

IBM Challenge Progress Report

Linus Ekstrøm

February 2022

IBM Quantum Challenge 2022

- Each team or participant may only contribute to one submission.
- Solution may only be executed on the designated device `ibmq_jakarta`.
- Each submission must use Trotterization to evolve the specified state, under the specified Hamiltonian, for the specified duration with at least 4 Trotter steps.
- Only use Open Pulse and or pulsed gates functionality.
- Only use libraries that can be installed using either `pip install` or `conda install` and no purchased libraries.
- Document code with concise, clear language about the chosen methodology.
- State tomography fidelity (for 4 or more trotter steps) must meet a minimum value of 30%.

Judgement Criteria

- Performance as measured by the state tomography fidelity in comparison to other submissions (Max 15 points).
- Clarity of provided documentation and solution code (Max 5 points).
- Creativity in developing a unique, innovative, and original solution (Max 5 points).

Introducing the Problem

As specified in the competition rules the simulation must utilize Trotterization with no less than 4 Trotter steps. This is to be executed on the IBM Jakarta 7-qubit device. The state tomography fidelity is calculated on qubits number 1, 3, 5 on the device. The other 4 qubits are regarded as trash qubits and can be utilized in any way.

We are told to simulate the evolution of the state $|110\rangle$ under the homogenous Heisenberg 3-particle model up to a final time $t = \pi$

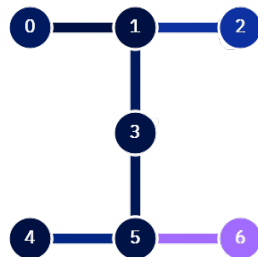


Figure: Jakarta Topology from [IBMQ](#)

Homogenous Heisenberg Model

We will be dealing with the homogenous Heisenberg spin chain model

$$H = \sum_{\langle ij \rangle}^N J \left(\sigma_x^{(i)} \sigma_x^{(j)} + \sigma_y^{(i)} \sigma_y^{(j)} + \sigma_z^{(i)} \sigma_z^{(j)} \right) \quad (1)$$

where $N = 3$ in our case, and we keep the tensor products implicit. Since we are dealing with the homogenous case $J = 1$ and after writing this out we have

$$\begin{aligned} H = & \sigma_x^{(1)} \sigma_x^{(2)} + \sigma_y^{(1)} \sigma_y^{(2)} + \sigma_z^{(1)} \sigma_z^{(2)} \\ & + \sigma_x^{(2)} \sigma_x^{(3)} + \sigma_y^{(2)} \sigma_y^{(3)} + \sigma_z^{(2)} \sigma_z^{(3)} \end{aligned} \quad (2)$$

Molto semplice

The Propagator and the Exact Propagation

The exact propagator is

$$U(t) = e^{-itH} = e^{-it\left(\sum_{\langle ij \rangle}^3 \sigma_x^{(i)} \sigma_x^{(j)} + \sigma_y^{(i)} \sigma_y^{(j)} + \sigma_z^{(i)} \sigma_z^{(j)}\right)} \quad (3)$$

To obtain the exact evolution up to $t = \pi$ we evaluate the inner product for some number of points which yields. We see the evolution has periodicity π . We are able to calculate the exact evolution due to the limited system size. For $N = 3$ our propagator is $(2^3 \times 2^3)$, whereas for $N = 50$ the propagator will be $(2^{50} \times 2^{50})$

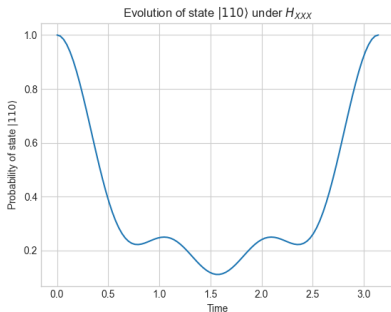


Figure: Exact evolution

Decomposing the Propagator into Quantum Gates

Trotterization

One way of splitting the Hamiltonian is to think of it as two separate two-body Hamiltonians

$$H = H_1 + H_2 \rightarrow U(t) = e^{-it(H_1+H_2)} \quad (4)$$

Because $[H_1, H_2] \neq 0$ we cannot simply split the exponential. To approximate the evolution we can use the first-order Lie-Trotter formula [1]

$$\begin{aligned} U(t) &= e^{-it(H_1+H_2)} \\ &= \left(e^{-i\frac{t}{n}H_1} e^{-i\frac{t}{n}H_2} \right)^n + \mathcal{O}(\delta^2) \end{aligned} \quad (5)$$

Where n is the number of trotter steps, and we have defined the trotter time-step $\delta = t/n$.

Note: *the way of splitting the Hamiltonian is not unique.*

Decomposing the Propagator into Quantum Gates

Moving to Native Quantum Gates

We define the operators

$$XX(2t) = e^{-it\sigma_x^{(i)}\sigma_x^{(j)}}, \quad YY(2t) = e^{-it\sigma_y^{(i)}\sigma_y^{(j)}}, \quad ZZ(2t) = e^{-it\sigma_z^{(i)}\sigma_z^{(j)}} \quad (6)$$

Which allows us to write our trotterized propagator as

$$U(t) \approx \left((XX(2\delta)YY(2\delta)ZZ(2\delta))^{(1,2)} (XX(2\delta)YY(2\delta)ZZ(2\delta))^{(2,3)} \right)^n \quad (7)$$

Note: *again the splitting and ordering is not unique.*

One Way to Construct Circuit

Multiple ways of making the circuit has been tested during this work. The first one is the following (*example provided by challenge text*)

$$\begin{aligned}
 ZZ(2t) &= e^{-it\sigma_z^{(i)}\sigma_z^{(j)}} = C_X(\mathbb{1} \otimes R_Z(2t))C_X \\
 YY(2t) &= (R_x(\frac{\pi}{2}) \otimes R_x(\frac{\pi}{2}))ZZ(2t)(R_x(-\frac{\pi}{2}) \otimes R_x(-\frac{\pi}{2})) \\
 XX(2t) &= (R_y(\frac{\pi}{2}) \otimes R_y(\frac{\pi}{2}))ZZ(2t)(R_y(-\frac{\pi}{2}) \otimes R_y(-\frac{\pi}{2}))
 \end{aligned} \tag{8}$$

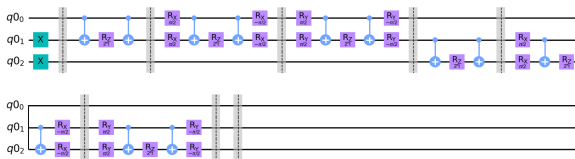


Figure: ZYXZYX-trotter step following from the gate compositions above. Repeating this circuit n times yields the approximate propagator in (7).

A Second Way of Splitting the Hamiltonian

Following is the second way I tried splitting the Hamiltonian in the trotterization.

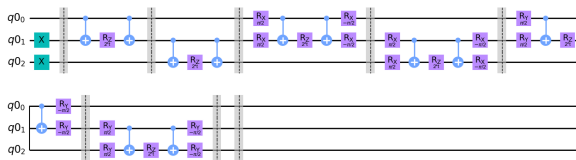


Figure: ZZYYXX-trotter step following from the non-uniqueness of the decomposition.

$$U(t) \approx \left(ZZ(2\delta)^{(1,2)} ZZ(2\delta)^{(2,3)} YY(2\delta)^{(1,2)} YY(2\delta)^{(2,3)} \right. \\ \left. XX(2\delta)^{(1,2)} XX(2\delta)^{(2,3)} \right)^n \quad (9)$$

Work Log from First Weeks of January 2022

- 1 Cleaned up the code from the IBM Jupyter notebook, organizing into functions and making the program flow more intuitive.
- 2 Implemented the trotter step functions for the $zyxzyx$ and $zzyyxx$ decompositions.
- 3 Added functionality to automate the process of running the numerical experiments.
- 4 Implemented functionality to compare the implemented circuits to the exact evolution using the L_2 operator norm. *This still needs polishing to work without hardcoding things.*
- 5 Reading papers on error correction and error mitigation. Zero noise protection using Mitiq was explored.
- 6 Started learning about symmetry protection (SP) and implemented $SU(2)$ symmetry protection.
- 7 Read paper [2] and started on implementing this way of trotterizing the Hamiltonian.

Preliminary Results

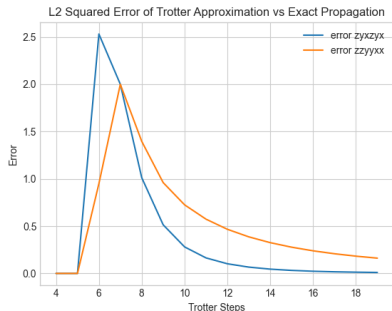


Figure: Error calculation comparison of two different trotter decompositions. Am working on adding others to this plot. Checked if this code is still working and it's definitely not..



Figure: First noisy simulation of zyxyzx and zzyyxx trotter decompositions. This is the first time I achieved the condition for entry into the competition of $> 30\%$ state tomography fidelity.

Symmetry Protection

Add more theory!

Should definitely add more technical explanation of the concept here!

The concept behind symmetry protection is to introduce unitaries that preserve the symmetry group of the Hamiltonian such that the rotations are able to cancel in the commutator in the trotter error. The homogenous Heisenberg model is $SU(2)$ -invariant, and we can use Hadamards, *in theory*, cancel the commutation error. The paper we have based our implementation on is [3]. Implementing the one mentioned using Hadamards we see a jump in fidelity for the zzyyxx-step of around 20%. We suspect the errors from non exact physical implementations of the Hadamard accumulates making more than four steps worse. There are definitely more avenues to explore with regard to symmetry protection (SP).

Symmetry Protection Result



Figure: Result of noisy simulation using the Hadamard sandwich symmetry protection

Work Log From Last Weeks of January and First Week of February 2022

- 1 Read paper on efficient implementation of arbitrary **U(4)** gates on quantum computer [2].
- 2 Read a couple of papers on quantum autoencoders and quantum machine learning: [4], [5], [6], [7], [8]
- 3 Achieved a fidelity on the noisy Jakarta backend of 80%, around a 30% improvement over previous best result!
- 4 Possible approach for Master: Quantum Denoising Autoencoder → Adapt Variational Fast Forwarding to fixed end time → Variational Denoising → Implement variable ansatz with cancellation rules → Test on quantum many-body Hamiltonian e.g. Jordan-Wigner / Bravyi-Kitaev transforms.
- 5 Starting to refactor my code to become more flexible by using a class based approach to avoid Spaghetti code.
- 6 Looking into Quantum Assisted Compilation. I should ask Morten to speak to Ryan LaRose at MSU as he is one of the authors.

Another Way of Constructing Trotter Circuits

Another way of constructing trotter circuits of the type we are considering is outlined in the 2004 paper by Farrokh Vatan and Colin Williams[2]. They provide *optimal* ways of constructing operators on the form

$$N(\alpha, \beta, \gamma) = e^{i(\alpha\sigma_x^{(i)}\sigma_x^{(j)} + \beta\sigma_y^{(i)}\sigma_y^{(j)} + \gamma\sigma_z^{(i)}\sigma_z^{(j)})} \quad (10)$$

In this work, *so far*, we have focused on their construction in . The essential result of that paper is “*Every two-qubit gate in $\mathbf{U}(4)$ can be realized, up to a global phase, by a circuit consisting of 15 elementary one-qubit gates and 3 CNOT gates.*”

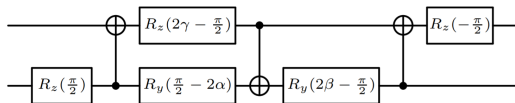


Figure: Figure (6) from [2]

Preliminary Results

Optimal Composition of $U(4)$ Operators.



Figure: Ran on noisy simulator with $SU(2)$ symmetry protection. Running again without gives the same results for $XX + YY + ZZ$ optimal construction step. Next step on this thread is to start doing cancellations. After talking with Ale, this might lead to an exact decomposition of just SWAPs.. If so I need to write up the mathematical argument.

Quantum Autoencoders

Summary of paper [5]

Paper List

Collecting QAE papers here: [5], [4], [6]

The main point of [5] is to demonstrate an application of quantum autoencoders (QAE) to denoise Greenberger-Horne-Zeilinger (GHZ) states. We start by specifying the quantum neuron like [7].

Let $\{|0\rangle, |1\rangle\}$ be our basis. Our training data set is $\{x_i, x_i\}_{i=1}^L$. The k -th layer, $k > 1$, of m neurons maps the state ρ_{k-1} of layer $k - 1$ onto it.

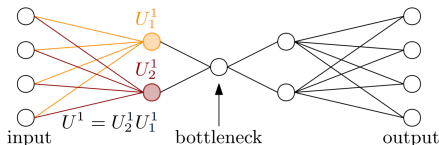


Figure: QAE Architecture from [5] Fig (1)

Quantum Autoencoders

Cost Function

$$\mathcal{N}^k(\rho_{k-1}) \equiv \text{Tr}\left(U(\rho_{k-1} \otimes (|0\rangle_{\text{out}} \langle 0|)^{\otimes m} U^\dagger)\right) \quad (11)$$

where the unitary $U = U_m \cdots U_1$ is subject to optimization. The full quantum channel of the QAE network with M layers is

$$\mathcal{N}(\rho^{\text{in}}) = \mathcal{N}^M\left(\cdots \mathcal{N}^2(\rho^{\text{in}} \cdots)\right) \quad (12)$$

The distance measure used is $1 - F$. For training data $\{\rho_i^{\text{in}}, |\psi_i^{\text{ref}}\rangle\}_{i=1}^L$ with pure desired outputs $F(\rho, |\psi\rangle) = \langle\psi|\rho|\psi\rangle$ which allows us to construct our cost function

$$C\left(\{\rho_i^{\text{in}}, |\psi_i^{\text{ref}}\rangle\}_{i=1}^L\right) = 1 - \bar{F}\left(\{\mathcal{N}(\rho_i^{\text{in}}), |\psi_i^{\text{ref}}\rangle\}_{i=1}^L\right) \quad (13)$$

where we have defined \bar{F} to be the average fidelity over all the samples L

$$\bar{F}\left(\{\rho_i, |\psi_i\rangle\}_{i=1}^L\right) = \frac{1}{L} \sum_{i=1}^L F(\rho_i, |\psi_i\rangle) \leq 1 \quad (14)$$

Quantum Autoencoders

Success Measure

In practice one has no access to the desired outputs of the neural network (NN) $\{|\psi_i^{\text{id}}\rangle\}_{i=1}^L$ the performance of autoencoders is best studied in a setting where the target states are known. We call the learning process successful if the mean validation function

$$F_{\text{val}}^{-}\left(\{\rho_i^{\text{in}}, |\psi_i^{\text{id}}\rangle\}_{i=1}^L\right) = \bar{F}\left(\{\mathcal{N}(\rho_i^{\text{in}}), |\psi_i^{\text{id}}\rangle\}_{i=1}^L\right) \quad (15)$$

is large. Particularly when compared to $\bar{F}(\{\rho_i^{\text{in}}, |\psi_i^{\text{id}}\rangle\}_{i=1}^L)$.

Important Points

In a densely connected QNN the gate number scales exponentially with the width. However moving to a sparse network, we can obtain a linear scaling with the circuit width. In addition, the number of gates scales only linearly with the depth of the network. Once the error probability passes beyond $p = 0.3$ then the approach seems to break down.

Variational Fast Forwarding

Summary of paper [8]

The main concept behind variational fast forwarding is to use a variational circuit to learn a diagonalization of a trotter step which is shorter in circuit depth than the trotter approximation. *“Given a Hamiltonian H on n -qubits evolved for a short time Δt with the simulation unitary $e^{-iH\Delta t}$, the goal is to find an approximation that allows the simulation at later times T to be fast-forwarded beyond the coherence time.”*

Algorithm Schematic

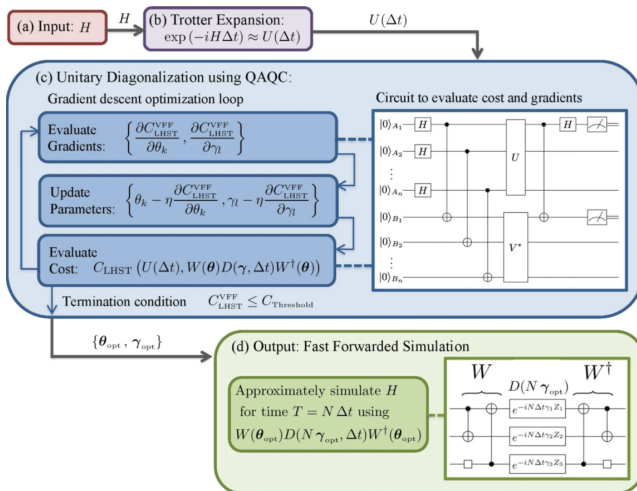


Figure: Image from [8]

Variational Fast Forwarding

Summary of paper [8]

The outline of the method is as follows:

- 1 Implement a unitary circuit $U(\Delta t)$ to approximate $e^{-iH\Delta t}$, the digital quantum simulation of a small timestep.
- 2 Compile $U(\Delta t)$ to a diagonal factorization $V(\theta, \gamma, \Delta t) = W(\theta)D(\gamma, \Delta t)W(\theta)^\dagger$ and use variational optimization to find optimal values for θ, γ . They use Local Hilbert-Schmidt Test (LHST)[9] to evaluate the cost function and use the Quantum-assisted Quantum Compiling (QAQC)[9] algorithm for the compilation of V .
- 3 Approximately fast-forward the simulation at large time $T = N\Delta t$ using the same circuit depth L : $e^{-iHT} \approx WD^N W^\dagger$

Quantum Assisted Quantum Compilation

Algorithm Overview

The goal of Quantum Assisted Quantum Compilation (QAQC) is to take a unitary U and return a gate sequence V , executable on a quantum computer, that has approximately the same action as U on any given input state. The success measure of the algorithm is the fidelity averaged over the Haar distribution

$$\bar{F}(U, V) = \int_{\psi} |\langle \psi(V) | \psi(U) \rangle|^2 d\psi \quad (16)$$

If $\bar{F} = 1$ we call V an *exact* compilation of U . If $\bar{F} \geq 1 - \epsilon$ we call V an *approximate* compilation of U .

Quantum Assisted Quantum Compilation

Consider an alphabet $\mathcal{A} = \{G_k(\alpha)\}$ of quantum gates. For a given quantum computer, the problem of compiling U to a gate sequence of length L is to determine

$$\begin{aligned} (\vec{\alpha}_{\text{opt.}}, \vec{k}_{\text{opt.}}) &= \underset{(\vec{\alpha}, \vec{k})}{\operatorname{argmin}} C(U, V_{\vec{k}}(\vec{\alpha})) \\ V_{\vec{k}}(\vec{\alpha}) &= G_{k_L}(\alpha_L) G_{k_{L-1}}(\alpha_{L-1}) \cdots G_{k_1}(\alpha_1) \end{aligned} \tag{17}$$

The optimization contains two parts





- 1 Continuous optimization of the gate parameters $\vec{\alpha}$ for a given gate structure.
- 2 Discrete optimization over the finite set of gate structures parametrized by \vec{k}

Quantum Assisted Quantum Compilation

Local Hilbert-Schmidt Test

The so called Local Hilbert-Schmidt Test [9, 10] is used by the authors of [8] in order to evaluate the gradient of the cost function.

References I

-  S. Lloyd, “Universal quantum simulators,” *Science*, vol. 273, no. 5278, pp. 1073–1078, 1996. [Online]. Available: <http://www.jstor.org/stable/2899535>
-  F. Vatan and C. Williams, “Optimal quantum circuits for general two-qubit gates,” *Physical Review A*, vol. 69, no. 3, Mar 2004. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevA.69.032315>
-  M. C. Tran, Y. Su, D. Carney, and J. M. Taylor, “Faster digital quantum simulation by symmetry protection,” *PRX Quantum*, vol. 2, no. 1, Feb 2021. [Online]. Available: <http://dx.doi.org/10.1103/PRXQuantum.2.010323>
-  V. Ngairangbam, M. Spannowsky, and M. Takeuchi, “Anomaly detection in high-energy physics using a quantum autoencoder,” 12 2021.

References II



D. Bondarenko and P. Feldmann, “Quantum autoencoders to denoise quantum data,” *Physical Review Letters*, vol. 124, no. 13, Mar 2020. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevLett.124.130502>



J. Romero, J. P. Olson, and A. Aspuru-Guzik, “Quantum autoencoders for efficient compression of quantum data,” *Quantum Science and Technology*, vol. 2, no. 4, p. 045001, Aug 2017. [Online]. Available: <http://dx.doi.org/10.1088/2058-9565/aa8072>



K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, “Training deep quantum neural networks,” *Nature Communications*, vol. 11, no. 1, Feb 2020. [Online]. Available: <http://dx.doi.org/10.1038/s41467-020-14454-2>

References III



C. Cîrstoiu, Z. Holmes, J. Iosue, L. Cincio, P. J. Coles, and A. Sornborger, "Variational fast forwarding for quantum simulation beyond the coherence time," *npj Quantum Information*, vol. 6, no. 1, Sep 2020. [Online]. Available: <http://dx.doi.org/10.1038/s41534-020-00302-0>







K. Sharma, S. Khatri, M. Cerezo, and P. J. Coles, "Noise resilience of variational quantum compiling," *New Journal of Physics*, vol. 22, no. 4, p. 043006, Apr 2020. [Online]. Available: <http://dx.doi.org/10.1088/1367-2630/ab784c>



S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles, "Quantum-assisted quantum compiling," *Quantum*, vol. 3, p. 140, May 2019. [Online]. Available: <https://doi.org/10.22331/q-2019-05-13-140>

References IV

-  M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
-  M. Schuld, *Supervised learning with quantum computers*. Springer, 2018.
-  A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, “Elementary gates for quantum computation,” *Phys. Rev. A*, vol. 52, pp. 3457–3467, Nov 1995. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.52.3457>
-  H. A. et al, *Qiskit: An Open-source Framework for Quantum Computing*, 2019.