
&\ldots \langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle \frac{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} \mathsf{O} | \mathbf{n}^{2i} \rangle}{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle} \langle \mathbf{n}^{2i} | \mathrm{e}^{-\Delta\tau H_1} | \mathbf{n}^{2i-1} \rangle \ldots \\
&\ldots \langle \mathbf{n}^3 | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^2 \rangle \langle \mathbf{n}^2 | \mathrm{e}^{-\Delta\tau H_1} | \mathbf{n}^1 \rangle \\
&= \left.\begin{aligned}
&\frac{1}{Z} \sum_{\{\mathbf{n}\}} \langle \mathbf{n}^1 | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2L} \rangle \langle \mathbf{n}^{2L} | \mathrm{e}^{-\Delta\tau H_1} | \mathbf{n}^{2L-1} \rangle \ldots \\
&\ldots \langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle \langle \mathbf{n}^{2i} | \mathrm{e}^{-\Delta\tau H_1} | \mathbf{n}^{2i-1} \rangle \ldots \\
&\ldots \langle \mathbf{n}^3 | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^2 \rangle \langle \mathbf{n}^2 | \mathrm{e}^{-\Delta\tau H_1} | \mathbf{n}^1 \rangle
\end{aligned}\right\} \text{Boltzmann weight} \\
&\quad \left.\frac{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} \mathsf{O} | \mathbf{n}^{2i} \rangle}{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle}\right\} \text{Measurable}
\end{aligned}
$$

Quantum Monte Carlo: World Line Algorithm

The previous expression takes the form:

$$\langle \mathsf{O} \rangle = \sum_{\{\mathbf{n}\}} P_B[\{n_i\}] \frac{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} \mathsf{O} | \mathbf{n}^{2i} \rangle}{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle}$$

- We need to calculate the ratio of the matrix elements
- For quantities which are diagonal in occupation, this is very simple.

$$\begin{aligned} \langle \mathcal{F}(\hat{n}) \rangle &= \sum_{\{\mathbf{n}\}} P_B[\{n_i\}] \frac{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} \mathcal{F}(\hat{n}) | \mathbf{n}^{2i} \rangle}{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle} \\ &= \sum_{\{\mathbf{n}\}} P_B[\{n_i\}] \frac{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} \mathcal{F}(n) | \mathbf{n}^{2i} \rangle}{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle} \\ &= \sum_{\{\mathbf{n}\}} \mathcal{F}(n) P_B[\{n_i\}] \frac{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle}{\langle \mathbf{n}^{2i+1} | \mathrm{e}^{-\Delta\tau H_2} | \mathbf{n}^{2i} \rangle} \end{aligned}$$

Quantum Monte Carlo: World Line Algorithm

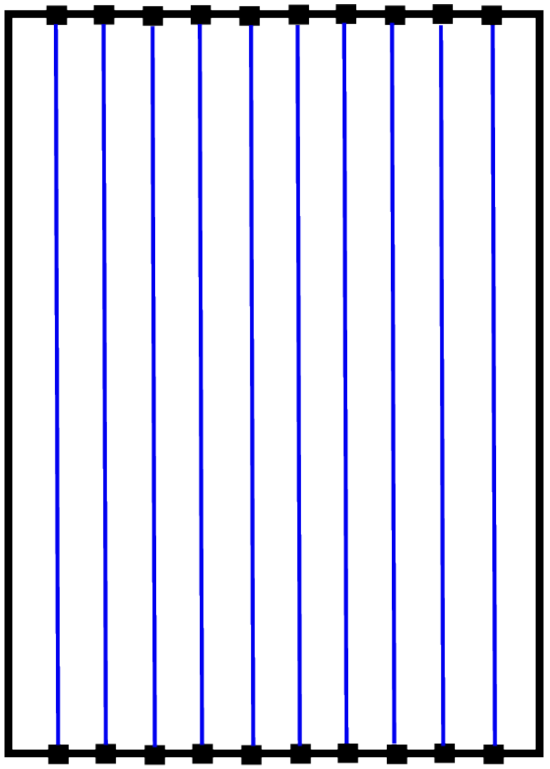
- We can calculate static (equal imaginary “time”) correlation functions.
- We can also calculate dynamic (separated in imaginary “time”) correlation functions.
- Use maximum entropy, for example, to get dynamic properties, like excited states etc.
- Correlations in the “time” direction also give us the superfluid density.
- We cannot calculate Green functions with this algorithm because the world lines are continuous, they do not break.

Quantum Monte Carlo: World Line Algorithm

Avery interesting quantity for Bose systems is the superfluid density:

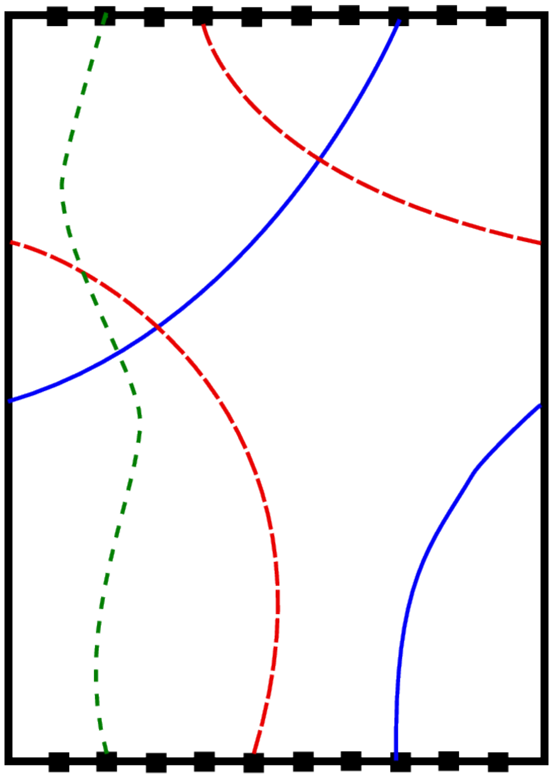
$$\rho_s = \frac{\langle W^2 \rangle}{2t\beta L^{d-2}}$$

where *W* is the winding number of the world line configuration.



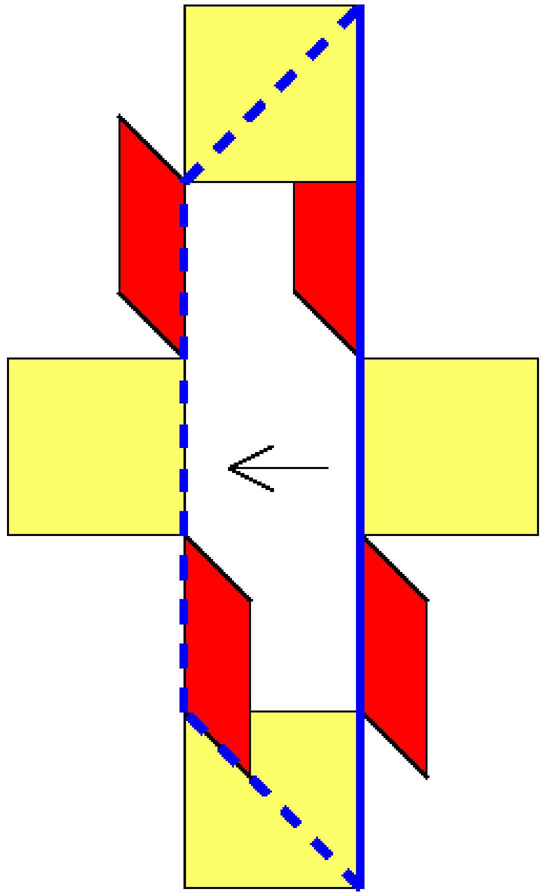
All windings = 0

Green *W* = 0
Blue *W* = +1
Red *W* = - 1

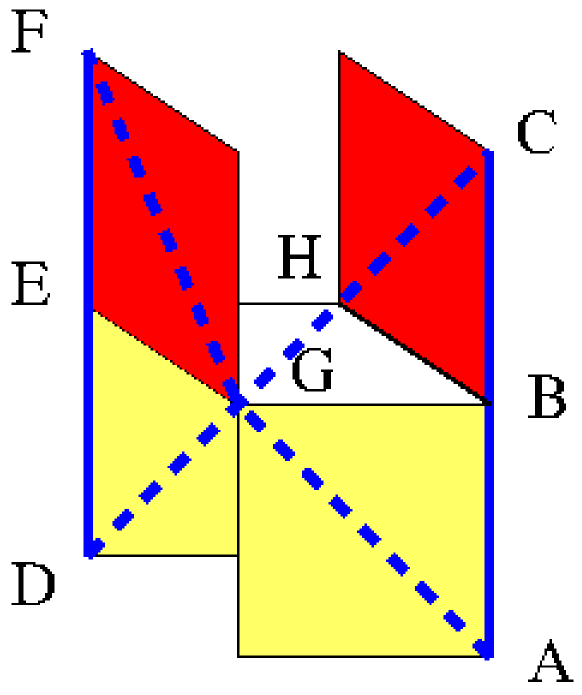


Quantum Monte Carlo: World Line Algorithm

In two dimensions, we do the same. However, one full time step is now accomplished with 4 exponentials due to the breakup of the H.



Drag moves



Twist moves

Quantum Monte Carlo: Stochastic Green Function Algorithm

References:

- ``Worm Algorithm for Problems of Quantum and Classical Statistics'', Nikolay Prokof'ev and Boris Svistunov, *Understanding Quantum Phase Transitions*, CRC Press, Taylor and Francis Group, Lincoln D. Carr (editor), (2011).
- ``Quantum Monte Carlo simulation in the canonical ensemble at finite temperature'', K. Van Houcke, S. M. A. Rombouts, and L. Pollet, Phys. Rev. E73, 056703 (2006).
- ``Stochastic Green function algorithm'', V. G. Rousseau, Phys. Rev. E77, 056705 (2008).
- ``Directed update for the stochastic Green function algorithm'', V. G. Rousseau, Phys. Rev. E78, 056707 (2008).

Quantum Monte Carlo: Stochastic Green Function Algorithm

General features:

- Extension of the canonical worm algorithm.
- Works well in canonical and grand canonical ensembles.
- Continuous imaginary time: No $\Delta\tau$ errors.
- World lines are broken and ends move around: Can calculate Green functions.
- Good for complicated Hamiltonians.

Quantum Monte Carlo: Stochastic Green Function Algorithm

Examples of Hamiltonians where SGF is useful:

Spin-1/2 Bose-Hubbard model:

$$\begin{aligned} H &= -t \sum_{\sigma,\langle \mathbf{r},\mathbf{r}'\rangle} \left(a^\dagger_{\sigma\mathbf{r}} a_{\sigma\mathbf{r}'} + a^\dagger_{\sigma\mathbf{r}'} a_{\sigma\mathbf{r}} \right) - \mu \sum_{\sigma,\mathbf{r}} \hat{n}_{\sigma\mathbf{r}} \\ &\quad + \frac{U_0}{2} \sum_{\sigma,\mathbf{r}} \hat{n}_{\sigma\mathbf{r}}(\hat{n}_{\sigma\mathbf{r}} - 1) + (U_0 + U_2) \sum_{\mathbf{r}} \hat{n}_\uparrow(\mathbf{r}) \hat{n}_\downarrow(\mathbf{r}) \\ &\quad - \frac{U_2}{2} \sum_{\mathbf{r}} \left(a^\dagger_{\uparrow\mathbf{r}} a^\dagger_{\uparrow\mathbf{r}} a_{\downarrow\mathbf{r}} a_{\downarrow\mathbf{r}} + a^\dagger_{\downarrow\mathbf{r}} a^\dagger_{\downarrow\mathbf{r}} a_{\uparrow\mathbf{r}} a_{\uparrow\mathbf{r}} \right) \end{aligned}$$

The one dimensional Rabi-Hubbard model:

$$\begin{aligned} H &= -J \sum_i (a^\dagger_i a_{i+1} + a^\dagger_{i+1} a_i) + \sum_i \left(\omega_s \sigma^+_i \sigma^-_i + \omega_l a^\dagger_i a_i \right) \\ &\quad - g \sum_i (\sigma^-_i + \sigma^+_i)(a_i + a^\dagger_i) \end{aligned}$$

Quantum Monte Carlo: Stochastic Green Function Algorithm

As usual, we start with the partition function:

$$Z(\beta) = \mathrm{Tr}\, \mathrm{e}^{-\beta H}$$

Now we will write the *H* in the form:

$$H = \mathsf{U} - \mathsf{T}$$

Where U is the diagonal part and depends only on number operators. T is the nondiagonal part which can be more complicated than just the hopping term: It can have ring exchange operators, conversion from one particle type to another, etc ...

The minus sign in front of *T* is needed next page. *T* itself now is assumed to have positive signs.

Define the imaginary time-dependent operator:

$$\mathsf{T}(\tau) \equiv \mathrm{e}^{\tau \mathsf{U}}\ \mathsf{T}\ \mathrm{e}^{-\tau \mathsf{U}}$$

Quantum Monte Carlo:
Stochastic Green Function Algorithm

In the interaction picture we can write:

$$
\begin{aligned}
\mathrm{e}^{-\beta H} &= \mathrm{e}^{-\beta U} \exp\left(\int_0^\beta \mathrm{d}\tau \ \mathsf{T}(\tau) \right) \\
&= \mathrm{e}^{-\beta U} \left\{ 1 + \int_0^\beta \mathrm{d}\tau \ \mathsf{T}(\tau) + \int_0^\beta \mathrm{d}\tau_1 \int_{\tau_1}^\beta \mathrm{d}\tau_2 \ \mathsf{T}(\tau_2)\mathsf{T}(\tau_1) + \ldots \right\} \\
&= \mathrm{e}^{-\beta \mathsf{U}} \sum_{n \geq 0} \int_{0<\tau_1<\tau_2<\cdots<\tau_n<\beta} \mathrm{d}\tau_1 \ldots \mathrm{d}\tau_n \mathsf{T}(\tau_n) \ldots \mathsf{T}(\tau_2)\mathsf{T}(\tau_1)
\end{aligned}
$$

The exponential of the nondiagonal part is **time ordered**. It takes into account all possible (imaginary) times when the *T* can act. So, the partition function can be written as:

$$
Z(\beta) = \mathrm{Tr} \left\{ \mathrm{e}^{-\beta \mathsf{U}} \sum_{n \geq 0} \int_{0<\tau_1<\tau_2<\cdots<\tau_n<\beta} \mathrm{d}\tau_1 \ldots \mathrm{d}\tau_n \mathsf{T}(\tau_n) \ldots \mathsf{T}(\tau_2)\mathsf{T}(\tau_1) \right\}
$$

Quantum Monte Carlo: Stochastic Green Function Algorithm

$$
\begin{aligned}
Z(\beta) &= \sum_{\{\Psi_0\},n\geq 0} \int_{0<\tau_1<\tau_2<\cdots<\tau_n<\beta} \mathrm{d}\tau_1 \ldots \mathrm{d}\tau_n \langle \Psi_0 | \mathrm{e}^{-\beta \mathsf{U}}\, \mathsf{T}(\tau_n) \ldots \mathsf{T}(\tau_2) \mathsf{T}(\tau_1) | \Psi_0 \rangle \\
&= \sum_{\{\Psi_n\},n\geq 0} \int_{0<\tau_1<\tau_2<\cdots<\tau_n<\beta} \mathrm{d}\tau_1 \ldots \mathrm{d}\tau_n \langle \Psi_0 | \mathrm{e}^{-\beta \mathsf{U}}\, \mathsf{T}(\tau_n) | \Psi_{n-1} \rangle \\
&\qquad \langle \Psi_{n-1} | \mathsf{T}(\tau_{n-1}) | \Psi_{n-2} \rangle \langle \Psi_{n-2} | \mathsf{T}(\tau_{n-2}) | \Psi_{n-3} \rangle \ldots \\
&\qquad \ldots \langle \Psi_2 | \mathsf{T}(\tau_2) | \Psi_1 \rangle \langle \Psi_1 | \mathsf{T}(\tau_1) | \Psi_0 \rangle
\end{aligned}
$$

So far this is just the usual partition function written is a slightly different form.

Note: The imaginary time insertions are not at discretized imaginary time steps, they can be anywhere between 0 and β. That is what makes this a continuous time algorithm.

Quantum Monte Carlo: Stochastic Green Function Algorithm

Now let us write an “extended” partition function:

$Z(\beta,\tau) \equiv \mathrm{Tr}\, \mathrm{e}^{-(\beta-\tau)H}\,\mathsf{G}\,\mathrm{e}^{-\tau H}$ G is an arbitrary operator for now.

Defining, $\mathsf{G}(\tau) \equiv \mathrm{e}^{\tau \mathsf{U}}\,\mathsf{G}\,\mathrm{e}^{-\tau \mathsf{U}}$ we can show as before:

$$\begin{aligned} Z(\beta,\tau) \quad = \quad & \sum_{\{\Psi_n\},n\geq 0} \int_{0<\tau_1<\tau_2<\cdots<\tau_n<\beta} \mathrm{d}\tau_1 \ldots \mathrm{d}\tau_n \langle \Psi_0 | \mathrm{e}^{-\beta \mathsf{U}}\, \mathsf{T}(\tau_n) | \Psi_{n-1} \rangle \\ & \quad \langle \Psi_{n-1} | \mathsf{T}(\tau_{n-1}) | \Psi_{n-2} \rangle \langle \Psi_{n-2} | \mathsf{T}(\tau_{n-2}) | \Psi_{n-3} \rangle \ldots \\ & \ldots \langle \Psi_{L+1} | \mathsf{T}(\tau_L) | \Psi_L \rangle \langle \Psi_L | \mathsf{G}(\tau) | \Psi_R \rangle \langle \Psi_R | \mathsf{T}(\tau_r) | \Psi_{R-1} \rangle \\ & \ldots \langle \Psi_2 | \mathsf{T}(\tau_2) | \Psi_1 \rangle \langle \Psi_1 | \mathsf{T}(\tau_1) | \Psi_0 \rangle \end{aligned}$$

This is the same as the previous partition function, except we have this G acting at τ which we introduced but not yet explained.

Quantum Monte Carlo: Stochastic Green Function Algorithm

Define:

$$\mathsf{A}^\dagger \equiv \mathsf{a}^\dagger \frac{1}{\sqrt{\mathsf{n}+1}}, \quad \mathsf{A} \equiv \frac{1}{\sqrt{\mathsf{n}+1}}\mathsf{a} \longrightarrow \mathsf{A}^\dagger|n\rangle = |n+1\rangle, \quad \mathsf{A}|n\rangle = |n-1\rangle$$

Then, *G* is defined by:

$$\mathsf{G} \equiv \sum_{p,q=0}^{\infty} g_{pq} \sum_{\{i_p|j_q\}} \prod_{k=1}^{p} \mathsf{A}^\dagger_{i_k} \prod_{\ell=1}^{q} \mathsf{A}_{j_\ell}$$

- g_{pq} is a matrix of our choice which depends on the application.
- It determines how many creation and annihilation operators can be be inserted.
- Typically, it decreases rapidly with number of insertions.
- Example: $$\begin{aligned} g_{pq} &= 1 & (p+q \leq 2) \\ &= \mathrm{e}^{-4(2-p-q)^2} & (p+q > 2) \end{aligned}$$

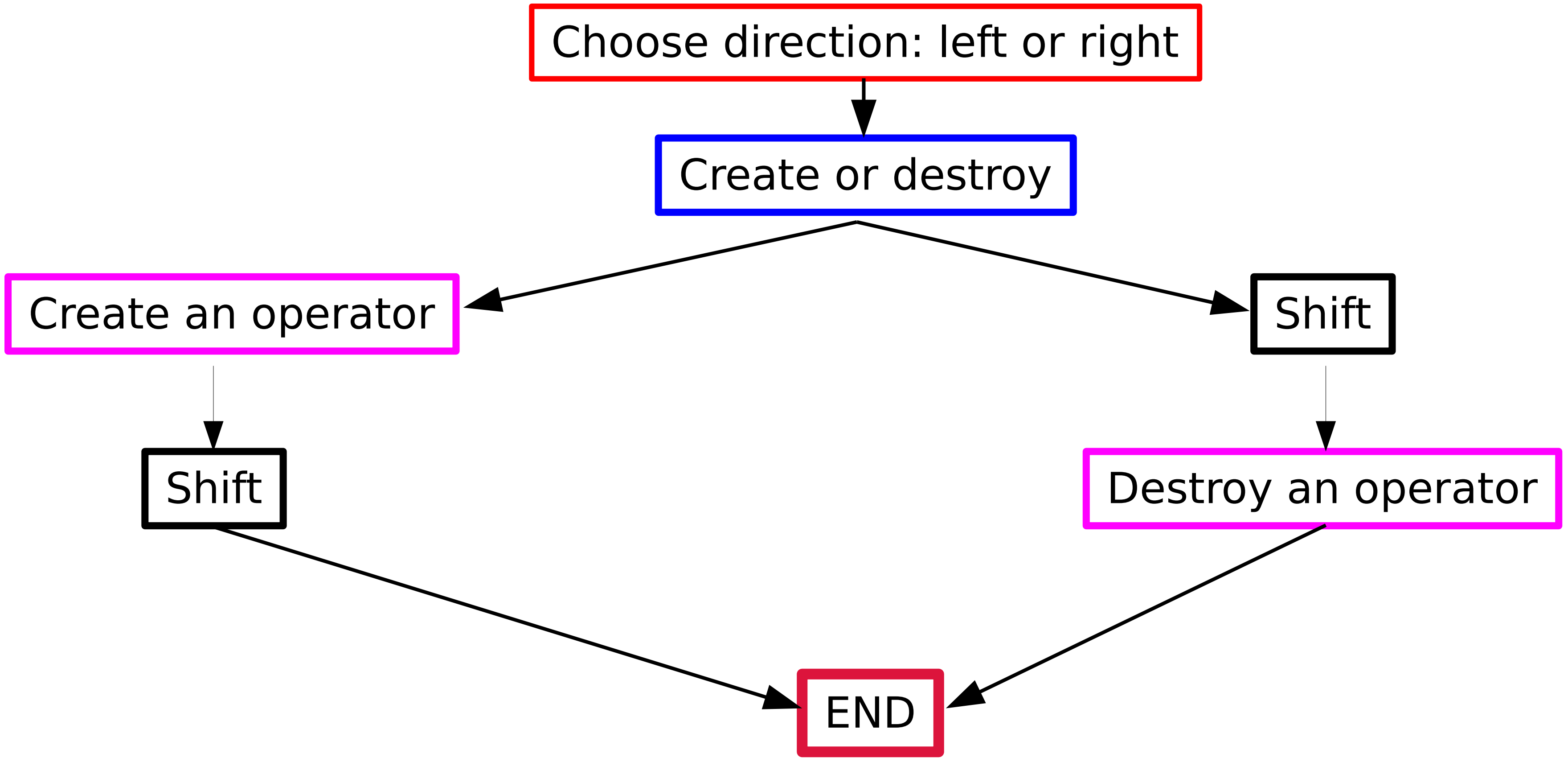
Quantum Monte Carlo: Stochastic Green Function Algorithm

$$\mathsf{G} \equiv \sum_{p,q=0}^{\infty} g_{pq} \sum_{\{i_p|j_q\}} \prod_{k=1}^{p} \mathsf{A}^{\dagger}_{i_k} \prod_{\ell=1}^{q} \mathsf{A}_{j_\ell}$$

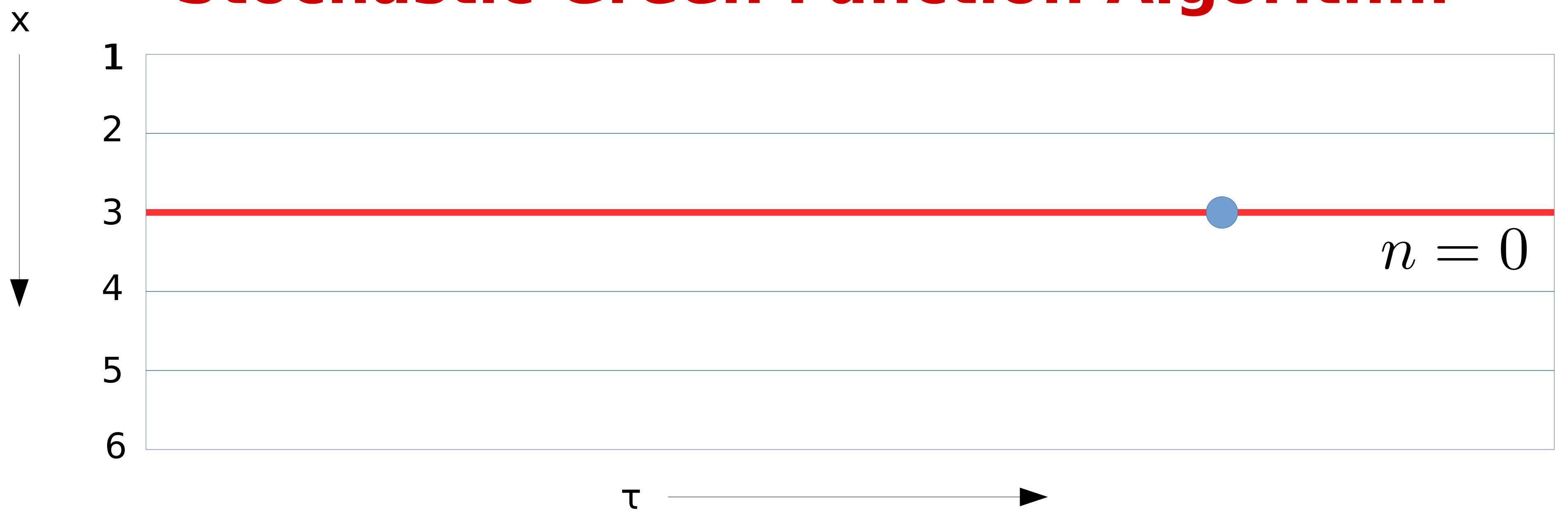
- i_k and j_l are site indices where we insert the operators
- $\{i_p|j_q\}$ denotes that all site indices in subset *i* are different from all indices in subset *j*. But one or more indices in a subset can be equal.
- The operator G inserts inserts breaks in the world lines.
- These are the “worms” in the algorithm.
- When the world lines close, we have a configuration of continuous lines.
- We can then measure diagonal quantities.

Quantum Monte Carlo: Stochastic Green Function Algorithm

$$
\begin{aligned}
Z(\beta,\tau) \quad = \quad & \sum_{\{\Psi_n\},n\geq 0} \int_{0<\tau_1<\tau_2<\cdots<\tau_n<\beta} \mathrm{d}\tau_n \ldots \mathrm{d}\tau_n \langle \Psi_0 | \mathrm{e}^{-\beta \mathsf{U}}\, \mathsf{T}(\tau_n) | \Psi_{n-1} \rangle \\
& \quad \langle \Psi_{n-1} | \mathsf{T}(\tau_{n-1}) | \Psi_{n-2} \rangle \langle \Psi_{n-2} | \mathsf{T}(\tau_{n-2}) | \Psi_{n-3} \rangle \ldots \\
& \ldots \langle \Psi_{L+1} | \mathsf{T}(\tau_L) | \Psi_L \rangle \langle \Psi_L | \mathsf{G}(\tau) | \Psi_R \rangle \langle \Psi_R | \mathsf{T}(\tau_r) | \Psi_{R-1} \rangle \\
& \ldots \langle \Psi_2 | \mathsf{T}(\tau_2) | \Psi_1 \rangle \langle \Psi_1 | \mathsf{T}(\tau_1) | \Psi_0 \rangle
\end{aligned}
$$

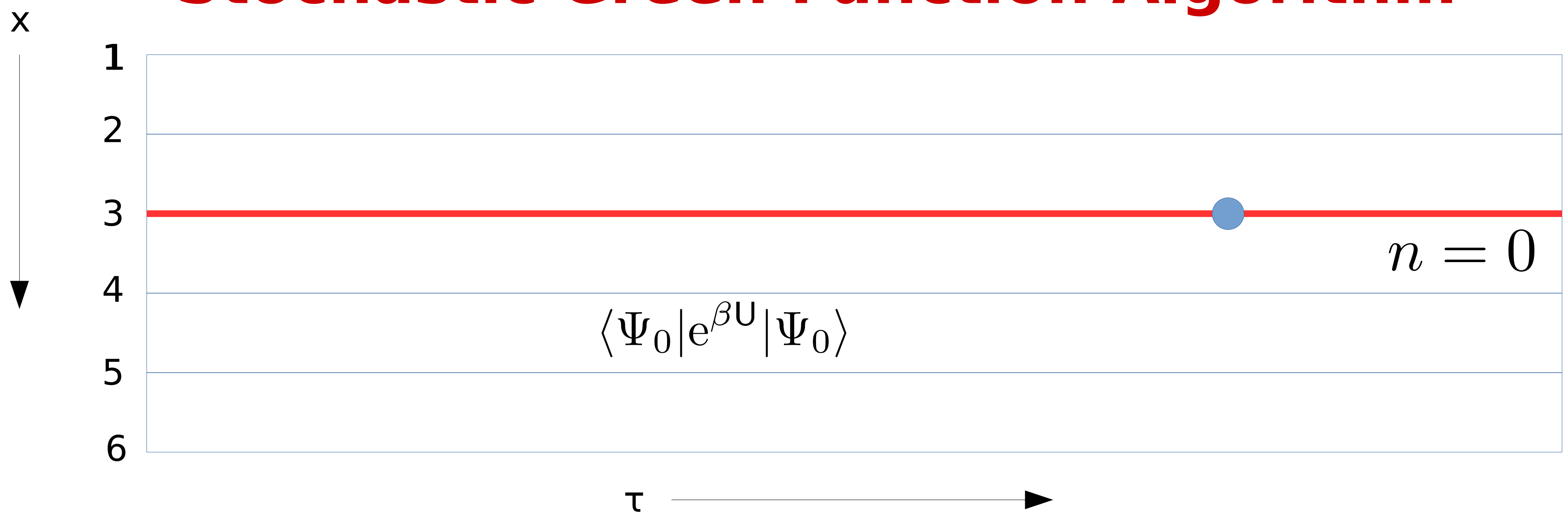


Quantum Monte Carlo: Stochastic Green Function Algorithm

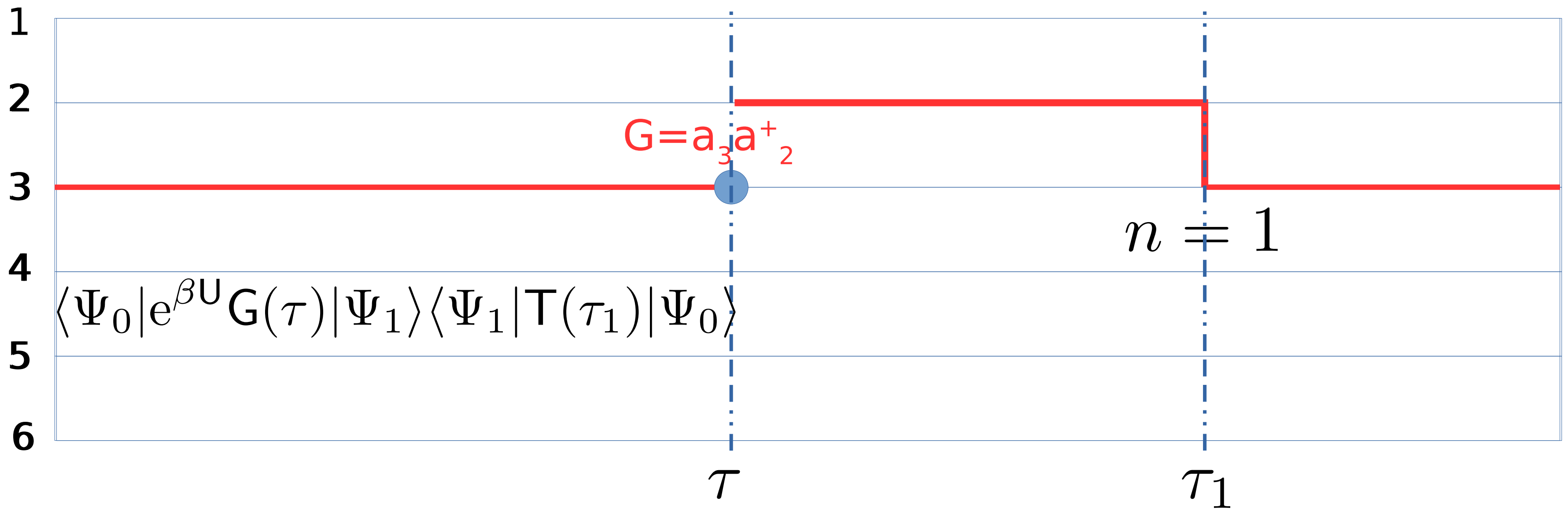


Choose: Move left. Choose: Create $\longrightarrow$ Insert T at current location, move left

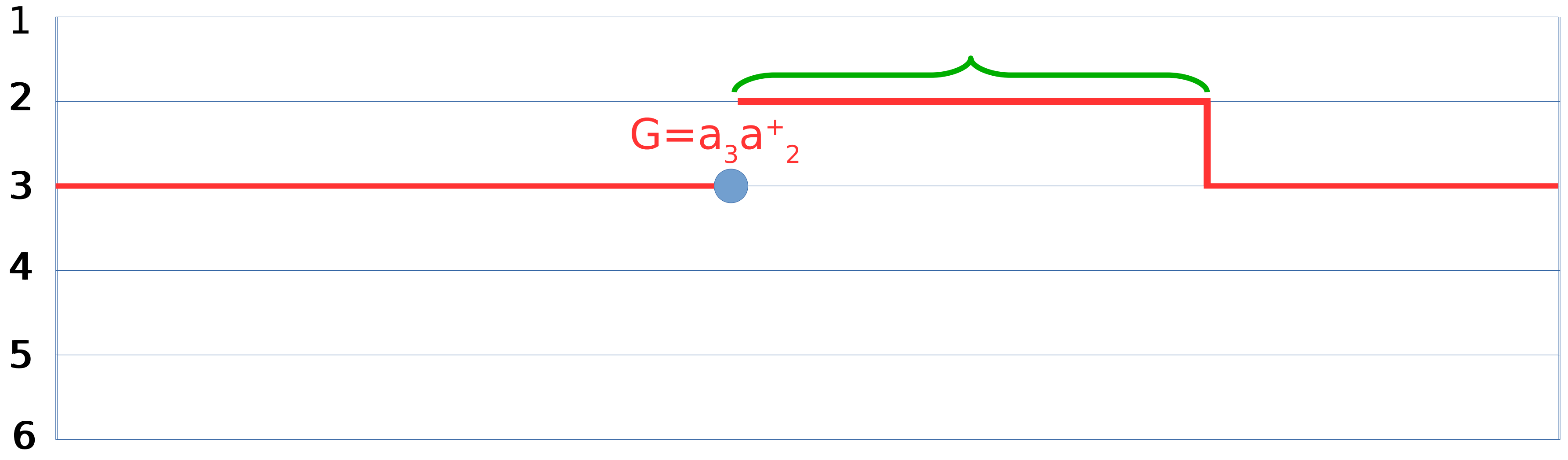
Quantum Monte Carlo: Stochastic Green Function Algorithm



Choose: Move left. Choose: Create $\longrightarrow$ Insert T at current location, move left

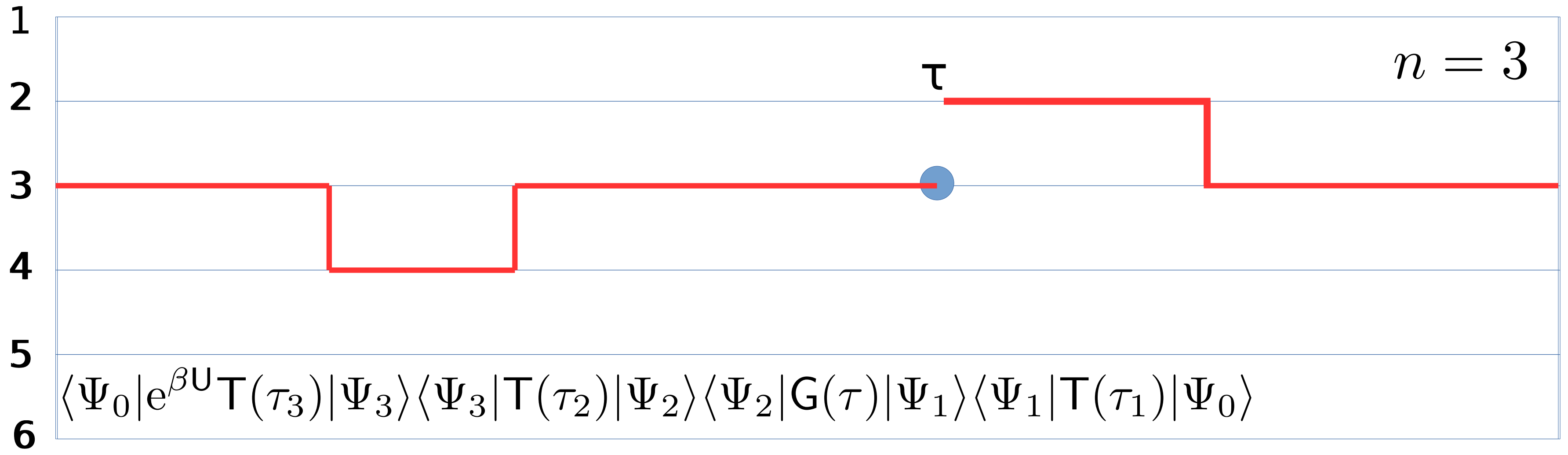

1
2
3
4
5
6
$G=a_3 a^+_2$
$n=1$
$\langle \Psi_0 | e^{\beta \mathsf{U}} \mathsf{G}(\tau) | \Psi_1 \rangle \langle \Psi_1 | \mathsf{T}(\tau_1) | \Psi_0 \rangle$
τ
τ_1

Quantum Monte Carlo: Stochastic Green Function Algorithm



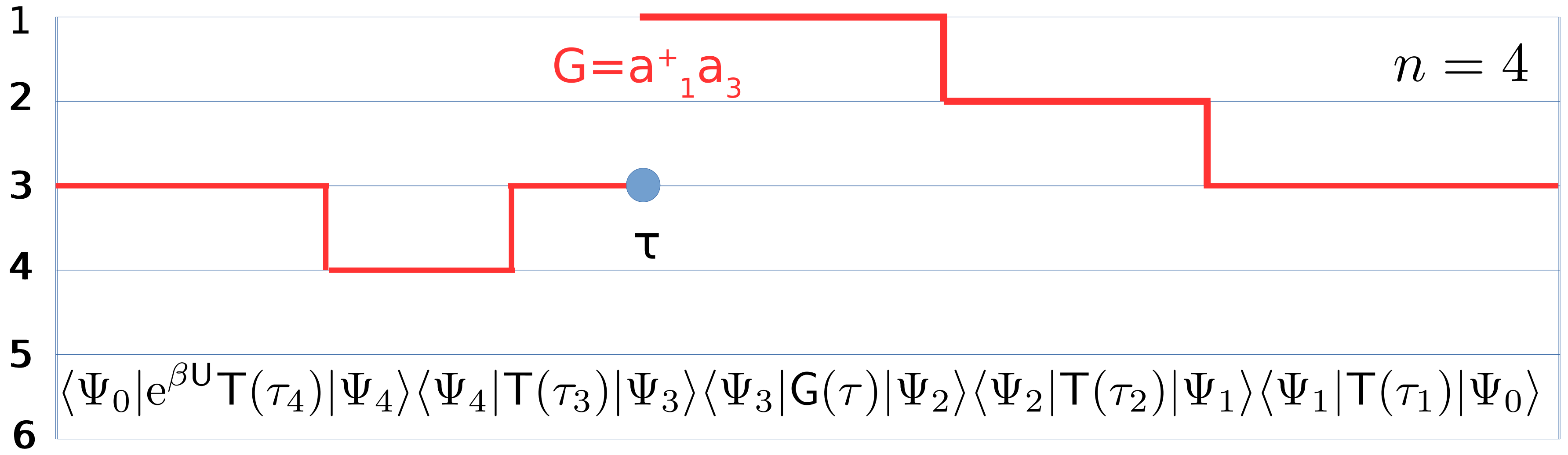
- For illustration, I have taken T to be a hop between near neighbors.
- In general, the nondiagonal operator can be more complicated.
- How far to shift? It is chosen according to a probability distribution, typically exponential.

Quantum Monte Carlo: Stochastic Green Function Algorithm



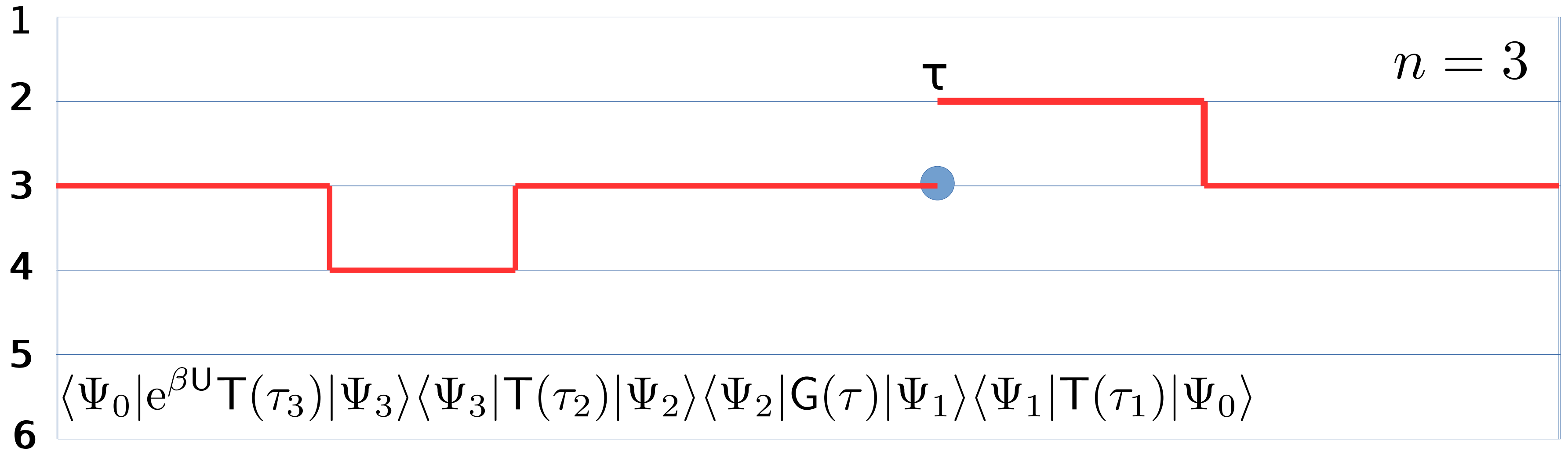
$\langle\Psi_0|e^{\beta U}T(\tau_3)|\Psi_3\rangle\langle\Psi_3|T(\tau_2)|\Psi_2\rangle\langle\Psi_2|G(\tau)|\Psi_1\rangle\langle\Psi_1|T(\tau_1)|\Psi_0\rangle$

Now, shift left and create: The shift must stop before τ_L because of time ordering



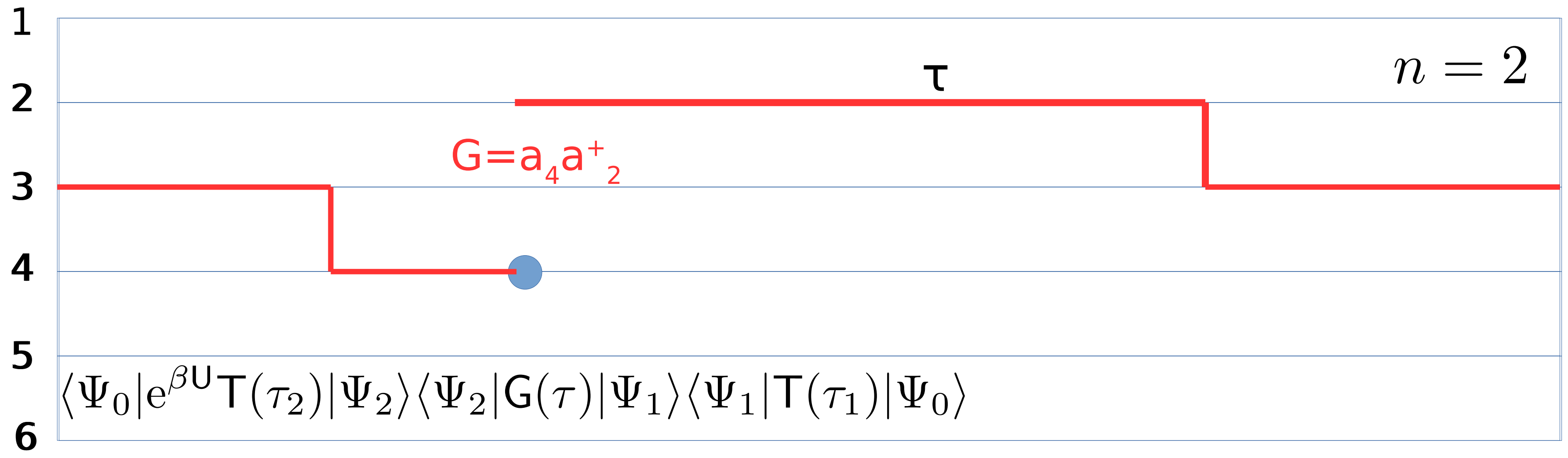
$\langle\Psi_0|e^{\beta U}T(\tau_4)|\Psi_4\rangle\langle\Psi_4|T(\tau_3)|\Psi_3\rangle\langle\Psi_3|G(\tau)|\Psi_2\rangle\langle\Psi_2|T(\tau_2)|\Psi_1\rangle\langle\Psi_1|T(\tau_1)|\Psi_0\rangle$

Quantum Monte Carlo: Stochastic Green Function Algorithm



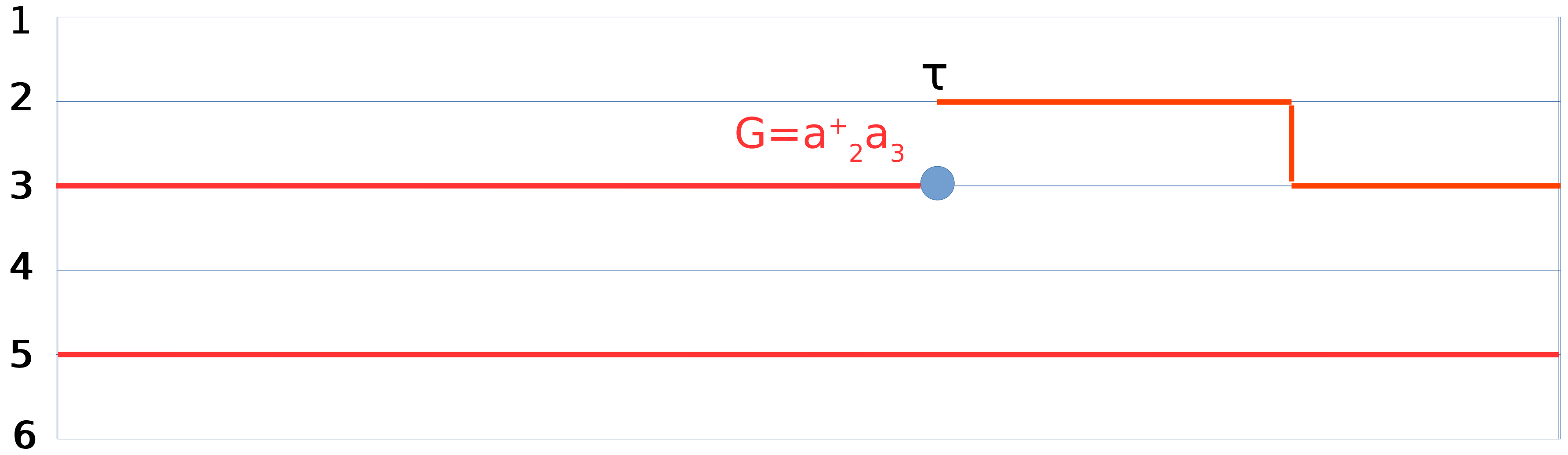
$\langle\Psi_0|e^{\beta \mathrm{U}}\mathrm{T}(\tau_3)|\Psi_3\rangle\langle\Psi_3|\mathrm{T}(\tau_2)|\Psi_2\rangle\langle\Psi_2|\mathrm{G}(\tau)|\Psi_1\rangle\langle\Psi_1|\mathrm{T}(\tau_1)|\Psi_0\rangle$

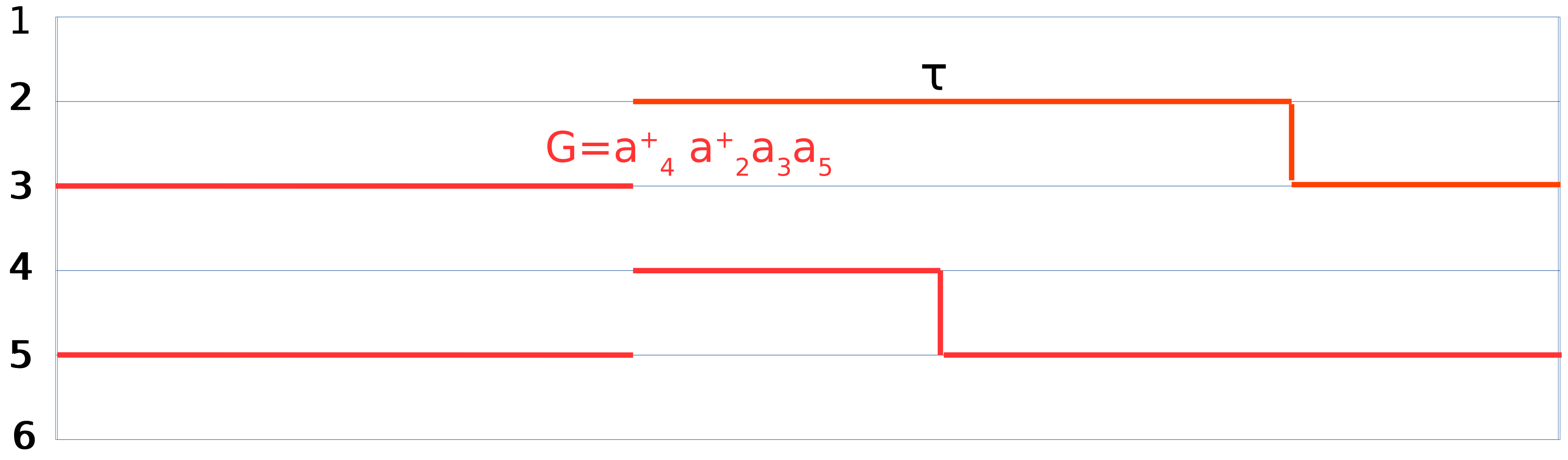
If we shift left and destroy, then:



$\langle\Psi_0|e^{\beta \mathrm{U}}\mathrm{T}(\tau_2)|\Psi_2\rangle\langle\Psi_2|\mathrm{G}(\tau)|\Psi_1\rangle\langle\Psi_1|\mathrm{T}(\tau_1)|\Psi_0\rangle$

Quantum Monte Carlo: Stochastic Green Function Algorithm





Quantum Monte Carlo: Stochastic Green Function Algorithm

- The head and tail of all worms are at the same imaginary time.
- Insures canonical ensemble.
- The canonical worm algorithm: Only one head and tail (one worm) and at the same imaginary time.
- Worm algorithm: Only the head or tail moves and is not at the same time.
- Worm algorithm can change the number of world lines, it is grand canonical.
- Measure Green functions: Just count how often a worm of a certain length appears.
- Measure diagonal quantities: Only when worms close. Then measurement is trivial.

Quantum Monte Carlo: Stochastic Green Function Algorithm

Detailed Balance:

- Let P_i (P_f) be the probability of the initial (final) configuration.
- $S_{i\longrightarrow f}$ the transition probability from *i* to *f.*
- $A_{i\longrightarrow f}$ the acceptance rate for the transition from *i* to *f.*

Detailed balance condition is: $$P_i S_{i\to f} A_{i\to f} = P_f S_{f\to i} A_{f\to i}$$

Typically we choose the Metropolis solution:

$$A_{i\to f} = \min(1, q)$$

with $$q = \frac{P_f S_{f\to i}}{P_i S_{i\to f}}$$

For details of the probabilities, P, S and the probabilities to choose left or right, create or destroy, see the references. They are discussed in detail.

Quantum Monte Carlo: Some Last comments

- The world line algorithm is intuitive and easy to code, especially in one dimension.
- It is still very useful for simpler bosonic models.
- Cannot measure Green functions.
- The SGF and other worm algorithms are powerful and, depending on the application, should be chosen.
- Always choose an algorithm that is adapted to the problem.
- Choose if it is more practical to simulate in the canonical or grand canonical ensembles.
- Algorithm development is an active, and fun, part of what we do!