Qutip hamiltonian simulation as Qiskit backend for QAOA

Jiakai Wang[1]

[1] *Department of Physics, University of Wisconsin-Madison, Madison, WI 53562, USA*

(Dated: December 12, 2024)

We present the demonstration of a custom quantum circuit simulator that bridges the gap between ideal circuit-level quantum operation definition and realistic Hamiltonian-level pulse-based quantum computations for neutral atom quantum computers. Our implementation extends Qiskit's BackendV1 interface through two main components: MyQiskitSimulatorBase+MyQiskitPulseSimulator, which handle the core simulation logic and measurement statistics, and minimal_processor, which incorporates physical noise models and pulse-based Hamiltonian simulations. In Hamiltonian simulation, we consider two dominant error channels which are 1) Rydberg decay characterized by γr and branching ratios, and 2) control error, characterized limited-precision rotation angles. This well-interfaced software enables direct comparison between ideal circuit execution and more realistic pulse-level implementations, providing valuable insights into the impact of physical constraints and noise on quantum algorithm performance. The simulator uses the gateset (RZ, RX, CZ) for universality and can easily be extended in the future. We present using this simulator for solving a QAOA problem, and briefly discuss how to adapt the code to more recent Qiskit versions.

I. INTRODUCTION

Hamiltonian-level simulation combined with the ability to take in Qiskit circuit description can be a powerful tool for tasks like algorithm error analysis and realistic quantum error correction simulation.

We organize this paper as follows: We begin by describing our implementation of fundamental quantum gates in Section II, followed by the detailed architecture of our simulator in Section III. To validate our implementation, we present extensive benchmarking results in Section IV. Finally, we demonstrate the practical utility of our simulator through a concrete application: implementing and analyzing the Quantum Approximate Optimization Algorithm (QAOA) in Section V.

II. BASIS GATES

A. RX gate

The RX gate implementation is based on the Hamiltonian evolution of a two-level system. The gate Hamiltonian is given by:

$$H_{RX} = \frac{\Omega_{01}}{2}(|0\rangle\langle 1| + |1\rangle\langle 0|) \tag{1}$$

where Ω_{01} represents the coupling strength between the ground state $|0\rangle$ and excited state $|1\rangle$. The time evolution under this Hamiltonian is computed using the master equation:

$$\dot{\rho} = -i[H_{RX}, \rho] \tag{2}$$

The evolution time $t = \theta/10$ is chosen to achieve the desired rotation angle θ (10 is an arbitrary number). This implementation allows for precise control of single-qubit rotations around the X-axis while accounting for the underlying physical dynamics of the system.

To optimize the coupling strength parameter Ω_{01}, we employ a fidelity-based optimization approach. We define a set of target states corresponding to different rotation angles $\{\frac{\pi}{2}, \frac{\pi}{4}, \frac{\pi}{6}\}$:

$$|\psi_{\text{target}}(\theta)\rangle = R_x(\theta)|0\rangle \tag{3}$$

The optimization objective function is defined as the negative average fidelity between the simulated final states and their corresponding target states:

$$f(\Omega_{01}) = -\frac{1}{N}\sum_{\theta} F(\rho_{\text{final}}(\Omega_{01}, \theta), |\psi_{\text{target}}(\theta)\rangle) \tag{4}$$

where F denotes the quantum state fidelity and N is the number of test angles. The optimal value of Ω_{01} is obtained through numerical minimization of this objective function using the Scipy optimization toolkit. We obtained optimized $\Omega_{01} = 9.9994$.

B. RZ gate

The RZ gate is implemented through a phase accumulation Hamiltonian acting on the excited state. The gate Hamiltonian takes the form:

$$H_{RZ} = \delta_1 |1\rangle\langle 1| \tag{5}$$

where δ_1 represents the energy detuning of the excited state. This Hamiltonian generates phase rotations through the time evolution:

$$\dot{\rho} = -i[H_{RZ}, \rho] \tag{6}$$

Unlike the RX gate optimization which uses the ground state as the initial state, the RZ gate optimization employs a superposition state:

$$|\psi_{\text{init}}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (7)$$

The target states for optimization are defined using QuTip-qip's definition of $R_z(\theta)$:

$$|\psi_{\text{target}}(\theta)\rangle = R_z(\theta)\frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad (8)$$

for $\theta \in \{\frac{\pi}{2}, \frac{\pi}{4}, \frac{\pi}{6}\}$. The optimization objective function is similarly defined as:

$$f(\delta_1) = -\frac{1}{N}\sum_{\theta} F(\rho_{\text{final}}(\delta_1, \theta), |\psi_{\text{target}}(\theta)\rangle) \quad (9)$$

where the evolution time is set to $t = \theta/10$ for each rotation angle θ (10 is an arbitrary number). The optimal detuning parameter δ_1 is then obtained through numerical minimization to be -9.998.

C. CZ gate

We use the CZ gate from [1]. This protocol utilizes Rydberg interactions between atoms, with a total protocol duration of $t_{\text{tot}} = 0.540\,\mu$s. The Hamiltonian includes time-dependent controls and strong Rydberg-Rydberg interactions:

$$H(t) = \sum_{i=1}^{2}\left[\frac{\Omega(t)}{2}(|r\rangle_i \langle 1|_i + |1\rangle_i \langle r|_i) + \Delta(t)\,|r\rangle_i \langle r|_i\right] + B\,|rr\rangle \langle rr| \quad (10)$$

where $|r\rangle$ represents the Rydberg state, $B = 2\pi \times 200\,\text{MHz}$ is the Rydberg interaction strength, and the time-dependent Rabi frequency $\Omega(t)$ is given by:

$$\Omega(t) = \Omega_{\max} f(t) \quad (11)$$

The detuning $\Delta(t)$ follows a cosine modulation with an amplitude of $\Delta_{\max} = 2\pi \times 23\,\text{MHz}$, and switches sign at the midpoint of the protocol:

$$\Delta(t) = \begin{cases} -\Delta_{\max}\cos(2\pi t/t_{\text{tot}}) & t < t_{\text{tot}}/2 \\ \Delta_{\max}\cos(2\pi t/t_{\text{tot}}) & t \geq t_{\text{tot}}/2 \end{cases} \quad (12)$$

The pulse shape function $f(t)$ is designed as a double Gaussian pulse:

$$f(t) = \begin{cases} \frac{e^{-(t-t_{\text{tot}}/4)^4/\tau^4} - a}{1-a} & t < t_{\text{tot}}/2 \\ \frac{e^{-(t-3t_{\text{tot}}/4)^4/\tau^4} - a}{1-a} & t \geq t_{\text{tot}}/2 \end{cases} \quad (13)$$

where $\tau = 0.175 t_{\text{tot}}$, $a = e^{-(t_{\text{tot}}/4)^4/\tau^4}$, $\Omega_{\max} = 2\pi \times 17\,\text{MHz}$, and $\Delta_{\max} = 2\pi \times 23\,\text{MHz}$. The sign function $g(t)$ alternates between -1 and 1 at $t_{\text{tot}}/2$.

The complete CZ operation is implemented as a sequence:

$$\text{CZ} = R_z^{(2)}(\pi) \cdot U_{\text{Rydberg}}(t_{\text{tot}}) \cdot R_z^{(1)}(\pi) \quad (14)$$

where $R_z^{(i)}(\pi)$ represents a π phase rotation on qubit i, and U_{Rydberg} is the evolution under the Rydberg Hamiltonian. The local ration is to ensure the overall CZ gate unitary be consistent with the common definition, instead of $diag[1, -1, -1, -1]$. The numerical evolution is performed using QuTiP's `mesolve` with 10,000 time steps to ensure accurate resolution of the fast dynamics.

III. ASSEMBLING THE SIMULATOR

Our simulator implementation consists of three main components: MyQiskitSimulatorBase, MyQiskitPulseSimulator, and minimal_processor. The base class MyQiskitSimulatorBase and the wrapper MyQiskitPulseSimulator extends Qiskit's BackendV1 interface, providing essential functionality for quantum state manipulation and measurement statistics. It handles shot sampling and result formatting, ensuring compatibility with standard Qiskit workflows.

The minimal_processor class implements the physical model of the quantum system for pulse-level simulation. It defines a four-level system for each qubit (ground, excited, leakage, and Rydberg states) and implements the Hamiltonian evolution for each gate operation. The processor supports three basis gates: RX, RZ, and CZ, each implemented through optimized pulse-level control sequences as described in Section II. The processor also includes a framework for modeling decoherence through collapse operators, characterized by the relaxation rate γ_r.

The simulator accepts standard Qiskit circuits as input, which are first transpiled into the supported basis gate set. Each gate is then executed through Hamiltonian evolution using QuTiP's master equation solver. This approach bridges the gap between circuit-level quantum algorithms and their physical implementation, enabling detailed analysis of realistic quantum computation effects.

A. Sanity test

To validate our pulse-level simulator, we performed comparative tests against an ideal circuit simulator using three test circuits of increasing complexity as shown in FIG. 1:

For each circuit, we compared our pulse simulator against Qiskit's BasicSimulator through two metrics:

- State fidelity: $F(\rho_{\text{pulse}}, \rho_{\text{ideal}})$

```
circ1 = QuantumCircuit(2)
circ1.h(0)
circ1.h(1)

circ2 = QuantumCircuit(2)
# circ2.h(0)
circ2.rx(0.01, 1)
circ2.h(1)
circ2.rz(0.01, 1)

circ3 = QuantumCircuit(2)
circ3.rz(0.5, 0)
circ3.rx(1.0, 1)
circ3.cz(0, 1)

circ3.rz(1.2, 1)
circ3.rx(0.8, 0)
circ3.cz(1, 0)

circ3.rz(0.3, 0)
circ3.rx(1.5, 1)
circ3.cz(0, 1)

circ3.rz(0.7, 1)
circ3.rx(1.1, 0)
circ3.cz(1, 0)
```

FIG. 1. Circuits for sanity test

- Statistical distance: $\sqrt{\sum_s (P_{\text{pulse}}(s) - P_{\text{ideal}}(s))^2 / N^2}$

where $P(s)$ represents the probability of measuring bit string s, and N is the number of qubits. This dual comparison allows us to validate both the quantum state evolution and measurement statistics of our simulator. The results we see is that our simulator (with zero decay rate) is highly consistent with the ideal Qiskit simulator.

IV. RANDOMIZED BENCHMARKING

We implemented Qiskit's StandardRB protocol to assess the performance of our pulse-level simulator. The benchmarking sequence was configured as follows:

$$\text{lengths} = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] \tag{15}$$

with 10 samples per length. Each sequence was transpiled to our basis gate set:

$$\{\text{RX}, \text{RZ}, \text{CZ}\} \tag{16}$$

The results revealed that our transpiled circuits contained significantly more gates than the original sequences, after compiling to these basis gates:

```
['cz', 'rx', 'rz', 'measure', 'rest']
```

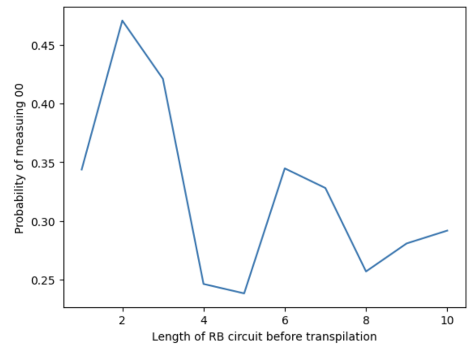

FIG. 2. RB result

The probability of measuring the $|00\rangle$ state was plotted against before-compilation gate sequence length in FIG. 2 showing a wierd pattern. The wierd behavior of fidelity with as a function of before-compilation gate sequence length suggests that optimization of the transpilation process could be crucial for improving the simulator's performance in complex quantum circuits.

Due to the computational cost of Hamiltonian simulation, we gave up fixing the RB experiment.

V. QAOA IMPLEMENTATION

The Quantum Approximate Optimization Algorithm (QAOA) alternates between two Hamiltonians: the problem Hamiltonian H_P and the mixer Hamiltonian H_M. For the MaxCut problem on a graph $G = (V, E)$, these Hamiltonians take the form:

$$H_P = \sum_{(i,j)\in E} \frac{1}{2}(I - Z_i Z_j) \tag{17}$$

$$H_M = \sum_{i\in V} X_i \tag{18}$$

The QAOA circuit implements a sequence of unitary operations:

$$|\psi(\vec{\beta}, \vec{\gamma})\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_P} \cdots e^{-i\beta_1 H_M} e^{-i\gamma_1 H_P} |+\rangle^{\otimes n} \tag{19}$$

where $|+\rangle^{\otimes n}$ is the initial state created by applying Hadamard gates to $|0\rangle^{\otimes n}$. The parameters $\vec{\beta}$ and $\vec{\gamma}$ are optimized to maximize the expectation value:

$$\langle H_P \rangle = \langle \psi(\vec{\beta}, \vec{\gamma})| H_P |\psi(\vec{\beta}, \vec{\gamma})\rangle \tag{20}$$

In our implementation, the problem Hamiltonian evolution is realized through RZZ gates:

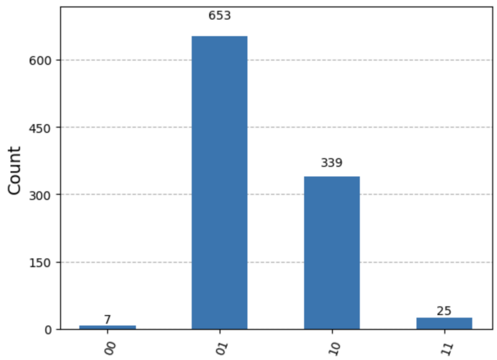

FIG. 3. QAOA measurement with optimized parameters

$$e^{-i\gamma H_P} = \prod_{(i,j)\in E} e^{-i\gamma Z_i Z_j/2} \tag{21}$$

while the mixer Hamiltonian evolution is implemented using RX gates:

$$e^{-i\beta H_M} = \prod_{i\in V} e^{-i\beta X_i}. \tag{22}$$

After optimizing the QAOA parameters, we find the two output with the highest amplitudes correspond to maxcut solutions.

VI. CONCLUSION AND OUTLOOK

During the software lab, I tried to use the newer BackendV2 of updated Qiskit. The main compatibility issue of the current implementation with BackendV2 is that BackendV2 our measurement scheme doesn't work anymore. Future work could continue on my partially completed simulator using BackendV2 where I've figured out how to setup the Target attribute of a backend.

[1] M. Saffman, I. I. Beterov, A. Dalal, E. J. Páez, and B. C. Sanders, Symmetric rydberg controlled-z gates with adiabatic pulses, Phys. Rev. A **101**, 062309 (2020).