Simulating the evolution of a Heisenberg model Hamiltonian for a three-particle system on IBM Quantum's 7-qubit Jakarta system: *our approach to IBM Quantum Awards: Open Science Prize 2021 using Qiskit defaults*

André J. Ferreira–Martins,[1] Askery Canabarro,[2] Anton S. Albino,[3] Rafael Chaves,[4,5] and Rodrigo Pereira[4]

[1] *Itaú-Unibanco SA, Praça Alfredo de Souza Aranha 100, Torre Setúbal Parque, 04344-902, Sao Paulo-SP, Brazil*

[2] *Grupo de Física da Matéria Condensada, Núcleo de Ciências Exatas - NCEx, Campus Arapiraca, Universidade Federal de Alagoas, 57309-005, Arapiraca-AL, Brasil*

[3] *Latin American Quantum Computing Center/Senai Cimatec, Salvador-BA, Brasil*

[4] *Instituto Internacional de Física, Universidade Federal do Rio Grande do Norte, 59078-970, P. O. Box 1613, Natal-RN, Brasil*

[5] *School of Science and Technology, Federal University of Rio Grande do Norte, 59078-970 Natal, Brazil*

The IBM Quantum Awards: Open Science Prize 2021 problem: to simulate the evolution of a Heisenberg model Hamiltonian for a three-particle system on IBM Quantum's 7-qubit Jakarta system. The goal is to simulate the evolution of a known quantum state with the best fidelity as possible via Trotterization, either using use Qiskit Pulse or Qiskit defaults. In the pursuit of an optimal solution, we considered the interplay of three major ingredients: (i) the order of the Trotter-Suzuki decomposition, (ii) the number of steps, and (iii) the time duration for each step. As will be detailed below, after simulating a number of scenarios we focused on first and second order Trotter decompositions only. We tried different numbers of trotterization steps ranging from 4 (the minimum allowed) up to 8. One key ingredient of our approach was to optimize the duration of each trotterization step in order to achieve the highest fidelity at $t = \pi$ using the Sequential Least SQuares Programming (SLSQP) optimizer. We chose to solve the problem using Qiskit defaults.

Keywords: Quantum Computing, Quantum Simulation, Trotterization.

I. THE PROBLEM, THE GOAL AND OUR APPROACH

The IBM Quantum Awards: Open Science Prize 2021 challenge features an important problem from the field of quantum simulation. Simulating physical systems on quantum computers is a promising application of quantum processors in the NISQ era, as one can represent quantum states (e.g., the spin states) of particles as the computational states of qubits.

The problem proposed in the challenge is the simulation of the evolution of a Heisenberg model Hamiltonian for a three-particle system on IBM Quantum's 7-qubit Jakarta system. The goal is to simulate the evolution of a known quantum state with the best fidelity as possible via Trotterization and employing the tools of either Qiskit Pulse or Qiskit defaults to implement the algorithm to be run in the Jakarta system.

In our solution, we explored the interplay of three major ingredients: the order of the Trotter-Suzuki decomposition; the number of steps, and the time duration for each step, not taken to be uniform. As will be detailed below, after simulating a number of scenarios we focused on first and second order Trotter decompositions only. We tried different number of trotterization steps ranging from 4 to 8. One key ingredient of our approach was to optimize the duration of each trotterization step in order to achieve the highest fidelity at $t = \pi$ using the Sequential Least SQuares Programming (SLSQP) optimizer. This places our solution within the variational/parameterized quantum circuits landscape. We solved the problem using Qiskit defaults. Please check our GitHub Repository for a detailed implementation of our solution [1].

II. THE TROTTER-SUZUKI DECOMPOSITION

We propose a Trotter-Suzuki decomposition with different times for each step, provided that the time steps sum up to the total evolution time ($t = \pi$). That is, we write the trotterized operator $U_{\text{Heis3}}(t)$ as

$$U_{\text{Heis3}}(t) \approx \prod_{i=1}^{n} U_i^{(k)}(t_i), \tag{1}$$

where $U_i^{(k)}(t_i)$ represents the i-th trotterization step, using a trotter decomposition of k-th order; and $n \geq 4$ is the number of steps. We considered Trotter decompositions of orders $k = 1, 2$.

A. First-order trotterization ($k = 1$)

The unitary operator for the first-order trotterization is given by

$$U_i^{(1)}(t_i) = XX(2t_i)^{(1,2)}YY(2t_i)^{(1,2)}ZZ(2t_i)^{(1,2)}XX(2t_i)^{(0,1)}YY(2t_i)^{(0,1)}ZZ(2t_i)^{(0,1)}. \tag{2}$$

Fig. 1 represents the quantum circuit of a single trotter step for the first-order trotterization described in Eq. 2. The competition rules requires at least four steps, regardless of the trotterization order.

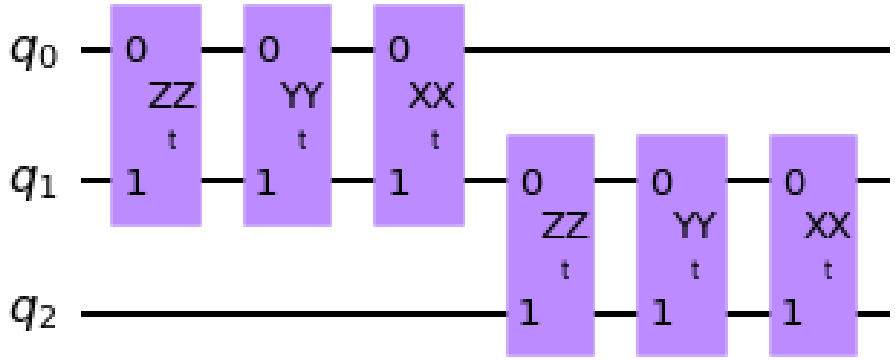

Figure 1. Representation of a single trotter step circuit for 1st order trotterization.

B. Second-order trotterization ($k = 2$)

For the second-order trotterization, the operator is given by

$$U_i^{(2)}(t_i) = XX(t_i)^{(0,1)}YY(t_i)^{(0,1)}ZZ(t_i)^{(0,1)}XX(2t_i)^{(1,2)}YY(2t_i)^{(1,2)}ZZ(2t_i)^{(1,2)}XX(t_i)^{(0,1)}YY(t_i)^{(0,1)}ZZ(t_i)^{(0,1)}. \tag{3}$$

Fig. 2 represents the quantum circuit of a single trotter step for the second-order trotterization described in Eq. 3.

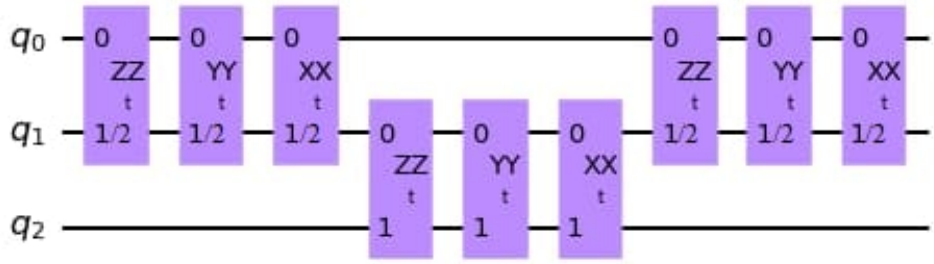

Figure 2. Representation of a single trotter step circuit for 2nd order trotterization. In this figure, the "1/2" indicates that the parameters of the gates acting on the first two qubits are halved. However, in our code, the circuit is simply drawn with the label t in all gates (instead of $t/2$), because a single parameter vector was used, which makes the code more concise. But notice that in the code this parameter is halved, to correctly match Eq. 3, so that this is only a matter of visualization, and the actual quantum circuit is correct.

The superscripts $(0, 1)$ and $(1, 2)$ in Eqs. 2 and 3 indicate, respectively, that the operator is acting on the first/second and second/third qubit pairs.

C. Non-uniform time-decomposition steps and optimization

Following a first-discretize-then-optimize approach, for any trotterization order, the total evolution time is decomposed into n steps of duration t_i, such that $\sum_{i=1}^{n} t_i = t$.

One important aspect of our approach is that we do not assume a fixed interval, such as $t_i = t/n$, but rather allow for non-uniform times, as long as the total evolution time corresponds to the target time $t = \pi$. This was a key ingredient in our approach, as will be clear ahead.

Now, we treated the step times t_i as free parameters, such that the full troterrization circuit is seen as a variational circuit. The goal then becomes to find the best parameter values, subject to the $\sum_{i=1}^{n} t_i = t$ constraint.

With that aim, we used the fidelity at $t = \pi$ as the loss function to be optimized with SciPy's implementation of the Sequential Least Squares Programming (SLSQP) for constrained optimization.

SLSQP minimizes a function of several variables with constraints [2]. Given a number of steps n, say $n = 4$, we minimize the negative of fidelity (a natural cost function because we we want to maximize the fidelity) in function of an array of time steps, say $[t_1, t_2, t_3, t_4]$ with the constraint that the time evolution sums up to π, where t_i represents the time duration of the i-th trotter step. We also explicitly impose the restriction that the individual t_i must be non-null, so that the respective step is effectively present in the circuit, i.e. $t_i > \epsilon$, with $\epsilon > 0$. We can then play with the optimization performance in function of ϵ, as shown below.

For this, we used the SciPyOptimizer, please check [2] for an overview of the optimizer and its main parameters. Seeking to reduced local minima/barren plateaus issues, we tried three step size (*eps*) values used for numerical approximation, $eps = [0.1, 0.01, 0.001]$, taking the best (lowest cost function) as our optimum time schedule. More details can be found by inspecting the file trotter_utils.py, which contains the definition of all the functions constructed for this process.

Since the optimization of the variational circuit demands a large number of circuit executions, it was done using a simulator. However, to better capture Jakarta's behavior, we used its noise profile for the simulation. That is, the fidelity used as the loss function was obtained from a noisy simulation, an approach that made the optimization process harder, but certainly more realistic.

III. RESULTS AND DISCUSSIONS

Before we dive deeper in our results, it is worth mentioning that the best simulated values may not be exactly reproducible due to IBM calibration events, meaning that executions of the same code in diverse dates may differ [3]. In fact, our best results were compiled from jobs executed by different members of the team in different dates and we are using the best values achieved.

Interestingly, by imposing the constraint that all times must be positive, in many cases the optimization process yielded null step times for most of the parameters, leaving only two non-zero times, equally divided as $\frac{\pi}{2}$ each. This would correspond to effectively performing only two steps. Therefore, to fulfill the requirements of the challenge, this is why we imposed that all steps must be non-zero, by requiring that $t_i > \epsilon$, with $\epsilon > 0$.

Some of our first best results were obtained setting $\epsilon = 0.01$, which was indeed picked as the values for most of the parameters. Thus, for any chosen number of trottrerization steps n, all such steps are effectively present and performed by the quantum circuit, satisfying the conditions imposed by the challenge at hand.

Inspired by the fact that smaller time steps could provide better solutions, we also tested $\epsilon = 0.001$ e $\epsilon = 0.0005$, for different number of steps (n), again with $n \in [4, 5, 6, 7, 8]$. In total, we tried four values of $\epsilon \in [0.1, 0.01, 0.001, 0.0005]$. We decided not to further decrease the minimum time duration, ϵ, beyond the forth decimal place because it would be smaller than the precision we provide for the computation of the fidelity of the target state at π. As already mentioned, we tested only first and second order trotterization.

Overall, we studied ans experimented with 40 different combinations of k, n, ϵ. A summary of our simulated investigations versus hardware execution is given in the Table II. For each tuple (k, n, ϵ) we provide the mean fidelity, the best fidelity and the corresponding standard deviation (std.) of 8 hardware executions, as well as the mean simulated fidelity (also with its std.) along with the best parameters for the time step schedule. The table is sorted in descending order with respect to mean of the simulated fidelity in the fake jakarta backend.

It is clear that the real executions provide worst results than the simulated ones. As can be seen, both for the real executions on Jakarta's backend and its simulator, the best results are achieved considering $k = 1$ (first-order trotterization) and $n = 4$ (the minimum allowed number of steps).

In two of the cases, shown in boldface, we could exceed the minimum required of 0.3 for the fidelity at $t = \pi$ both for the best of the individual experiment and the mean of the 8 hardware executions. The best individual and mean fidelities achieved were respectively 0.387 and 0.324 for the hardware execution, with $k = 1$, $n = 4$ and $\epsilon = 0.001$.

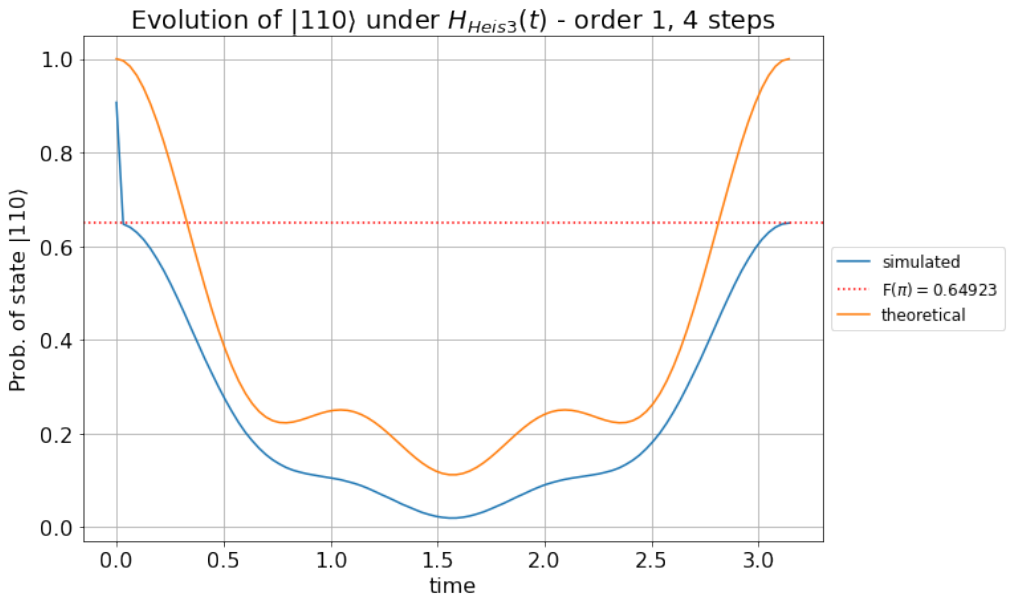

Figure 3. Evolution of the state $|110\rangle$ under $U_{\text{Heis3}}(t)$ for a first-order trotterization with 4 steps.

As aforementioned, we performed the variational circuit optimization by using only information of $t = \pi$ in the cost function. However, we wondered if it was possible to capture the whole dynamics even with such limited information. In other words, we wanted to know if our solution made complete sense. And, quite remarkably, it was possible to recover the whole dynamics by the following procedure:

- We first determined the best parameters t_i by optimizing the loss calculated at $t = \pi$;
- This yields a n-dimensional parameters vector $\vec{t}$, whose components sum to $t = \pi$;
- We then normalized this vector, so that its components sum to 1, by dividing each component by $t = \pi$, $\vec{t}_{\text{norm}} = \frac{1}{t}\vec{t}$. The components of this normalized vector indicates the percentual contribution of each step duration to the whole evolution;
- We produced 100 intermediate times between 0 and π. The normalized vector was then used to parametrize the evolution using each one of the intermediate times as the target time t_{target}, i.e., keeping the same proportion found by optmization at $t = \pi$, and making the parameter vector sum to t_{target}: $\vec{t}_{\text{target}} = t_{\text{target}}\vec{t}_{\text{norm}}$;
- The fidelity of the target state was then measured at each one of these intermediate target times.

And indeed, we can see that the dynamics was correctly captured for all the intermediate steps as shown in Fig. 3.

Thus, by learning the proportional contribution of each step to the total evolution with information only available at $t = \pi$, we were able to reconstruct, qualitatively and quantitatively, the full dynamics. This is a quite interesting result.

However, this reconstruction is, in general, not possible when null time step is allowed (e.g. $[0, 0, 0, \pi]$, representing a simple rotation). Therefore, our imposition of non-null time, besides of better complying the rules, avoids these trivial solutions. In the folder 'figs' in our full solution material and repository, more evolution plots are presented.

As the results mentioned above showed an accentuated discrepancy between the hardware and the simulation, we considered yet another possibility for the constraints on the parameters values: instead of imposing only positive, non-null step times, we allowed negative values for these parameters.

The procedure is similar to one described above. However, we relax the constraints and allow negative sign values for the parameters, so that $t_i \in (-\pi, \pi)$. Again, the constraint is that sum has to add up to π. Of course, only solutions with more than 4 non-null terms should be considered a valid solution. Furthermore, negative "times" in the decomposition may be taken to be, for instance, counterclockwise rotations, given that, in the end, the "times" are merely rotation angles.

Now, despite the discussion above, we see that this approach of allowing negative times is situated in a gray area of compliance with the rules of the competition. Given that, the results are only briefly mentioned, but for the sake

k	n	ϵ	Best Fid.	Mean Fid.	Mean Sim. Fid.	Best time schedule
1	**7**	**-π**	**0.6107**	**0.4459 ± 0.0995**	**0.7369 ± 0.0015**	**[-3.1416, 3.1416, -0.007, 3.1416, 3.1416, -3.1416, 0.007]**
1	**5**	**-π**	**0.4096**	**0.3908 ± 0.0142**	**0.6719 ± 0.0018**	**[0.0148, 3.1416, 1.556, 1.570, -3.1416]**
1	8	-π	0.2691	0.2637 ± 0.0045	0.5107 ± 0.0019	[-0.787, 3.1416, 2.360, 2.353, -2.350, -3.1416, -0.796, 2.362]
1	4	-π	0.2844	0.2583 ± 0.0228	0.6149 ± 0.0016	[-5.29e-03, 1.576, 1.570, 2.05e-04]
1	6	-π	0.2361	0.2198 ± 0.0134	0.5641 ± 0.0012	[-2.30e-03, 3.1416, 1.576, 1.568, -1.568, -1.573]
2	5	-π	0.1784	0.1705 ± 0.0050	0.4962 ± 0.0026	[-3.1416, 3.1416, 1.580, -0.004, 1.566]
2	7	-π	0.1403	0.1362 ± 0.0032	0.3729 ± 0.0016	[0.006, 3.1416, 3.1416, 1.575, -1.612, -1.126, -1.986]
2	4	-π	0.1400	0.1354 ± 0.0033	0.5166 ± 0.0026	[-1.573, 1.576, 1.570, 1.569]
2	8	-π	0.1395	0.1327 ± 0.0043	0.3727 ± 0.0017	[-0.006, 3.1416, 1.568, 1.585, -1.550, -1.597 , -3.1416, 3.1416]
2	6	-π	0.1352	0.1280 ±0.0034	0.4274 ± 0.0022	[0.0048, 3.1416, 1.572, 1.5638, -1.564, -1.576]

Table I. A summary of our simulated investigations versus hardware execution with negative parameters allowed. The columns descriptions are the same as in Table II.

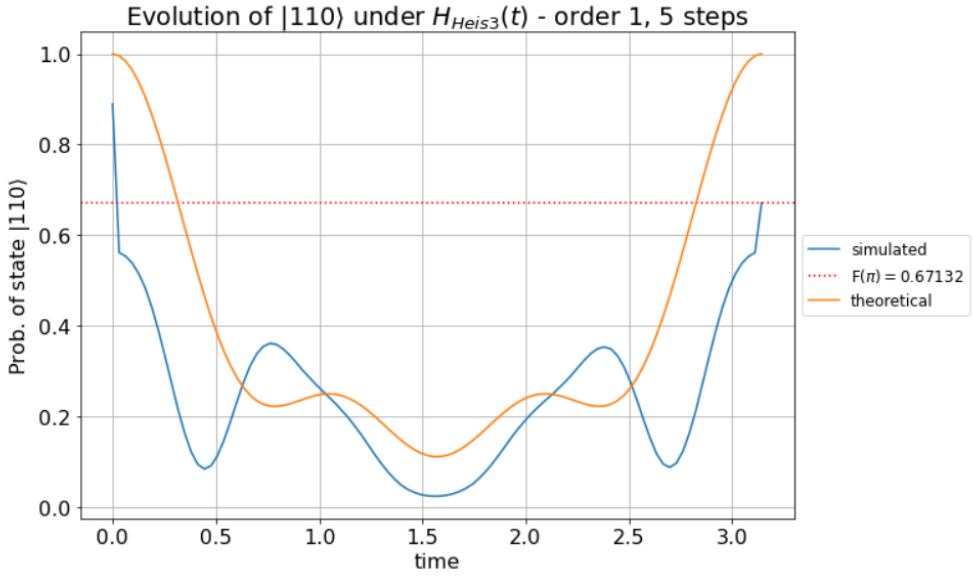

Figure 4. Evolution of the state $|110\rangle$ under $U_{\text{Heis3}}(t)$ for a first-order trotterization with 5 steps and allowed negative times.

of completeness of our approach, we decide to include them here, as well as in the code material. Please check Table I for a quick overview.

Again, in two cases we got results which beat the minimum fidelity required. Again only for first order, but with 7 and 5 trotterization steps, not 4 steps, as was observed when we imposed positive, non-negative parameters.

Now, quite interestingly, we achieved much better results with negative times. The mean fidelity with hardware execution reached 0.4459 for the best scenario and more than 0.61 for an individual run. It is worth mentioning that the time schedule never returned null times, although we observed parameters which where almost perfect opposites.

Also interestingly, in both these cases ($n = 5$ and $n = 7$ steps), the overall dynamics was relatively well reconstructed, as shown in Figs 5 and 4. This indicates that what happens when one allows for negative times is not the same that was observed when we had a single non-null step size of π. Although in a grey area, we see this as an interesting result, worthy further exploration.

Given the nature of our approach, it may be argued that the resulting trotterization unitary is very close to the identity, in both cases: the ones in which the trotterization steps are quite small, as well as in the ones in which we allow for negative parameters (which converged to parameters with opposite values). Naturally, the identity would yield a quite high fidelity at $t = \pi$, given the periodicity of the problem. Now, the reconstruction of the full dynamics

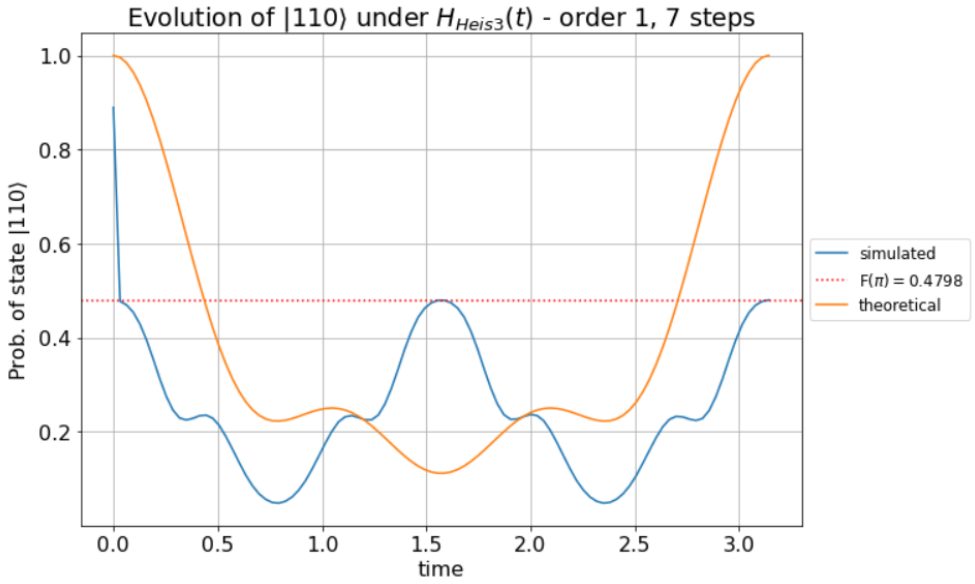

Figure 5. Evolution of the state $|110\rangle$ under $U_{\text{Heis3}}(t)$ for a first-order trotterization with 7 steps and allowed negative times.

(as shown in Figs. 3, 4 and 5, and all others in the "figs/" folder) prove that our circuit is not implementing the identity (otherwise, we would expect a constant high fidelity throughout the full evolution, i.e., for all $t \in (0, \pi)$.

But in order to check this fact more quantitatively, we construct the explicit unitary (using Qiskit's unitary simulator) for each one of the experiments, and compared them to the identity, via the Frobenius norm of their difference, $\|U_{\text{Heis3}} - \mathbb{1}\|_F = \left(\sum_{ij} |(U_{\text{Heis3}} - \mathbb{1})_{ij}|^2\right)^{1/2}$. We considered the matrices to be equal if this norm is smaller than a cuttof of 0.0001. For the 40 experiments, only 5 combinations met this criteria, which means that for the vast majority, the resulting trotterized unitary is significantly different from the identity operator.

All the results described above can be found consolidated in a table in the "final_results_all_experiments.parquet", which is in the "results/" folder. All the results are consolidated in the "final_results_analysis.ipynb" notebook. Please refer to these files for further details.

In summary, we believe that our approach shed new light on how to implement a variational-aided Trotter-Suzuki expansion of a given Hamiltonian. Although we only explicitly explored the Hamiltonian proposed in the challenge, our approach is sufficiently generic to be applied to other Hamiltonians, which shall be done in future studies.

The limitations of scalability of the parameters optimization procedure with the number of steps or the order of the trotterization (in virtue of the availability of scalable quantum computational resources) can be satisfactorily circumvented by the use of a simulator, as we have shown above. Indeed, the usage of both classical and quantum computing resources in hybrid routines, characteristic of NISQ era quantum computing, has been show to be a fruitful approach for a myriad of problems. With our project, we present yet another example of a successful application of this approach.

---

[1] GitHub Repository with our solution, https://github.com/askery/Ibm_openprize_2022_qtime/ .
[2] Sequential Least SQuares Programming optimizer, https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.SLSQP.html/.
[3] System properties of IBM Quantum, https://quantum-computing.ibm.com/lab/docs/iql/manage/systems/properties.

k	n	ϵ	Best Fid.	Mean Fid.	Mean Sim. Fid.	Best time schedule
1	**4**	**0.001**	**0.3871**	**0.3236 ± 0.0556**	**0.6494 ± 0.0014**	**[0.001, 1.573, 0.001, 1.573]**
1	**4**	**0.0005**	**0.3485**	**0.3098 ± 0.0405**	**0.6495 ± 0.0019**	**[0.0005, 1.574, 0.0005, 1.567]**
1	5	0.0005	0.3621	0.2864 ± 0.0657	0.5999 ± 0.0013	[1.566, 1.574, 0.0005, 0.0005, 0.0005]
1	5	0.001	0.3098	0.2834 ± 0.0127	0.6014 ± 0.0023	[0.002, 1.566, 1.572, 0.001, 0.001]
1	4	0.01	0.3386	0.2438 ± 0.0538	0.5981 ± 0.0024	[0.01, 1.555, 0.01, 1.567]
1	4	0.1	0.2528	0.2319 ± 0.0110	0.5872 ± 0.0012	[0.1, 1.448, 0.1, 1.494]
1	6	0.001	0.2334	0.2098 ± 0.0285	0.5569 ± 0.0019	[0.001, 1.571, 1.567, 0.001, 0.001, 0.001]
1	6	0.0005	0.2331	0.2226 ± 0.0079	0.5563 ± 0.0020	[4.46e-03, 1.570, 1.566, 9.11e-04, 0.0005, 0.0005]
1	5	0.01	0.1780	0.1724 ± 0.0047	0.5534 ± 0.0023	[0.01, 1.563, 1.549, 0.01, 0.01]
2	4	0.0005	0.1551	0.1497 ± 0.0044	0.5475 ± 0.0025	[0.0005, 0.0005, 3.138, 2.17e-03]
2	4	0.001	0.1504	0.1399 ± 0.0057	0.5459 ± 0.0014	[0.001, 1.573, 1.565 , 2.27e-03]
1	5	0.1	0.1831	0.1767 ± 0.0043	0.5208 ± 0.0019	[0.1, 1.424, 1.418, 0.1, 0.1]
1	7	0.0005	0.1873	0.1737 ± 0.0122	0.5180 ± 0.0015	[1.6e-03, 1.571, 1.567, 0.0005, 0.0005, 0.0005, 0.0005]
1	7	0.001	0.1636	0.1224 ± 0.0194	0.5162 ± 0.0025	[0.001, 1.570, 1.566, 0.001, 0.001, 0.001, 0.001]
1	6	0.01	0.1658	0.1577 ± 0.0042	0.5100 ± 0.0023	[0.01, 1.553, 1.549, 0.01, 0.01, 0.01]
2	4	0.01	0.1838	0.1756 ± 0.0040	0.4917 ± 0.0020	[0.01, 1.562, 1.559, 0.01]
2	5	0.001	0.1171	0.1106 ± 0.0040	0.4901 ± 0.0019	[0.001, 1.983, 1.155, 0.001, 0.001]
2	5	0.0005	0.1086	0.1035 ± 0.0039	0.4899 ± 0.0020	[0.0005, 2.172, 0.963, 4.76e-03, 0.0005]
1	8	0.001	0.1486	0.1347 ± 0.0068	0.4823 ± 0.0018	[0.006, 1.563, 1.568, 0.001, 0.001, 0.001, 0.001, 0.001]
1	8	0.0005	0.1441	0.1398 ± 0.0031	0.4818 ± 0.0016	[8.38e-04, 1.573, 1.565, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005]
1	7	0.01	0.1655	0.1578 ± 0.0044	0.4731 ± 0.0013	[0.01, 1.559, 1.533, 0.01, 0.01, 0.01, 0.01]
1	6	0.1	0.1594	0.1499 ± 0.0045	0.4544 ± 0.0037	[0.1, 1.382, 1.360, 0.1, 0.1, 0.1]
1	8	0.01	0.1562	0.1500 ± 0.0038	0.4427 ± 0.0023	[0.01, 1.553, 1.529, 0.01, 0.01, 0.01, 0.01, 0.01]
2	6	0.001	0.1105	0.1004 ± 0.0060	0.4399 ± 0.0018	[0.001, 1.567, 1.570, 0.001, 0.001, 0.001]
2	6	0.0005	0.0973	0.0937 ± 0.0032	0.4398 ± 0.0023	[2.33e-03, 1.563, 1.573, 1.01e-03, 1.08e-03, 0.0005]
2	4	0.1	0.1785	0.1554 ± 0.0115	0.4262 ± 0.0029	[0.1, 1.47, 1.468, 0.1]
2	5	0.1	0.1363	0.1322 ± 0.0031	0.4043 ± 0.0018	[0.1, 0.1, 2.742, 0.1, 0.1]
2	5	0.01	0.1645	0.1485 ± 0.010	0.4043 ± 0.0018	[0.01, 2.052, 1.060, 0.01, 0.01]
2	7	0.0005	0.1038	0.0882 ± 0.0090	0.4018 ± 0.0018	[1.08e-03, 1.564, 1.573, 0.0005,1.82e-03,0.0005, 0.0005]
2	7	0.001	0.1093	0.1028 ± 0.0048	0.4010 ± 0.0023	[0.001, 1.568, 1.569, 0.001, 0.001, 0.001, 0.001]
2	6	0.01	0.1031	0.0952 ± 0.0040	0.3961 ± 0.0021	[0.01, 1.550, 1.551, 0.01, 0.01, 0.01]
1	7	0.1	0.1650	0.1551 ± 0.0073	0.3918 ± 0.0026	[0.1, 1.347, 1.294, 0.1, 0.1, 0.1, 0.1]
2	8	0.0005	0.1284	0.1107 ± 0.0131	0.3665 ± 0.0019	[6.56e-04, 3.137, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005, 0.0005]
2	8	0.001	0.1029	0.0950 ± 0.0057	0.3659 ± 0.0018	[1.77e-02, 3.117, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001]
2	7	0.01	0.1145	0.1105 ± 0.0025	0.3620 ± 0.0024	[0.01, 1.547, 1.544, 0.01, 0.01, 0.01, 0.01]
2	8	0.01	0.1173	0.1030 ± 0.0080	0.3388 ± 0.0018	[0.01, 3.072, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01]
1	8	0.1	0.1529	0.1500 ± 0.0024	0.3356 ± 0.0022	[0.1, 1.312, 1.230, 0.1, 0.1, 0.1, 0.1, 0.1]
2	7	0.1	0.1292	0.1247 ± 0.0053	0.3203 ± 0.0020	[0.1, 0.1 , 2.541, 0.1, 0.1, 0.1, 0.1]
2	6	0.1	0.1227	0.1200 ± 0.0021	0.2438 ± 0.0023	[0.1, 1.393, 1.348, 0.1, 0.1, 0.1]
2	8	0.1	0.1136	0.1099 ± 0.0025	0.2266 ± 0.0023	[0.156, 0.374, 0.505, 0.407, 0.373, 0.440, 0.510, 0.378]

Table II. A summary of our simulated investigations versus hardware execution. k is the trotterization order, n is the number of trotter steps, ϵ is the minimum time step allowed. "Best Fid." stands for the maximum fidelity out of 8 hardware executions, "Mean Fid." is the mean fidelity out of 8 hardware executions and the corresponding standard deviation, "Mean Sim. Fid." is the mean fidelity out of 8 fake Jakarta executions and the corresponding std., "Best time schedule" is the best time step array computed with our SLSQP optimization approach. It is worth mentioning that the best simulated values may not be exactly reproducible due to IBM calibration events, meaning that executions of the same code in diverse dates may differ.