Variational Monte Carlo Method

Applied to find ground-states of an harmonic oscillator and the 4He nucleus

Damiano Santoferrara

damiano.santoferrara@studio.unibo.it
Università di Bologna

November 14, 2024

Motivation

Where did I use VMC Method:

- VMC is used to sample from a parametrized distribution ansatz
- In this case the distributions are the wave functions of the two systems analyzed and the variables are positions in space

Why did I use VMC Method:

- VMC uses the metropolis algorithm to sample variables from a given distribution. This is better than sampling over all possible values of x without taking information from the distribution

VMC Algorithm applied to the harmonic oscillator scheme

Here we have the usual harmonic oscillator with Hamiltonian: $H = -\frac{d^2}{dx^2} + x^2$

- Sample positions using the Metropolis algorithm
- Use sampled positions to compute $\Psi(x) = \frac{\sqrt{\alpha}}{\pi^{1/4}} e^{-\alpha^2 x^2/2}$ for a given α and local energies $E_L = \frac{H\Psi_0}{\Psi_0} = \alpha^2 + x^2(1-\alpha^4)$
- Average over all the energies.
- Repeat the previous steps for many values of α and find the minimum energy. This should correspond to the ground-state $\Psi_0(x) = \frac{1}{\pi^{1/4}} e^{-x^2/2}$

Metropolis sampling algorithm

Metropolis sampling algorithm is composed by a proposal and of an acceptance criterion. The new step is connected to the previous one.

```
samples = []
    x = np.random.uniform(-1, 1) # Initial position

    for _ in range(num_samples):
        # Propose a new position
        x_new = x + step_size * (np.random.rand() - 0.5) #random.rand
            generates rdm between 0 and 1 which becomes between -0.5 and 0.5

        # Metropolis acceptance criterion
        if np.random.rand() < (psi_trial(x_new, alpha) / psi_trial(x, alpha)
            ) ** 2:
            x = x_new # Accept the move

        samples.append(x)
```

Metropolis acceptance criterion

The probability of accepting a proposed sample is:

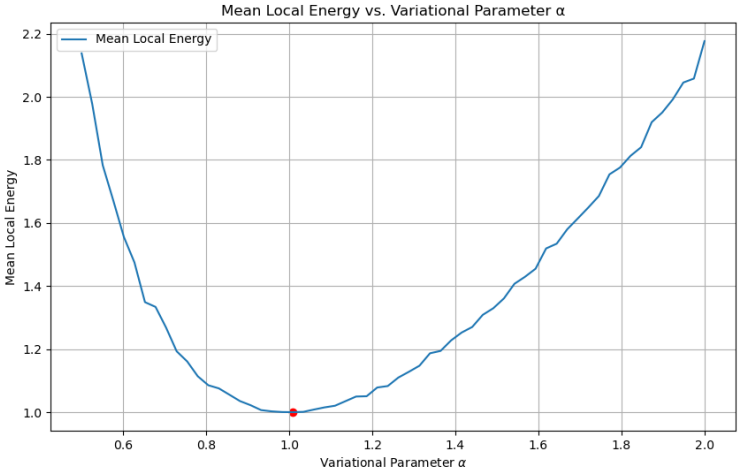
$$A = \min\left(1, \frac{|\Psi(x_{new})|^2}{|\Psi(x_{old})|^2}\right) \tag{1}$$

- When $|\Psi(x_{new})|^2 > |\Psi(x_{old})|^2$ the probability of accepting the "move" is 1.
- When $|\Psi(x_{new})|^2 < |\Psi(x_{old})|^2$ the probability is proportional to the distribution. So there is a tiny probability to accept less probable points. This prevents the algorithm from being stuck in only one region.

In the code this is implemented with:

```
# Metropolis acceptance criterion
    if np.random.rand() < (psi_trial(x_new, alpha) / psi_trial(x, alpha))
        ** 2:
          x = x_new # Accept the move
```

Results


Mean Local Energy vs. Variational Parameter α
Mean Local Energy
Mean Local Energy
Variational Parameter α
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0

Theory

To model the inter-nucleon potential we use the Afnan-Tang S3 interaction. After having averaged from thentwo channels $l = 0$ (singlet) and $l = 3$ (triplet), we obtain:

$$V = 1000e^{-3r^2} - 163.35e^{-1.05r^2} - 21.5e^{-0.6r^2} - 83e^{-0.8r^2} - 11.5e^{-0.4r^2} \tag{2}$$

To model an ansatz for the wave-function we use a Jastrow factor (that models the) multiplied by radial functions from the harmonic oscillator. A clear way to write this is:

$$\Psi(r) = e^{-\gamma r^2} + ae^{-(\beta+\gamma)r^2} \tag{3}$$

This time we have the three parameters a, β, γ.
Remark: we do not consider the spin-isospin orientation in this simulation. Still wew will be able to obtain a good result.

Metropolis sampling

```
for step in range(n_steps):
        new_positions = positions + np.random.normal(0, 0.5, positions.shape
            )
        psi_old = trial_wave_function(positions, a, beta, gamma)
        psi_new = trial_wave_function(new_positions, a, beta, gamma)
        acceptance_ratio = (psi_new / psi_old) ** 2
        if np.random.rand() < acceptance_ratio:
            positions = new_positions

        E_local = local_energy(positions, a, beta, gamma)
        energy_samples.append(E_local)
```

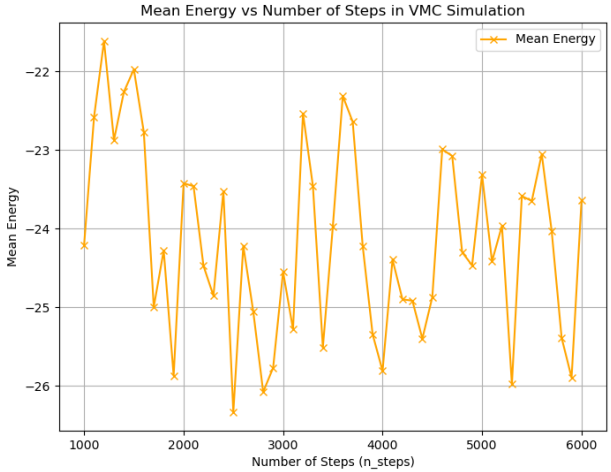
Thermalization

Since the initial position is chosen at random, it may take some steps before the sampling arrives to correctly represent the wanted distribution. To solve this problem we may take some "Thermalization steps".

```
# Thermalization phase (accepted steps without recording the resulting energy)
for _ in range(thermalization_steps):
    # Proposing new positions
    new_positions = positions + np.random.normal(0, 0.5, positions.shape)
    psi_old = trial_wave_function(positions, a, beta, gamma)
    psi_new = trial_wave_function(new_positions, a, beta, gamma)

    # Define the metropolis acceptance condition
    acceptance_ratio = (psi_new / psi_old) ** 2
    if np.random.rand() < acceptance_ratio:
        positions = new_positions # Accepting step
```

Results

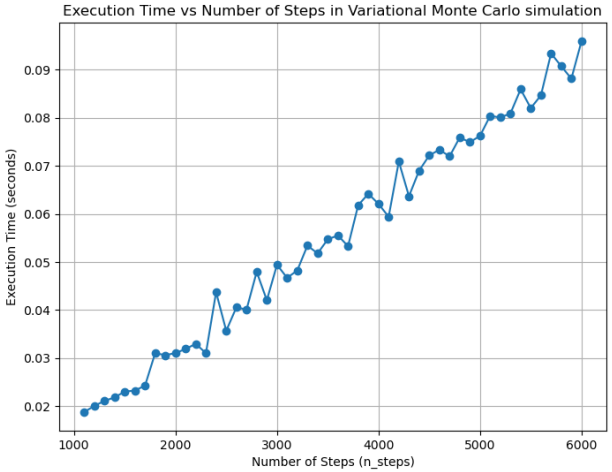

Mean Energy vs Number of Steps in VMC Simulation
Mean Energy
Mean Energy
−22
−23
−24
−25
−26
1000
2000
3000
4000
5000
6000
Number of Steps (n_steps)

Benchmark

```
# Define the range for n_steps
n_steps_values = range(1000, 6001, 100)
run_times = []

# Loop through each n_steps value and measure execution time
for na_steps in n_steps_values:
    start_time = time.time() # Start timing
    mean_energy, std_error = variational_monte_carlo(n_particles=4, n_steps=
        na_steps, thermalization_steps=300, a=a_fixed, beta=beta_fixed,
        gamma=gamma_fixed)
    end_time = time.time() # End timing
    run_times.append(end_time - start_time) # Calculate and store the run
        time
```

Benchmark results



VMC Algorithm applied to the 4He nucleus scheme

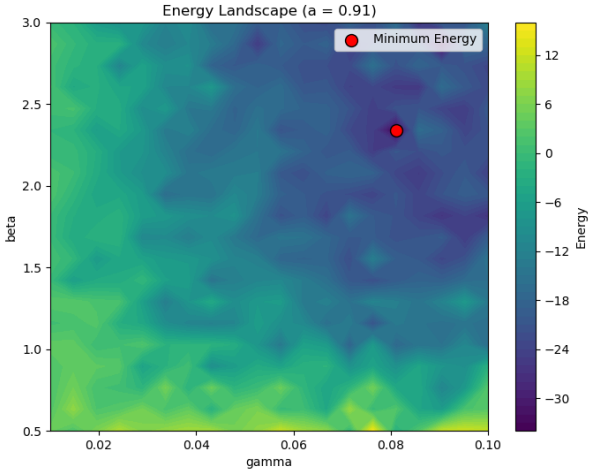
- Sample positions using the Metropolis algorithm
- Use sampled positions to compute $\Psi(x)$ for given (a, β, γ) and local energies.
- Average over all the energies.
- Repeat the previous steps for many values of (a, β, γ) and find the minimum energy.

parameters optimization

```
# Range of parameters
a_values = np.linspace(0.1, 1.0, 20)
beta_values = np.linspace(0.5, 3.0, 20)
gamma_values = np.linspace(0.01, 0.1, 20)

# Optimizing alpha and computing the minimum local energy
for a in a_values:
    for beta in beta_values:
        for gamma in gamma_values:
            mean_energy, std_error = variational_monte_carlo(n_particles=4,
                n_steps=1000, thermalization_steps=300, a=a, beta=beta, gamma
                =gamma)
            energy_results.append((a, beta, gamma, mean_energy))
            if mean_energy < best_energy:
                best_energy = mean_energy
                best_params = (a, beta, gamma)
```

Results for $(a, \beta, \gamma) = (0.905, 2.342, 0.081)$ and Estimated ground state energy: $-33.89 MeV$



Bibliography

- Rafael Guardiola: Monte Carlo Methods in Quantum Many-Body Theories
- https://github.com/Damiano-Santoferrara/Variational-Monte-Carlo-Exercises

Thank you for listening !

Damiano Santoferrara

damiano.santoferrara@studio.unibo.it