

Diffusion Monte Carlo

The trajectories.

We're going to write a series of implementations of diffusion Monte Carlo. Each implementation will generate data in the following form:

tau	step	local energy	average	weight	local energy	variance	weight	variance	reference energy
We will do our initial testing for $\tau \simeq 0.01$ and using $\alpha = 2, \beta = 0.5$ in the Slater-Jastrow wave function.									

1 Importance sampling

For 1000 walkers, generate dynamics as follows:

$$x_{n+1} = x_n + \sqrt{\tau}\chi + \tau \frac{\nabla \Psi_T(x_n)}{\Psi_T(x_n)}, \quad (1)$$

where χ is a random variable. You can add an acceptance/rejection step to this just like in VMC, although the reasoning is quite different. Set all weights to one. Make a CSV file and use Pandas to analyze your data.

Remember:

$$a(x' \leftarrow x) = \min \left(1, \frac{\Psi_T^2(x')T(x \leftarrow x')}{\Psi_T^2(x)T(x' \leftarrow x)} \right) \quad (2)$$

$$T(x' \leftarrow x) = \frac{\left(x' - x - \tau \frac{\nabla \Psi_T(x)}{\Psi_T(x)} \right)^2}{2\tau} \quad (3)$$

- You should be able to get the VMC result for your trial wave function. Does it match?

2 Importance sampling with weights

Now we will update the weights w_i . Set

$$w* = \exp[-\tau E_L(x_{n+1})] \quad (4)$$

each step. You probably don't need to perform this for too many steps!

- Track the weights of the walkers and their values as a function of step. What happens?

3 Fixing the normalization

We want the weights to average around 1. Use a shift

$$w* = \exp[-\tau(E_L(x_{n+1}) - E_{\text{ref}})] \quad (5)$$

We can adjust E_{ref} to ensure the weights average to one:

$$E_{\text{ref}} = E_{\text{ref}} - \log(\langle w \rangle) \quad (6)$$

where $\langle w \rangle$ is the average weight.

- Now plot the weights. They should average to 1 but what happens to the variance?

4 Branching

We now want to split the walkers with too-large weights and kill the walkers with too-small weights. There are many ways to do this; we will choose one that lets us keep the number of walkers constant. Every step:

1. Stack up weights (`np.cumsum`)
2. Throw random numbers for each walker (`np.random.random`)
3. Copy the walker corresponding to where each random number landed (`np.searchsorted`)
4. Assign new walkers (slicing)
5. Set all weights to the average weight.

5 Using the algorithm

Congratulations! We have now implemented the DMC algorithm. Let's check some things:

- The exact energy is -2.903724. Do we get that in the $\tau \rightarrow 0$ limit?

6 More things to do

- As you change the number of walkers, how does the behavior of E_{ref} change?
- Implement a triplet wave function and fixed node. What is the node of this wave function?
- What might you change to improve the timestep error?