Kagome Lattice – Open Science 2022

This is a solution for the quantum state preparation of the Kagome Lattice in a quantum computer for the open science challenge 2022. The solution is implemented as a python program (kagome_solution.py) which delegates its work to the Kagome lattice class (kagome_lattice.py). An auxiliary program (kagome_expected.py) is used to calculate the expected ground state of the lattice.

The goal of the program is to find the ground state of the lattice using the rules described in the open science challenge:

- It uses the VQE algorithm outlined in the Jupyter Notebook.
- It tries to find the ground state with a relative error less than 1%.
- It uses the quantum processor ibmq_guadalupe.

Implementation

The solution is implemented by feeding all the parameters of the VQE algorithm outlined in the Notebook as command line arguments. Thus, the arguments can be classified as follows.

$ python kagome_solution.py -h

Common Arguments

All arguments are optional.

Name	Description	Default Value
-h	Display a command line help.	None
-c CN, --cn CN	Connection String: Hub/Group/Project.	ibm-q-community/ibmquantumawards/open-science-22
-b,--run_backend	Quantum processor.	Ibmq_guadalupe
-t, --transpile_backend	Transpilation backend.	Ibmq_guadalupe
-q, --num_qubits	Number of qubits of the QPU.	16
-v,--verbosity	Verbosity level (1-5).	1

Ansatz Options

Control the behavior of the VQE ansatz. All arguments are optional.

Name	Description	Default Value
-a, --ansatz_type	Ansatz type: • ExcitationPreserving • EfficientSU2 • PauliTwoDesign • TwoLocal • RealAmplitudes	ExcitationPreserving

Optimizer Options

All arguments are optional.

Name	Description	Default Value
-o, --optimizer_type	Optimizer type: • SPSA • SLSQP • COBYLA • UMDA • GSLS • GradientDescent • L_BFGS_B • NELDER_MEAD • POWELL • NFT	SPSA
-i, --max_iter	Maximum number of iterations or function evals used by the optimizer.	100

Run time options

Miscellaneous run time options.

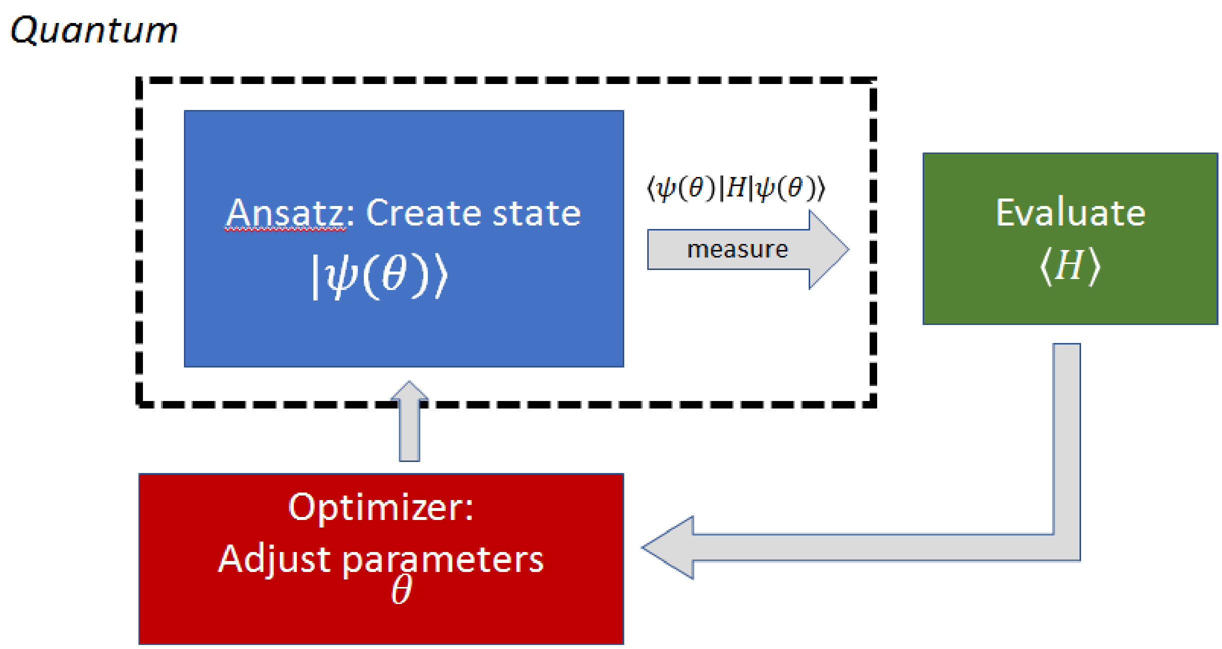
Name	Description	Default Value
-ol, --opt_level	Circuit optimization level (1-3)	1
-ui, --uniform_interaction	Heisenberg Model uniform interaction value.	Edge weight.
-up, --uniform_potential	Heisenberg Model uniform potential	0.0
-w, --weight	Lattice edge weight.	2.4
-s,--shots	Number of execution shots.	2048

Error Correction Options

Name	Description	Default Value
-r, --resilience_type	Resilience type (1-3): • T-Rex: 1 • ZNE: 2 • PEC: 3	2

How It Works

The solution leverages Qiskit's runtime VQE algorithm to find the ground state of the lattice (see figure 1).



Implementation wise, this is a python program that can be executed from the command line of your laptop. The program requires will execute the VQE algorithm in IBMQ Guadalupe using the following default parameters:

- Ansatz: The heuristic excitation-preserving wave function ansatz.
  - ExcitationPreserving (reps=1, entanglement='linear')
- Optimizer: Simultaneous Perturbation Stochastic Approximation (SPSA) optimizer.
  - SPSA (maxiter=100)
- Resilience/Error mitigation: Zero noise extrapolation - ZNE (1).
- Shots: 2048
- Edge weight: 1.0
- Heisenberg Model Uniform Interaction: 1.0
- Heisenberg Model Uniform potential: 0.0

These arguments can be changed at run time using command line arguments which are displayed by running:

$ python kagome_solution.py -h

```
usage: kagome_solution.py [-h] [-c CN] [-b RUNBACKEND] [-t TRANSPILE_BACKEND] [-q NUM_QUBITS]
                   [-s SHOTS] [-a
{ExcitationPreserving,EfficientSU2,PauliTwoDesign,TwoLocal,RealAmplitudes}]
                   [-o
{SPSA,SLSQP,COBYLA,UMDA,GSLS,GradientDescent,L_BFGS_B,NELDER_MEAD,POWELL,NFT}]
                   [-i MAX_ITER] [-ol {1,2,3}] [-ui UNIFORM_INTERACTION] [-up UNIFORM_POTENTIAL]
                   [-r {T-REx,ZNE,PEC}] [-w WEIGHT] [-v {1,2,3,4}]
```

optional arguments:

```
  -h, --help            show this help message and exit
  -c CN, --cn CN        Connection String: Hub/Group/Project (default: ibm-q-
community/ibmquantumawards/open-science-22)
  -b RUNBACKEND, --runbackend RUNBACKEND
                        Run backend
  -t TRANSPILE_BACKEND, --transpile_backend TRANSPILE_BACKEND
                        Transpile backend
  -q NUM_QUBITS, --num_qubits NUM_QUBITS
                        Run backend # of qubits
  -s SHOTS, --shots SHOTS
                        Shots
  -a {ExcitationPreserving,EfficientSU2,PauliTwoDesign,TwoLocal,RealAmplitudes}, --ansatz_type
{ExcitationPreserving,EfficientSU2,PauliTwoDe
sign,TwoLocal,RealAmplitudes}
                        Ansatz type
  -o {SPSA,SLSQP,COBYLA,UMDA,GSLS,GradientDescent,L_BFGS_B,NELDER_MEAD,POWELL,NFT}, --
optimizer_type {SPSA,SLSQP,COBYLA,UMDA,GSLS,GradientDe
scent,L_BFGS_B,NELDER_MEAD,POWELL,NFT}
                        Optimizer type
  -i MAX_ITER, --max_iter MAX_ITER
                        Maximum number of iterations
  -ol {1,2,3}, --opt_level {1,2,3}
                        Optimization level
  -ui UNIFORM_INTERACTION, --uniform_interaction UNIFORM_INTERACTION
                        HeisenbergModel uniform interaction
  -up UNIFORM_POTENTIAL, --uniform_potential UNIFORM_POTENTIAL
                        HeisenbergModel uniform potential
  -r {T-REx,ZNE,PEC}, --resilience_type {T-REx,ZNE,PEC}
                        Resilience type
  -w WEIGHT, --weight WEIGHT
                        Edge weight
  -v {1,2,3,4}, --verbosity {1,2,3,4}
                        Verbosity level
```

Scalability

The program is designed to run in any quantum processor with any number of qubits. For example to run in Geneva (27 qubits) with an NFT optimizer and uniform potential:

```
$ python3 kagome_solution.py -b ibm_geneva -t ibm_geneva -q 27 -a EfficientSU2 -w 1.0 -o NFT -up -1.0
```

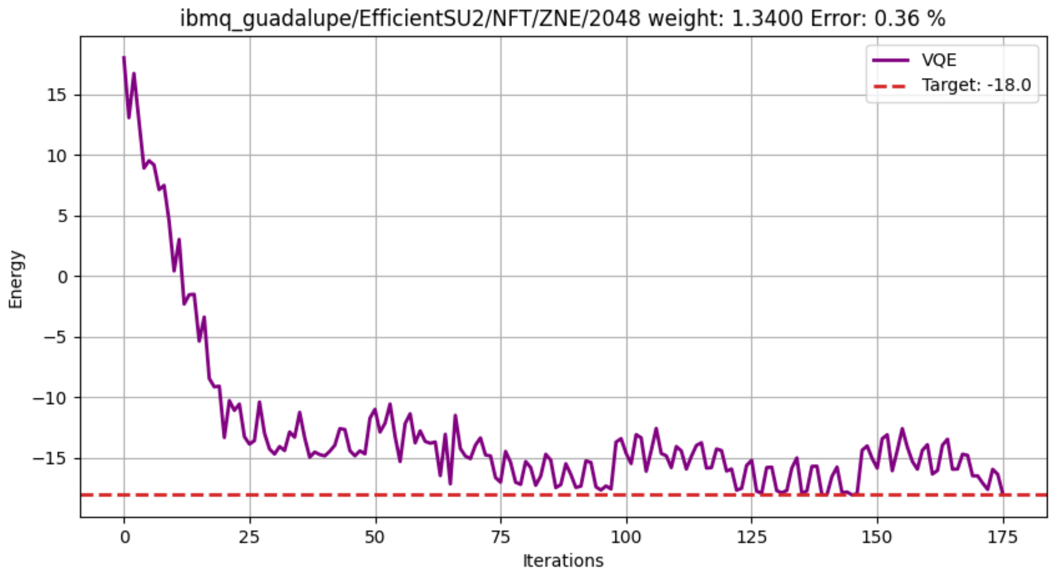
Results

Here is a list of multiple solutions obtained in ibmq_guadalupe:

Solution 1

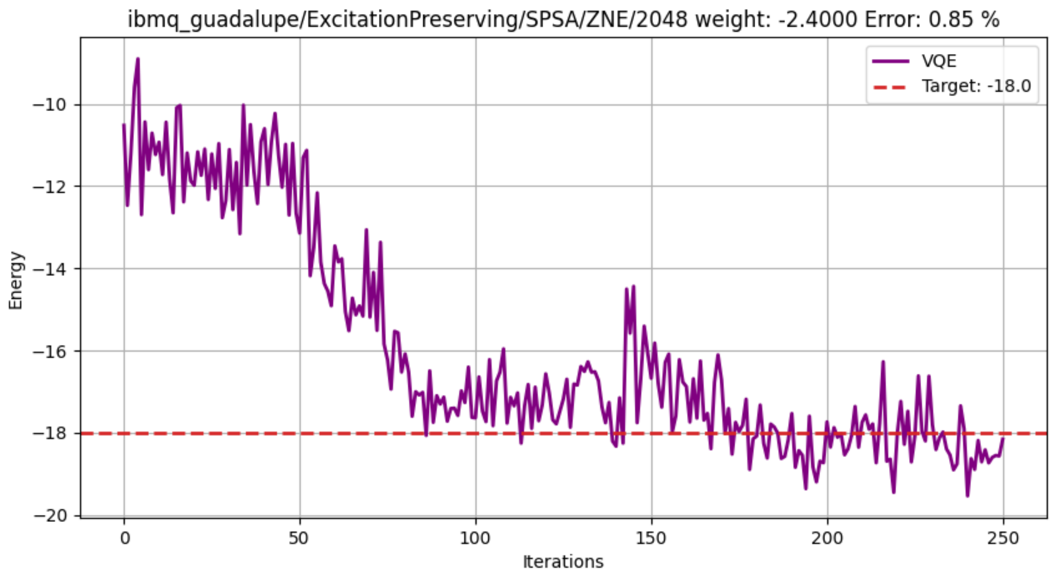
Backend: ibmq_guadalupe	Execution time (s): 19224.63

Ansatz: EfficientSU2 (reps=1, entanglement='reverse_linear') Optimizer : NFT(maxiter=175) Resilience : ZNE (2) Shots: 2048 Edge weight: 1.34	Expected ground state energy: -18.00000000 Computed ground state energy: -17.93602214 Result eigen value: -17.93602214 **Relative error: 0.35543258 %%**



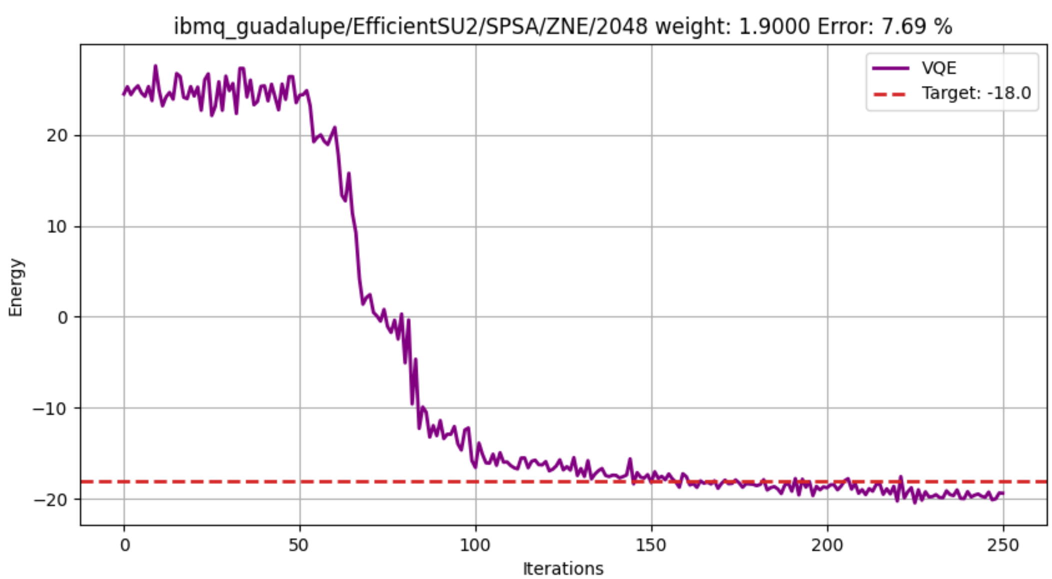
Solution 2

Backend: ibmq_guadalupe Ansatz: ExcitationPreserving(reps=1, entanglement='linear') Optimizer : SPSA(maxiter=100) Resilience : ZNE (2) Shots: 2048 Edge weight: -2.4000	Execution time (s): 113953.17 Expected ground state energy: -18.00000000 Computed ground state energy: -18.15273438 Result eigen value: -18.15273438 **Relative error: 0.848%**



Solution 3

Backend: ibmq_guadalupe Ansatz: EfficientSU2 (reps=1, entanglement='reverse_linear') Optimizer : SPSA(maxiter=100) Resilience : ZNE (2) Shots: 2048 Edge weight: 1.9000	Execution time (s): 102658.02 Expected ground state energy: -18.00000000 Computed ground state energy: -19.38377279 Result eigen value: -19.38377279 **Relative error: 7.687%**



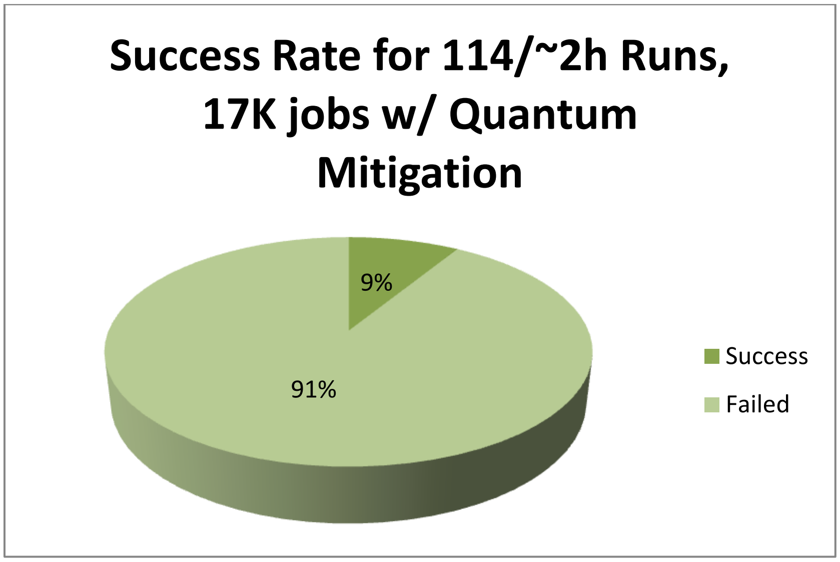
Final Results and Metrics

The following section details the error mitigation techniques applied both at the quantum and classical levels.

Quantum Error Mitigation

- Ansatz: ZNE (Zero Noise Extrapolation)
- Circuit optimization level = 1

Here is a chart of the success rate with Quantum mitigation only. It is pretty awful, runs failed 91% of the time. **Note: that success means the error threshold falls below 1%.**

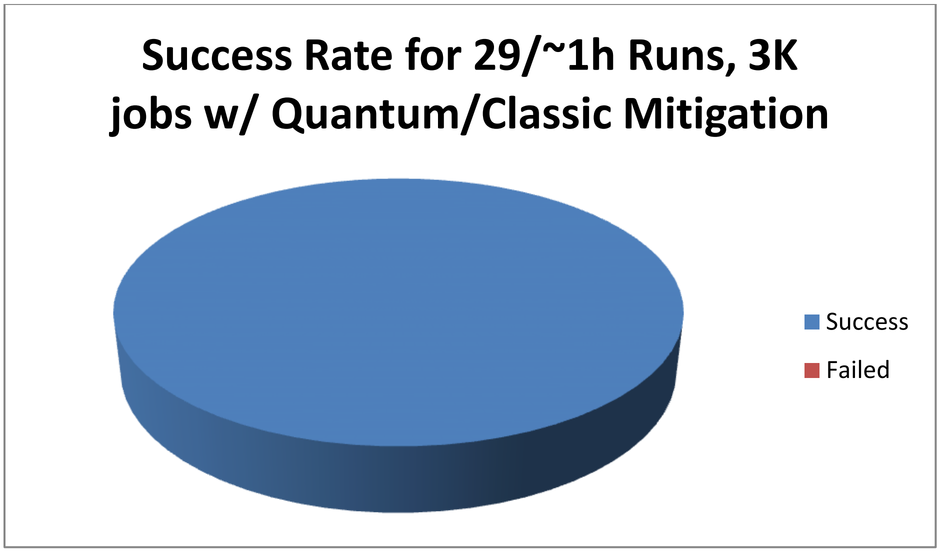


Classical Error Mitigation Techniques

The following classic mitigation techniques were applied after those awful error rates.

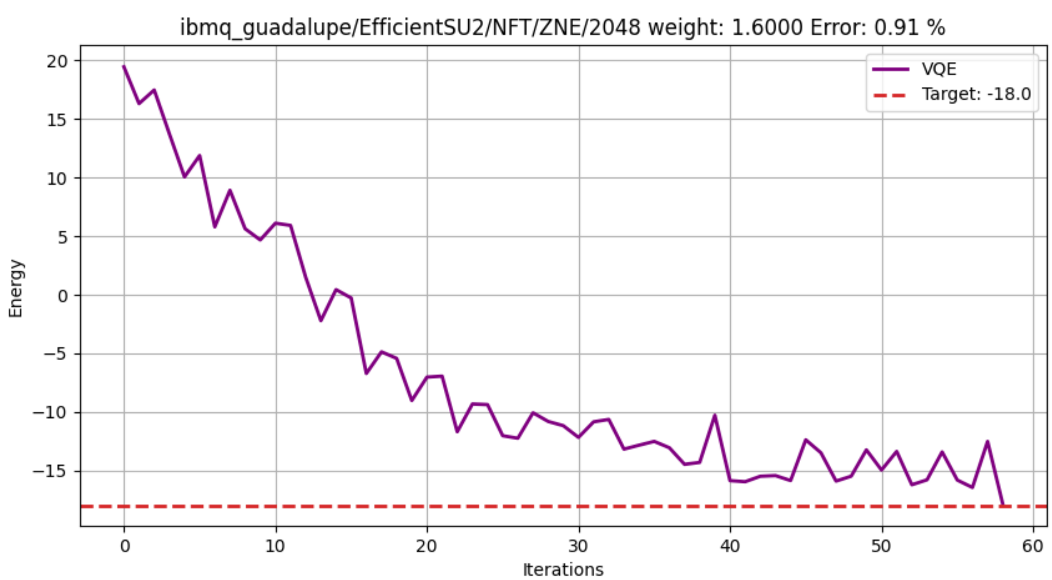
1. If at some stage of the optimization cycle the point falls below the error threshold (default 1%), the process is aborted and the optimization returns the collected data.
2. If a point falls below the target ground state by some delta = ||x| - |target||, then the uniform interaction (UI) is decreased on the fly (a new Hamiltonian is constructed with this UI and the process continues). This has the effect of moving the curve upwards towards the ground state.
3. If the optimization cycle completes and the final point is above the target ground state by delta. The VQE optimization recurses with a new Hamiltonian with an increased uniform interaction. This is to drive the curve downwards towards the target. As a failsafe to avoid infinite recursions, the maximum number of recursive calls allowed is 5. **Note that rule 1) acts as an exit condition for rules 2, 3** *(i.e. the optimization may abort in the middle of a recursive call if the error threshold falls below 1%)***.**

This changes things dramatically, with a final success rate of 100%. Not one experiment has failed!

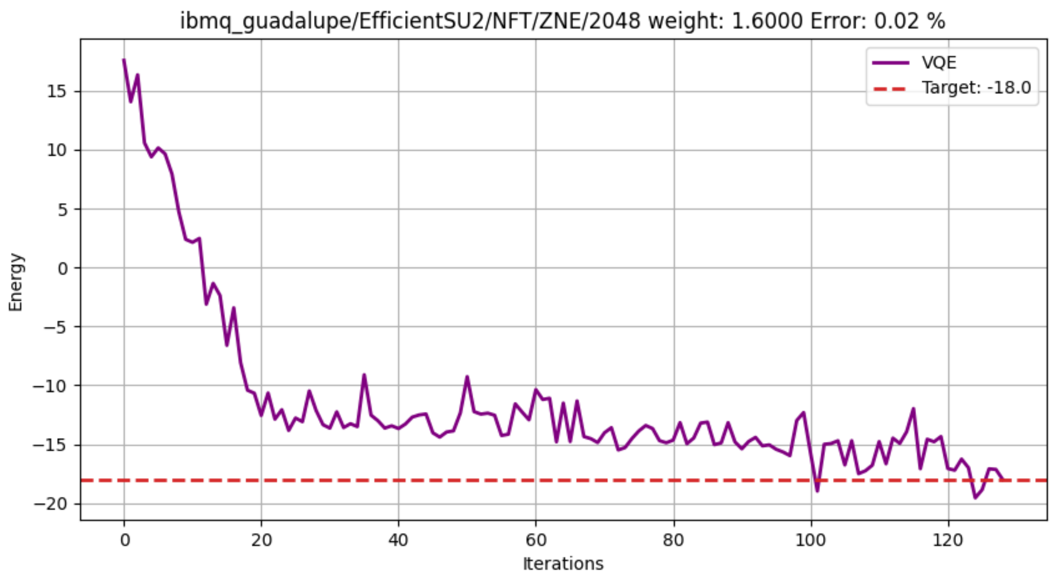


Total Runs	Success	Failed	Avg run time (h)	Maxiter	Total Jobs
114	10	104	2	150	17100
29	29	0	1	100	2900

Here we can see rule 1 in action: at iteraction ~68 the x point happened to fall below 1% aborting the optimization process (which defaults at 100 iteractions).



Here we can see rule 3 in action. The optimizer reached the end at 100 cycles below target, a recursive call kicks in with a lower uniform interaction, *however at loop ~130 rule 1 kicks in and the process completes.*



References

IBM Quantum Awards: Open Science Prize 2022 https://github.com/qiskit-community/open-science-prize-2022/blob/main/kagome-vqe.ipynb

IBM Quantum Awards Event site: https://ibmquantumawards.bemyapp.com/#/event

Qiskit Error Mitigation: https://qiskit.org/documentation/partners/qiskit_ibm_runtime/tutorials/Error-Suppression-and-Error-Mitigation.html