Lab 3: The Power of 'good sampling' for simulating a chemistry Hamiltonian with SQD

Welcome to the third coding challenge of the Qiskit Global Summer School. In this lab, we explore one of the most promising applications of quantum computing, which is quantum chemistry. We present you with a real-life example of simulating a molecule using a quantum computer based on a workflow that quantum chemists may usually follow. We will explain what each step takes to achieve this and walk you through the entire workflow.

Recommended study
Prior to going through this challenge, we recommend that you take a moment and check out the Variational Algorithm Design course and one of our latest courses on IBM Quantum Learning, Quantum Diagonalization Algorithms.

Table of Contents

Requirements

Before starting this tutorial, please make sure that you have the following installed:

- Qiskit SDK 2.0 or later with visualization support (`pip install 'qiskit[visualization]'`)
- Qiskit Runtime 0.40 or later (`pip install qiskit-ibm-runtime`)
- Qiskit Addons SQD 0.11.0 or later (`pip install qiskit-addon-sqd`)
- ffsim (`pip install ffsim`)

⚠️ **Note:** You need to have the below packages installed in order to run this lab, of which some are not available on Windows. If you are using Windows we recommend you to use an online lab environment.

```
# %pip install "qc-grader[qiskit,jupyter] @ git+https://github.com/qiskit-community/Quant
# %pip install pyscf
# %pip install ffsim
# %pip install qiskit_addon_sqd
```

```
Collecting qc-grader@ git+https://github.com/qiskit-community/Quantum-Challenge-Grade
  Cloning https://github.com/qiskit-community/Quantum-Challenge-Grader.git to /tmp/pi
  Running command git clone --filter=blob:none --quiet https://github.com/qiskit-comm
  Resolved https://github.com/qiskit-community/Quantum-Challenge-Grader.git to commit
  Preparing metadata (setup.py) ... done
Requirement already satisfied: typeguard in /usr/local/lib/python3.11/dist-packages (
Collecting jsonpickle==3.0.3 (from qc-grader@ git+https://github.com/qiskit-community
  Downloading jsonpickle-3.0.3-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: requests==2.32.3 in /usr/local/lib/python3.11/dist-pac
Collecting ipycytoscape (from qc-grader@ git+https://github.com/qiskit-community/Quar
  Downloading ipycytoscape-1.3.3-py2.py3-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: plotly in /usr/local/lib/python3.11/dist-packages (fro
Collecting networkx==3.2.1 (from qc-grader@ git+https://github.com/qiskit-community/Q
  Downloading networkx-3.2.1-py3-none-any.whl.metadata (5.2 kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.11/dist-packages (f
Collecting ibm-platform-services==0.66.1 (from qc-grader@ git+https://github.com/qisk
  Downloading ibm_platform_services-0.66.1-py3-none-any.whl.metadata (9.0 kB)
Collecting ibm_cloud_sdk_core<4.0.0,>=3.24.1 (from ibm-platform-services==0.66.1->qc-
  Downloading ibm_cloud_sdk_core-3.24.2-py3-none-any.whl.metadata (8.7 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
Collecting qiskit~=2.1.0 (from qiskit[visualization]~=2.1.0; extra == "qiskit"->qc-gr
  Downloading qiskit-2.1.1-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.m
Collecting qiskit-ibm-runtime (from qc-grader@ git+https://github.com/qiskit-communit
  Downloading qiskit_ibm_runtime-0.40.1-py3-none-any.whl.metadata (21 kB)
Collecting qiskit-aer (from qc-grader@ git+https://github.com/qiskit-community/Quantu
  Downloading qiskit_aer-0.17.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
Collecting qiskit_serverless (from qc-grader@ git+https://github.com/qiskit-community
  Downloading qiskit_serverless-0.25.2-py3-none-any.whl.metadata (5.4 kB)
Collecting jupyterlab (from qc-grader@ git+https://github.com/qiskit-community/Quantu
  Downloading jupyterlab-4.4.5-py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.11/dist-packages (
Collecting rustworkx>=0.15.0 (from qiskit~=2.1.0->qiskit[visualization]~=2.1.0; extra
  Downloading rustworkx-0.16.0-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.w
Requirement already satisfied: numpy<3,>=1.17 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: scipy>=1.5 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.11/dist-packages (
Collecting stevedore>=3.0.0 (from qiskit~=2.1.0->qiskit[visualization]~=2.1.0; extra
  Downloading stevedore-5.4.1-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: matplotlib>=3.3 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pydot in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: Pillow>=4.2.1 in /usr/local/lib/python3.11/dist-packag
Collecting pylatexenc>=1.4 (from qiskit[visualization]~=2.1.0; extra == "qiskit"->qc-
  Downloading pylatexenc-2.10.tar.gz (162 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 162.6/162.6 kB 5.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: seaborn>=0.9.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: ipywidgets>=7.6.0 in /usr/local/lib/python3.11/dist-pa
Collecting spectate>=1.0.0 (from ipycytoscape->qc-grader@ git+https://github.com/qisk
  Downloading spectate-1.0.1-py2.py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: debugpy>=1.0 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: ipython>=7.23.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.11/di
Requirement already satisfied: matplotlib-inline>=0.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (
```

```
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: tornado>=6.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: traitlets>=5.1.0 in /usr/local/lib/python3.11/dist-pac
Collecting async-lru>=1.0.0 (from jupyterlab->qc-grader@ git+https://github.com/qiski
  Downloading async_lru-2.0.5-py3-none-any.whl.metadata (4.5 kB)
Requirement already satisfied: httpx>=0.25.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: jinja2>=3.0.3 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.11/dist-package
Collecting jupyter-lsp>=2.0.0 (from jupyterlab->qc-grader@ git+https://github.com/qis
  Downloading jupyter_lsp-2.2.6-py3-none-any.whl.metadata (1.8 kB)
Collecting jupyter-server<3,>=2.4.0 (from jupyterlab->qc-grader@ git+https://github.c
  Downloading jupyter_server-2.16.0-py3-none-any.whl.metadata (8.5 kB)
Collecting jupyterlab-server<3,>=2.27.1 (from jupyterlab->qc-grader@ git+https://gith
  Downloading jupyterlab_server-2.27.3-py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: notebook-shim>=0.2 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: setuptools>=41.1.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: python-dateutil>=2.8.0 in /usr/local/lib/python3.11/di
Collecting requests-ntlm>=1.1.0 (from qiskit-ibm-runtime->qc-grader@ git+https://gith
  Downloading requests_ntlm-1.3.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: pydantic>=2.5.0 in /usr/local/lib/python3.11/dist-pack
Collecting ray<3,>=2.30 (from ray[default]<3,>=2.30->qiskit_serverless->qc-grader@ gi
  Downloading ray-2.48.0-cp311-cp311-manylinux2014_x86_64.whl.metadata (19 kB)
Requirement already satisfied: importlib-metadata<9,>=5.2.0 in /usr/local/lib/python3
Collecting cloudpickle==2.2.1 (from qiskit_serverless->qc-grader@ git+https://github.
  Downloading cloudpickle-2.2.1-py3-none-any.whl.metadata (6.9 kB)
Requirement already satisfied: tqdm<5,>=4.66.3 in /usr/local/lib/python3.11/dist-pack
Collecting opentelemetry-api<1.33.1,>=1.18.0 (from qiskit_serverless->qc-grader@ git+
  Downloading opentelemetry_api-1.33.0-py3-none-any.whl.metadata (1.6 kB)
Collecting opentelemetry-sdk<1.33.1,>=1.18.0 (from qiskit_serverless->qc-grader@ git+
  Downloading opentelemetry_sdk-1.33.0-py3-none-any.whl.metadata (1.6 kB)
Collecting opentelemetry-exporter-otlp-proto-grpc<1.33.1,>=1.18.0 (from qiskit_server
  Downloading opentelemetry_exporter_otlp_proto_grpc-1.33.0-py3-none-any.whl.metadata
Collecting s3fs>=2023.6.0 (from qiskit_serverless->qc-grader@ git+https://github.com/
  Downloading s3fs-2025.7.0-py3-none-any.whl.metadata (1.4 kB)
Collecting opentelemetry-instrumentation-requests>=0.40b0 (from qiskit_serverless->qc
  Downloading opentelemetry_instrumentation_requests-0.56b0-py3-none-any.whl.metadata
Collecting ipywidgets>=7.6.0 (from ipycytoscape->qc-grader@ git+https://github.com/qi
  Downloading ipywidgets-8.1.7-py3-none-any.whl.metadata (2.4 kB)
Collecting ipython>=7.23.1 (from ipykernel->qc-grader@ git+https://github.com/qiskit-
  Downloading ipython-8.37.0-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: pyarrow<19,>=16.0.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: aiohttp<4,>=3.10.0 in /usr/local/lib/python3.11/dist-p
Collecting zipp==3.19.1 (from qiskit_serverless->qc-grader@ git+https://github.com/qi
  Downloading zipp-3.19.1-py3-none-any.whl.metadata (3.5 kB)
Collecting certifi>=2017.4.17 (from requests==2.32.3->qc-grader@ git+https://github.c
  Downloading certifi-2024.7.4-py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (
INFO: pip is looking at multiple versions of ibm-cloud-sdk-core to determine which ve
Collecting ibm_cloud_sdk_core<4.0.0,>=3.24.1 (from ibm-platform-services==0.66.1->qc-
```

```
  Downloading ibm_cloud_sdk_core-3.24.1-py3-none-any.whl.metadata (8.7 kB)
Requirement already satisfied: PyJWT<3.0.0,>=2.8.0 in /usr/local/lib/python3.11/dist-
INFO: pip is looking at multiple versions of importlib-metadata to determine which ve
Collecting importlib-metadata<9,>=5.2.0 (from qiskit_serverless->qc-grader@ git+https
  Downloading importlib_metadata-8.6.1-py3-none-any.whl.metadata (4.7 kB)
  Downloading importlib_metadata-8.6.0-py3-none-any.whl.metadata (4.7 kB)
  Downloading importlib_metadata-8.5.0-py3-none-any.whl.metadata (4.8 kB)
  Downloading importlib_metadata-8.4.0-py3-none-any.whl.metadata (4.7 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (
Collecting jedi>=0.16 (from ipython>=7.23.1->ipykernel->qc-grader@ git+https://github
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in /usr/local/lib/python
Requirement already satisfied: pygments>=2.4.0 in /usr/local/lib/python3.11/dist-pack
Collecting stack_data (from ipython>=7.23.1->ipykernel->qc-grader@ git+https://github
  Downloading stack_data-0.6.3-py3-none-any.whl.metadata (18 kB)
Collecting traitlets>=5.1.0 (from ipykernel->qc-grader@ git+https://github.com/qiskit
  Downloading traitlets-5.14.3-py3-none-any.whl.metadata (10 kB)
Collecting comm>=0.1.3 (from ipywidgets>=7.6.0->ipycytoscape->qc-grader@ git+https://
  Downloading comm-0.2.2-py3-none-any.whl.metadata (3.7 kB)
Collecting widgetsnbextension~=4.0.14 (from ipywidgets>=7.6.0->ipycytoscape->qc-grade
  Downloading widgetsnbextension-4.0.14-py3-none-any.whl.metadata (1.6 kB)
Requirement already satisfied: jupyterlab_widgets~=3.0.15 in /usr/local/lib/python3.1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: argon2-cffi>=21.1 in /usr/local/lib/python3.11/dist-pa
Collecting jupyter-client>=6.1.12 (from ipykernel->qc-grader@ git+https://github.com/
  Downloading jupyter_client-8.6.3-py3-none-any.whl.metadata (8.3 kB)
Collecting jupyter-events>=0.11.0 (from jupyter-server<3,>=2.4.0->jupyterlab->qc-grad
  Downloading jupyter_events-0.12.0-py3-none-any.whl.metadata (5.8 kB)
Collecting jupyter-server-terminals>=0.4.4 (from jupyter-server<3,>=2.4.0->jupyterlab
  Downloading jupyter_server_terminals-0.5.3-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: nbconvert>=6.4.4 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: nbformat>=5.3.0 in /usr/local/lib/python3.11/dist-pack
Collecting overrides>=5.0 (from jupyter-server<3,>=2.4.0->jupyterlab->qc-grader@ git+
  Downloading overrides-7.7.0-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: prometheus-client>=0.9 in /usr/local/lib/python3.11/di
Requirement already satisfied: send2trash>=1.8.2 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: websocket-client>=1.7 in /usr/local/lib/python3.11/dis
Requirement already satisfied: babel>=2.10 in /usr/local/lib/python3.11/dist-packages
Collecting json5>=0.9.0 (from jupyterlab-server<3,>=2.27.1->jupyterlab->qc-grader@ gi
  Downloading json5-0.12.0-py3-none-any.whl.metadata (36 kB)
Requirement already satisfied: jsonschema>=4.18.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-pac
Collecting deprecated>=1.2.6 (from opentelemetry-api<1.33.1,>=1.18.0->qiskit_serverle
  Downloading Deprecated-1.2.18-py2.py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: googleapis-common-protos~=1.52 in /usr/local/lib/pytho
Requirement already satisfied: grpcio<2.0.0,>=1.63.2 in /usr/local/lib/python3.11/dis
Collecting opentelemetry-exporter-otlp-proto-common==1.33.0 (from opentelemetry-expor
  Downloading opentelemetry_exporter_otlp_proto_common-1.33.0-py3-none-any.whl.metada
Collecting opentelemetry-proto==1.33.0 (from opentelemetry-exporter-otlp-proto-grpc<1
  Downloading opentelemetry_proto-1.33.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: protobuf<6.0,>=5.0 in /usr/local/lib/python3.11/dist-p
Collecting opentelemetry-instrumentation==0.56b0 (from opentelemetry-instrumentation-
  Downloading opentelemetry_instrumentation-0.56b0-py3-none-any.whl.metadata (6.7 kB)
Collecting opentelemetry-semantic-conventions==0.56b0 (from opentelemetry-instrumenta
```

```
  Downloading opentelemetry_semantic_conventions-0.56b0-py3-none-any.whl.metadata (2.
Collecting opentelemetry-util-http==0.56b0 (from opentelemetry-instrumentation-reques
  Downloading opentelemetry_util_http-0.56b0-py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: wrapt<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-
INFO: pip is looking at multiple versions of opentelemetry-semantic-conventions to de
Collecting opentelemetry-instrumentation-requests>=0.40b0 (from qiskit_serverless->qc
  Downloading opentelemetry_instrumentation_requests-0.55b1-py3-none-any.whl.metadata
Collecting opentelemetry-instrumentation==0.55b1 (from opentelemetry-instrumentation-
  Downloading opentelemetry_instrumentation-0.55b1-py3-none-any.whl.metadata (6.7 kB)
Collecting opentelemetry-semantic-conventions==0.55b1 (from opentelemetry-instrumenta
  Downloading opentelemetry_semantic_conventions-0.55b1-py3-none-any.whl.metadata (2.
Collecting opentelemetry-util-http==0.55b1 (from opentelemetry-instrumentation-reques
  Downloading opentelemetry_util_http-0.55b1-py3-none-any.whl.metadata (2.6 kB)
Collecting opentelemetry-instrumentation-requests>=0.40b0 (from qiskit_serverless->qc
  Downloading opentelemetry_instrumentation_requests-0.55b0-py3-none-any.whl.metadata
Collecting opentelemetry-instrumentation==0.55b0 (from opentelemetry-instrumentation-
  Downloading opentelemetry_instrumentation-0.55b0-py3-none-any.whl.metadata (6.7 kB)
Collecting opentelemetry-semantic-conventions==0.55b0 (from opentelemetry-instrumenta
  Downloading opentelemetry_semantic_conventions-0.55b0-py3-none-any.whl.metadata (2.
Collecting opentelemetry-util-http==0.55b0 (from opentelemetry-instrumentation-reques
  Downloading opentelemetry_util_http-0.55b0-py3-none-any.whl.metadata (2.6 kB)
Collecting opentelemetry-instrumentation-requests>=0.40b0 (from qiskit_serverless->qc
  Downloading opentelemetry_instrumentation_requests-0.54b1-py3-none-any.whl.metadata
Collecting opentelemetry-instrumentation==0.54b1 (from opentelemetry-instrumentation-
  Downloading opentelemetry_instrumentation-0.54b1-py3-none-any.whl.metadata (6.8 kB)
Collecting opentelemetry-semantic-conventions==0.54b1 (from opentelemetry-instrumenta
  Downloading opentelemetry_semantic_conventions-0.54b1-py3-none-any.whl.metadata (2.
Collecting opentelemetry-util-http==0.54b1 (from opentelemetry-instrumentation-reques
  Downloading opentelemetry_util_http-0.54b1-py3-none-any.whl.metadata (2.6 kB)
Collecting opentelemetry-instrumentation-requests>=0.40b0 (from qiskit_serverless->qc
  Downloading opentelemetry_instrumentation_requests-0.54b0-py3-none-any.whl.metadata
Collecting opentelemetry-instrumentation==0.54b0 (from opentelemetry-instrumentation-
  Downloading opentelemetry_instrumentation-0.54b0-py3-none-any.whl.metadata (6.8 kB)
Collecting opentelemetry-semantic-conventions==0.54b0 (from opentelemetry-instrumenta
  Downloading opentelemetry_semantic_conventions-0.54b0-py3-none-any.whl.metadata (2.
Collecting opentelemetry-util-http==0.54b0 (from opentelemetry-instrumentation-reques
  Downloading opentelemetry_util_http-0.54b0-py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dis
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
Collecting symengine<0.14,>=0.11 (from qiskit~=2.1.0->qiskit[visualization]~=2.1.0; e
  Downloading symengine-0.13.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: msgpack<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages (fro
Collecting aiohttp_cors (from ray[default]<3,>=2.30->qiskit_serverless->qc-grader@ gi
  Downloading aiohttp_cors-0.8.1-py3-none-any.whl.metadata (20 kB)
Collecting colorful (from ray[default]<3,>=2.30->qiskit_serverless->qc-grader@ git+ht
  Downloading colorful-0.5.7-py2.py3-none-any.whl.metadata (16 kB)
Collecting py-spy>=0.2.0 (from ray[default]<3,>=2.30->qiskit_serverless->qc-grader@ g
  Downloading py_spy-0.4.0-py2.py3-none-manylinux_2_5_x86_64.manylinux1_x86_64.whl.me
Collecting opencensus (from ray[default]<3,>=2.30->qiskit_serverless->qc-grader@ git+
  Downloading opencensus-0.11.4-py2.py3-none-any.whl.metadata (12 kB)
Collecting opentelemetry-exporter-prometheus (from ray[default]<3,>=2.30->qiskit_serv
  Downloading opentelemetry_exporter_prometheus-0.56b0-py3-none-any.whl.metadata (1.8
Requirement already satisfied: smart_open in /usr/local/lib/python3.11/dist-packages
Collecting virtualenv!=20.21.1,>=20.0.24 (from ray[default]<3,>=2.30->qiskit_serverle
  Downloading virtualenv-20.32.0-py3-none-any.whl.metadata (4.5 kB)
```

```
Requirement already satisfied: cryptography>=1.3 in /usr/local/lib/python3.11/dist-pa
Collecting pyspnego>=0.4.0 (from requests-ntlm>=1.1.0->qiskit-ibm-runtime->qc-grader@
  Downloading pyspnego-0.11.2-py3-none-any.whl.metadata (5.4 kB)
Collecting aiobotocore<3.0.0,>=2.5.4 (from s3fs>=2023.6.0->qiskit_serverless->qc-grad
  Downloading aiobotocore-2.23.1-py3-none-any.whl.metadata (25 kB)
Requirement already satisfied: fsspec==2025.7.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages
Collecting pbr>=2.0.0 (from stevedore>=3.0.0->qiskit~=2.1.0->qiskit[visualization]~=2
  Downloading pbr-6.1.1-py2.py3-none-any.whl.metadata (3.4 kB)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-p
Collecting aioitertools<1.0.0,>=0.5.1 (from aiobotocore<3.0.0,>=2.5.4->s3fs>=2023.6.0
  Downloading aioitertools-0.12.0-py3-none-any.whl.metadata (3.8 kB)
Collecting botocore<1.38.47,>=1.38.40 (from aiobotocore<3.0.0,>=2.5.4->s3fs>=2023.6.0
  Downloading botocore-1.38.46-py3-none-any.whl.metadata (5.7 kB)
Collecting jmespath<2.0.0,>=0.7.1 (from aiobotocore<3.0.0,>=2.5.4->s3fs>=2023.6.0->qi
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.11/dist
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packa
Collecting python-json-logger>=2.0.4 (from jupyter-events>=0.11.0->jupyter-server<3,>
  Downloading python_json_logger-3.3.0-py3-none-any.whl.metadata (4.0 kB)
Collecting rfc3339-validator (from jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->
  Downloading rfc3339_validator-0.1.4-py2.py3-none-any.whl.metadata (1.5 kB)
Collecting rfc3986-validator>=0.1.1 (from jupyter-events>=0.11.0->jupyter-server<3,>=
  Downloading rfc3986_validator-0.1.1-py2.py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: defusedxml in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.11/dist-
Requirement already satisfied: mistune<4,>=2.0.3 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.11/dist
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (fr
Collecting distlib<1,>=0.3.7 (from virtualenv!=20.21.1,>=20.0.24->ray[default]<3,>=2.
  Downloading distlib-0.4.0-py2.py3-none-any.whl.metadata (5.2 kB)
Collecting opencensus-context>=0.1.3 (from opencensus->ray[default]<3,>=2.30->qiskit_
  Downloading opencensus_context-0.1.3-py2.py3-none-any.whl.metadata (3.3 kB)
Requirement already satisfied: google-api-core<3.0.0,>=1.0.0 in /usr/local/lib/python
INFO: pip is looking at multiple versions of opentelemetry-exporter-prometheus to det
Collecting opentelemetry-exporter-prometheus (from ray[default]<3,>=2.30->qiskit_serv
  Downloading opentelemetry_exporter_prometheus-0.55b1-py3-none-any.whl.metadata (1.9
  Downloading opentelemetry_exporter_prometheus-0.55b0-py3-none-any.whl.metadata (1.9
  Downloading opentelemetry_exporter_prometheus-0.54b1-py3-none-any.whl.metadata (1.9
  Downloading opentelemetry_exporter_prometheus-0.54b0-py3-none-any.whl.metadata (1.9
Collecting executing>=1.2.0 (from stack_data->ipython>=7.23.1->ipykernel->qc-grader@
  Downloading executing-2.2.0-py2.py3-none-any.whl.metadata (8.9 kB)
Collecting asttokens>=2.1.0 (from stack_data->ipython>=7.23.1->ipykernel->qc-grader@
  Downloading asttokens-3.0.0-py3-none-any.whl.metadata (4.7 kB)
Collecting pure-eval (from stack_data->ipython>=7.23.1->ipykernel->qc-grader@ git+htt
  Downloading pure_eval-0.2.3-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (
```

```
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: proto-plus<2.0.0,>=1.22.3 in /usr/local/lib/python3.11
Requirement already satisfied: google-auth<3.0.0,>=2.14.1 in /usr/local/lib/python3.1
Collecting fqdn (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupy
  Downloading fqdn-1.5.1-py3-none-any.whl.metadata (1.4 kB)
Collecting isoduration (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.
  Downloading isoduration-20.11.0-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: jsonpointer>1.13 in /usr/local/lib/python3.11/dist-pac
Collecting rfc3987-syntax>=1.1.0 (from jsonschema[format-nongpl]>=4.18.0->jupyter-eve
  Downloading rfc3987_syntax-1.1.0-py3-none-any.whl.metadata (7.7 kB)
Collecting uri-template (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11
  Downloading uri_template-1.3.0-py3-none-any.whl.metadata (8.8 kB)
Requirement already satisfied: webcolors>=24.6.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/dist-packag
Collecting lark>=1.2.2 (from rfc3987-syntax>=1.1.0->jsonschema[format-nongpl]>=4.18.0
  Downloading lark-1.2.2-py3-none-any.whl.metadata (1.8 kB)
Collecting arrow>=0.15.0 (from isoduration->jsonschema[format-nongpl]>=4.18.0->jupyte
  Downloading arrow-1.3.0-py3-none-any.whl.metadata (7.5 kB)
Collecting types-python-dateutil>=2.8.10 (from arrow>=0.15.0->isoduration->jsonschema
  Downloading types_python_dateutil-2.9.0.20250708-py3-none-any.whl.metadata (1.9 kB)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.11/dist
Downloading ibm_platform_services-0.66.1-py3-none-any.whl (363 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.9/363.9 kB 16.2 MB/s eta 0:00:00
Downloading jsonpickle-3.0.3-py3-none-any.whl (40 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.8/40.8 kB 2.4 MB/s eta 0:00:00
Downloading networkx-3.2.1-py3-none-any.whl (1.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 42.9 MB/s eta 0:00:00
Downloading qiskit-2.1.1-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.5/7.5 MB 61.0 MB/s eta 0:00:00
Downloading ipycytoscape-1.3.3-py2.py3-none-any.whl (3.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.6/3.6 MB 49.0 MB/s eta 0:00:00
Downloading jupyterlab-4.4.5-py3-none-any.whl (12.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.3/12.3 MB 48.7 MB/s eta 0:00:00
Downloading qiskit_aer-0.17.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.4/12.4 MB 59.2 MB/s eta 0:00:00
Downloading qiskit_ibm_runtime-0.40.1-py3-none-any.whl (3.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 54.6 MB/s eta 0:00:00
Downloading qiskit_serverless-0.25.2-py3-none-any.whl (56 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.0/57.0 kB 3.0 MB/s eta 0:00:00
Downloading certifi-2024.7.4-py3-none-any.whl (162 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 163.0/163.0 kB 10.5 MB/s eta 0:00:00
Downloading cloudpickle-2.2.1-py3-none-any.whl (25 kB)
Downloading zipp-3.19.1-py3-none-any.whl (9.0 kB)
Downloading async_lru-2.0.5-py3-none-any.whl (6.1 kB)
Downloading ibm_cloud_sdk_core-3.24.1-py3-none-any.whl (75 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.8/75.8 kB 4.3 MB/s eta 0:00:00
Downloading importlib_metadata-8.4.0-py3-none-any.whl (26 kB)
Downloading ipython-8.37.0-py3-none-any.whl (831 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 831.9/831.9 kB 28.9 MB/s eta 0:00:00
Downloading ipywidgets-8.1.7-py3-none-any.whl (139 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.8/139.8 kB 9.2 MB/s eta 0:00:00
Downloading jupyter_lsp-2.2.6-py3-none-any.whl (69 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 69.4/69.4 kB 3.1 MB/s eta 0:00:00
Downloading jupyter_server-2.16.0-py3-none-any.whl (386 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 386.9/386.9 kB 24.9 MB/s eta 0:00:00
Downloading jupyter_client-8.6.3-py3-none-any.whl (106 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.1/106.1 kB 6.5 MB/s eta 0:00:00
Downloading jupyterlab_server-2.27.3-py3-none-any.whl (59 kB)
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.7/59.7 kB 2.4 MB/s eta 0:00:00
Downloading opentelemetry_api-1.33.0-py3-none-any.whl (65 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.8/65.8 kB 3.7 MB/s eta 0:00:00
Downloading opentelemetry_exporter_otlp_proto_grpc-1.33.0-py3-none-any.whl (18 kB)
Downloading opentelemetry_exporter_otlp_proto_common-1.33.0-py3-none-any.whl (18 kB)
Downloading opentelemetry_proto-1.33.0-py3-none-any.whl (55 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.9/55.9 kB 3.7 MB/s eta 0:00:00
Downloading opentelemetry_instrumentation_requests-0.54b0-py3-none-any.whl (12 kB)
Downloading opentelemetry_instrumentation-0.54b0-py3-none-any.whl (31 kB)
Downloading opentelemetry_semantic_conventions-0.54b0-py3-none-any.whl (194 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.9/194.9 kB 13.9 MB/s eta 0:00:00
Downloading opentelemetry_util_http-0.54b0-py3-none-any.whl (7.3 kB)
Downloading opentelemetry_sdk-1.33.0-py3-none-any.whl (118 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.9/118.9 kB 8.5 MB/s eta 0:00:00
Downloading ray-2.48.0-cp311-cp311-manylinux2014_x86_64.whl (70.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 70.1/70.1 MB 7.9 MB/s eta 0:00:00
Downloading requests_ntlm-1.3.0-py3-none-any.whl (6.6 kB)
Downloading rustworkx-0.16.0-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 46.2 MB/s eta 0:00:00
Downloading s3fs-2025.7.0-py3-none-any.whl (30 kB)
Downloading spectate-1.0.1-py2.py3-none-any.whl (11 kB)
Downloading stevedore-5.4.1-py3-none-any.whl (49 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.5/49.5 kB 3.1 MB/s eta 0:00:00
Downloading traitlets-5.14.3-py3-none-any.whl (85 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.4/85.4 kB 5.9 MB/s eta 0:00:00
Downloading aiobotocore-2.23.1-py3-none-any.whl (84 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.2/84.2 kB 5.2 MB/s eta 0:00:00
Downloading comm-0.2.2-py3-none-any.whl (7.2 kB)
Downloading Deprecated-1.2.18-py2.py3-none-any.whl (10.0 kB)
Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 50.9 MB/s eta 0:00:00
Downloading json5-0.12.0-py3-none-any.whl (36 kB)
Downloading jupyter_events-0.12.0-py3-none-any.whl (19 kB)
Downloading jupyter_server_terminals-0.5.3-py3-none-any.whl (13 kB)
Downloading overrides-7.7.0-py3-none-any.whl (17 kB)
Downloading pbr-6.1.1-py2.py3-none-any.whl (108 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 109.0/109.0 kB 6.6 MB/s eta 0:00:00
Downloading py_spy-0.4.0-py2.py3-none-manylinux_2_5_x86_64.manylinux1_x86_64.whl (2.7
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.7/2.7 MB 57.1 MB/s eta 0:00:00
Downloading pyspnego-0.11.2-py3-none-any.whl (130 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 130.5/130.5 kB 9.1 MB/s eta 0:00:00
Downloading symengine-0.13.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.w
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.7/49.7 MB 11.1 MB/s eta 0:00:00
Downloading virtualenv-20.32.0-py3-none-any.whl (6.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.1/6.1 MB 37.6 MB/s eta 0:00:00
Downloading widgetsnbextension-4.0.14-py3-none-any.whl (2.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 59.8 MB/s eta 0:00:00
Downloading aiohttp_cors-0.8.1-py3-none-any.whl (25 kB)
Downloading colorful-0.5.7-py2.py3-none-any.whl (201 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 201.5/201.5 kB 9.4 MB/s eta 0:00:00
Downloading opencensus-0.11.4-py2.py3-none-any.whl (128 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 128.2/128.2 kB 6.1 MB/s eta 0:00:00
Downloading opentelemetry_exporter_prometheus-0.54b0-py3-none-any.whl (12 kB)
Downloading stack_data-0.6.3-py3-none-any.whl (24 kB)
Downloading aioitertools-0.12.0-py3-none-any.whl (24 kB)
Downloading asttokens-3.0.0-py3-none-any.whl (26 kB)
Downloading botocore-1.38.46-py3-none-any.whl (13.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.7/13.7 MB 84.7 MB/s eta 0:00:00
Downloading distlib-0.4.0-py2.py3-none-any.whl (469 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 469.0/469.0 kB 21.3 MB/s eta 0:00:00
```

```
Downloading executing-2.2.0-py2.py3-none-any.whl (26 kB)
Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Downloading opencensus_context-0.1.3-py2.py3-none-any.whl (5.1 kB)
Downloading python_json_logger-3.3.0-py3-none-any.whl (15 kB)
Downloading rfc3986_validator-0.1.1-py2.py3-none-any.whl (4.2 kB)
Downloading pure_eval-0.2.3-py3-none-any.whl (11 kB)
Downloading rfc3339_validator-0.1.4-py2.py3-none-any.whl (3.5 kB)
Downloading rfc3987_syntax-1.1.0-py3-none-any.whl (8.0 kB)
Downloading fqdn-1.5.1-py3-none-any.whl (9.1 kB)
Downloading isoduration-20.11.0-py3-none-any.whl (11 kB)
Downloading uri_template-1.3.0-py3-none-any.whl (11 kB)
Downloading arrow-1.3.0-py3-none-any.whl (66 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.4/66.4 kB 3.7 MB/s eta 0:00:00
Downloading lark-1.2.2-py3-none-any.whl (111 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 111.0/111.0 kB 6.5 MB/s eta 0:00:00
Downloading types_python_dateutil-2.9.0.20250708-py3-none-any.whl (17 kB)
Building wheels for collected packages: qc-grader, pylatexenc
  Building wheel for qc-grader (setup.py) ... done
  Created wheel for qc-grader: filename=qc_grader-0.22.12-py3-none-any.whl size=25190
  Stored in directory: /tmp/pip-ephem-wheel-cache-oaevrw_p/wheels/56/7f/8b/c045ba8136
  Building wheel for pylatexenc (setup.py) ... done
  Created wheel for pylatexenc: filename=pylatexenc-2.10-py3-none-any.whl size=136817
  Stored in directory: /root/.cache/pip/wheels/b1/7a/33/9fdd892f784ed4afda62b685ae370
Successfully built qc-grader pylatexenc
Installing collected packages: pylatexenc, py-spy, pure-eval, opencensus-context, dis
  Attempting uninstall: zipp
    Found existing installation: zipp 3.23.0
    Uninstalling zipp-3.23.0:
      Successfully uninstalled zipp-3.23.0
  Attempting uninstall: widgetsnbextension
    Found existing installation: widgetsnbextension 3.6.10
    Uninstalling widgetsnbextension-3.6.10:
      Successfully uninstalled widgetsnbextension-3.6.10
  Attempting uninstall: traitlets
    Found existing installation: traitlets 5.7.1
    Uninstalling traitlets-5.7.1:
      Successfully uninstalled traitlets-5.7.1
  Attempting uninstall: networkx
    Found existing installation: networkx 3.5
    Uninstalling networkx-3.5:
      Successfully uninstalled networkx-3.5
  Attempting uninstall: jsonpickle
    Found existing installation: jsonpickle 4.1.1
    Uninstalling jsonpickle-4.1.1:
      Successfully uninstalled jsonpickle-4.1.1
  Attempting uninstall: cloudpickle
    Found existing installation: cloudpickle 3.1.1
    Uninstalling cloudpickle-3.1.1:
      Successfully uninstalled cloudpickle-3.1.1
  Attempting uninstall: certifi
    Found existing installation: certifi 2025.7.14
    Uninstalling certifi-2025.7.14:
      Successfully uninstalled certifi-2025.7.14
  Attempting uninstall: importlib-metadata
    Found existing installation: importlib_metadata 8.7.0
    Uninstalling importlib_metadata-8.7.0:
      Successfully uninstalled importlib_metadata-8.7.0
  Attempting uninstall: jupyter-client
    Found existing installation: jupyter-client 6.1.12
    Uninstalling jupyter-client-6.1.12:
      Successfully uninstalled jupyter-client-6.1.12
```

```
      Successfully uninstalled jupyter_client-6.1.12
  Attempting uninstall: ipython
    Found existing installation: ipython 7.34.0
    Uninstalling ipython-7.34.0:
      Successfully uninstalled ipython-7.34.0
  Attempting uninstall: ipywidgets
    Found existing installation: ipywidgets 7.7.1
    Uninstalling ipywidgets-7.7.1:
      Successfully uninstalled ipywidgets-7.7.1
  Attempting uninstall: jupyter-server
    Found existing installation: jupyter-server 1.16.0
    Uninstalling jupyter-server-1.16.0:
      Successfully uninstalled jupyter-server-1.16.0
ERROR: pip's dependency resolver does not currently take into account all the package
google-colab 1.0.0 requires ipython==7.34.0, but you have ipython 8.37.0 which is inc
torch 2.6.0+cu124 requires nvidia-cublas-cu12==12.4.5.8; platform_system == "Linux" a
torch 2.6.0+cu124 requires nvidia-cuda-cupti-cu12==12.4.127; platform_system == "Linu
torch 2.6.0+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127; platform_system == "Linu
torch 2.6.0+cu124 requires nvidia-cuda-runtime-cu12==12.4.127; platform_system == "Li
torch 2.6.0+cu124 requires nvidia-cudnn-cu12==9.1.0.70; platform_system == "Linux" an
torch 2.6.0+cu124 requires nvidia-cufft-cu12==11.2.1.3; platform_system == "Linux" an
torch 2.6.0+cu124 requires nvidia-curand-cu12==10.3.5.147; platform_system == "Linux"
torch 2.6.0+cu124 requires nvidia-cusolver-cu12==11.6.1.9; platform_system == "Linux"
torch 2.6.0+cu124 requires nvidia-cusparse-cu12==12.3.1.170; platform_system == "Linu
torch 2.6.0+cu124 requires nvidia-nvjitlink-cu12==12.4.127; platform_system == "Linux
dask 2025.5.0 requires cloudpickle>=3.0.0, but you have cloudpickle 2.2.1 which is in
notebook 6.5.7 requires jupyter-client<8,>=5.3.4, but you have jupyter-client 8.6.3 w
jupyter-kernel-gateway 2.5.2 requires jupyter-client<8.0,>=5.2.0, but you have jupyte
distributed 2025.5.0 requires cloudpickle>=3.0.0, but you have cloudpickle 2.2.1 whic
Successfully installed aiobotocore-2.23.1 aiohttp_cors-0.8.1 aioitertools-0.12.0 arro
WARNING: Upgrading ipython, ipykernel, tornado, prompt-toolkit, pyzmq can
cause your runtime to repeatedly crash or behave in unexpected ways and is not
recommended. If your runtime won't connect or execute code, you can reset it
with "Disconnect and delete runtime" from the "Runtime" menu.

WARNING: The following packages were previously imported in this runtime:
  [IPython,certifi,importlib_metadata,traitlets,zipp]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
Collecting pyscf
  Downloading pyscf-2.9.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.met
Requirement already satisfied: numpy!=1.16,!=1.17,>=1.13 in /usr/local/lib/python3.11
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: h5py>=2.7 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
Downloading pyscf-2.9.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (50.9
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.9/50.9 MB 9.9 MB/s eta 0:00:00
Installing collected packages: pyscf
Successfully installed pyscf-2.9.0
Collecting ffsim
  Downloading ffsim-0.0.57-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.m
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: orjson in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: pyscf>=2.9 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: qiskit>=1.3 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: h5py>=2.7 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: rustworkx>=0.15.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: stevedore>=3.0.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.11/dist-packages
Downloading ffsim-0.0.57-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 63.0 MB/s eta 0:00:00
Installing collected packages: ffsim
Successfully installed ffsim-0.0.57
Collecting qiskit_addon_sqd
  Downloading qiskit_addon_sqd-0.11.0-py3-none-any.whl.metadata (26 kB)
Requirement already satisfied: jax>=0.4.30 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: jaxlib>=0.4.30 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: numpy>=1.26 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: pyscf>=2.5 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: qiskit<3,>=1.2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: scipy>=1.13.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: ml_dtypes>=0.4.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: opt_einsum in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: h5py>=2.7 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: rustworkx>=0.15.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: stevedore>=3.0.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.11/dist-packages
Downloading qiskit_addon_sqd-0.11.0-py3-none-any.whl (34 kB)
Installing collected packages: qiskit_addon_sqd
Successfully installed qiskit_addon_sqd-0.11.0
```

```
%pip install pyscf
%pip install ffsim
%pip install qiskit_addon_sqd
```

```
Collecting pyscf
  Using cached pyscf-2.9.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.me
Requirement already satisfied: numpy!=1.16,!=1.17,>=1.13 in /usr/local/lib/python3.11
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: h5py>=2.7 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
Using cached pyscf-2.9.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (50.
Installing collected packages: pyscf
Successfully installed pyscf-2.9.0
Collecting ffsim
  Using cached ffsim-0.0.57-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: orjson in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: pyscf>=2.9 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: qiskit>=1.3 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: h5py>=2.7 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: rustworkx>=0.15.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: stevedore>=3.0.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.11/dist-packages
Using cached ffsim-0.0.57-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1
Installing collected packages: ffsim
Successfully installed ffsim-0.0.57
Collecting qiskit_addon_sqd
  Downloading qiskit_addon_sqd-0.11.0-py3-none-any.whl.metadata (26 kB)
Requirement already satisfied: jax>=0.4.30 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: jaxlib>=0.4.30 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: numpy>=1.26 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: pyscf>=2.5 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: qiskit<3,>=1.2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: scipy>=1.13.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: ml_dtypes>=0.4.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: opt_einsum in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: h5py>=2.7 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: rustworkx>=0.15.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: stevedore>=3.0.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.11/dist-packages
Downloading qiskit_addon_sqd-0.11.0-py3-none-any.whl (34 kB)
Installing collected packages: qiskit_addon_sqd
Successfully installed qiskit_addon_sqd-0.11.0
```

```
import qiskit
import qc_grader
print(f"Qiskit version: {qiskit.__version__}")
print(f"Grader version: {qc_grader.__version__}")
```

```
Qiskit version: 2.1.1
Grader version: 0.22.12
```

```
# Save your API key to track your progress and have access to the quantum computers

your_api_key = "cCGO9IST5-SYZXfgICgB5msY3coo1NGuujE9gt9zTyBx"
your_crn = "crn:v1:bluemix:public:quantum-computing:us-east:a/df3fd2dfbb6a49c98d710485c6d

from qiskit_ibm_runtime import QiskitRuntimeService

QiskitRuntimeService.save_account(
    channel="ibm_quantum_platform",
    token=your_api_key,
    instance=your_crn,
    name="qgss-2025",
    overwrite=True
)
```

You should have Qiskit version `>=2.0.0` and Grader `>=0.22.12`. If you see a lower version, you need to reinstall the grader and restart the kernel. Also make sure you have set up everything according to lab 0 and test it with the cell below.

```
# Check that the account has been saved properly
from qiskit_ibm_runtime import QiskitRuntimeService

service = QiskitRuntimeService(name="qgss-2025")
service.saved_accounts()
```

```
{'qgss-2025': {'channel': 'ibm_quantum_platform',
  'url': 'https://cloud.ibm.com',
  'token': 'cCGO9IST5-SYZXfgICgB5msY3coo1NGuujE9gt9zTyBx',
  'instance': 'crn:v1:bluemix:public:quantum-computing:us-
east:a/df3fd2dfbb6a49c98d710485c6d2e892:3ff20315-1dd8-4e6a-9adc-6e01711ed8ca::',
  'verify': True,
  'private_endpoint': False}}
```

Imports

```
# Import common packages first
import numpy as np
from math import comb
import warnings
import pyscf
import matplotlib.pyplot as plt
import pickle
from functools import partial

# Import qiskit classes
from qiskit import QuantumCircuit, QuantumRegister
```

```
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit.visualization import plot_gate_map
from qiskit_addon_sqd.fermion import SCIResult, diagonalize_fermionic_hamiltonian, solve_

# Import qiskit ecosystems
import ffsim
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2 as Sampler
from qiskit_ibm_runtime import SamplerOptions

# Import grader
from qc_grader.challenges.qgss_2025 import (
    grade_lab3_ex1,
    grade_lab3_ex2,
    grade_lab3_ex3,
    grade_lab3_ex4,
    grade_lab3_ex5
)
```

1. Introduction

1.1 Objective

The objective of this Lab is to learn the basics and general workflow of quantum chemistry calculations. You will also learn about a useful hybrid quantum-classical algorithm called Sample-based Quantum Diagonalization (SQD) algorithm. SQD is a classical post-processing technique which acts on samples obtained from a quantum circuit after execution on a QPU. It is useful for finding eigenvalues and eigenvectors of quantum operators, such as the Hamiltonian of a quantum system, and uses quantum and distributed classical computing together.

1.2 What Are Atoms?

Everything around you is made of atoms—the tiny building blocks of matter. The word “atom” comes from a Greek word that means “can’t be split.” A long time ago, a man named Democritus thought that all things were made of tiny, unbreakable pieces called atoms.

In 1913, Niels Bohr said electrons move in circles (orbits) around the center of the atom, like planets around the sun. But in 1926, Erwin Schrödinger said electrons don’t really move in perfect paths. Instead, they buzz around in "clouds" where they’re most likely to be found.

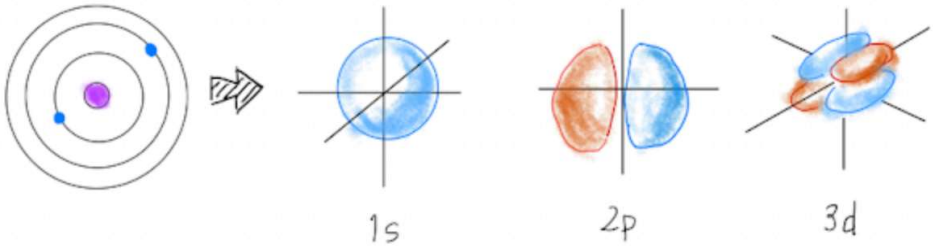


Fig 1. A sketch of Bohr's atomic model (left) and Schrödinger's atomic model based on quantum mechanical and wave nature of electrons (right).

These clouds (called orbitals) have different shapes and energy levels. Electrons usually stay in the lowest energy level (the ground state), but they can jump to higher ones if they get energy. When they fall back down, they give off energy.

1.3 The Schrödinger Equation

Erwin Schrödinger's contributions to quantum mechanics go beyond introducing a new electronic model; he established the famous Schrödinger equation. The time-independent Schrödinger equation is:

$$\hat{H}|\psi\rangle = E|\psi\rangle$$

$\hat{H}$: Hamiltonian operator
$|\psi\rangle$: Wave function (state of the system)
E : Measured energy (eigenvalue)

The goal of quantum chemistry is to solve the Schrödinger Equation for the wave function. The wave function that satisfies the Schrödinger Equation can help us find out interesting properties of that quantum system such as energy, momentum, spin, magnetization, and more.

In quantum chemistry, we are often interested in finding **the ground state energy**. This is because once we know the ground state energy of a molecule, we can learn a lot such as:

- the shape a molecule will most likely take in a stable form. (often called the "equilibrium" state)
- how molecules might change or react during chemical reactions.
- how a drug might stick to a protein in the body seen in Docking simulations.

In short, finding the ground state is like finding the starting point for all the interesting things molecules can do.

But for big molecules, **solving the Schrödinger Equation exactly is extremely difficult** because the wave function, which describes the spatial distribution of electrons, is highly complex. So, instead of solving it perfectly, scientists make good guesses or approximations that are close enough.

1.4 Basis Set Approximation - Smart Building

One of the key approximations in quantum chemistry is the use of a **basis set** approach. A basis set is **a collection of mathematical functions used to approximate the orbitals (wavefunctions) of electrons** in atoms and molecules. Since we can't solve the Schrödinger equation exactly for most systems, we use basis sets to make the problem computationally tractable. Instead of working with full, continuous atomic orbitals, we approximate them using a set of predefined mathematical functions—usually Gaussian-type or Slater-type orbitals.

This method is known as the Linear Combination of Atomic Orbitals (LCAO). The idea is simple: we build molecular orbitals by combining basis functions. It's like building a complex shape using a sum of simpler, known shapes.

- A small basis set (fewer functions) gives a rough but fast approximation.
- A larger basis set improves accuracy but requires much more computational power.

Types of Basis Functions

- **Slater-type orbitals (STOs)**: These are mathematical functions that look very similar to real atomic orbitals (the shapes electrons naturally form around atoms). They describe how electron density drops off with distance from the nucleus.
- **Gaussian-type orbitals (GTOs)**: Easier to work with computationally, even though they don't resemble orbitals as closely. Often used in practice.

The Slater-type orbital (STO) can be approximated by a combination of Gaussian-type orbitals (GTOs). In other words, we combine multiple Gaussians with different widths to mimic the shape of an STO. This combination is called a **contracted Gaussian function**. This idea is the basis for different kinds of basis sets, like minimal (simple) and split-valence (more accurate) sets.

- **Minimal Basis Set**: A basis set that uses just one function per orbital (e.g., one function for $1s$, one for $2s$, etc.). In STO-3G, for example, each STO is approximated using 3 Gaussians.
- **Split-Valence Basis Set**: These are more flexible and provide more accurate representations than minimal basis sets. They split the valence orbitals (the outermost orbital that contain electrons involved in bonding with other atoms) into two or more parts, each approximated with different combinations of Gaussians (like 6-31G). This allows for better accuracy in chemical bonding and reactions.

Common Basis Sets

Basis Set	Description
STO-3G	Uses 3 Gaussians (the "3G") to mimic 1 STO. (aka 'Minimal basis')
6-31G	Uses 6 Gaussians to mimic 1 STO (the "6") for each core orbital. For each valence (outer) orbital, a contracted functi
cc-pVDZ	Designed to improve electron correlation calculations. Adds polarization functions. Uses 2 basis functions for each

> NOTE: The computational cost referred to in the table above is the classical cost of computing the molecular integrals. The more functions in your basis set, the

> more accurate your results—but computations take longer.

To summarize, instead of solving the full Schrödinger equation exactly (which is almost always impossible for molecules), we build a flexible, approximate wave function using known functions—and then tune it to minimize the system's energy.

1.5 The Hamiltonian

At the heart of the Schrödinger equation, and to that extent in every quantum chemistry problem, there is a **Hamiltonian**. It's a central object in both classical mechanics and quantum mechanics, and is essentially a function that represents the total energy (kinetic + potential) of a physical system.

In quantum mechanics, the Hamiltonian $\hat{H}$ becomes an operator acting on the wave function $|\psi\rangle$. In a chosen basis, these wave functions $|\psi\rangle$ can be described as vectors and Hamiltonians in matrix form.

In most quantum chemistry problems, the first and most important goal was to calculate its ground state energy. This calculation can be done by diagonalizing a Hamiltonian (matrix) and computing its eigenvalues and eigenvectors.

The size or dimensions of a molecular Hamiltonian can be determined by finding the number of combinations the electrons can occupy the spatial orbitals. This is essentially a combination problem that can quickly grow exponential in size making the diagonalization uncontractable for most interesting molecules.

Example: Calculate the size of the N_2 Hamiltonian

We would like you to get a sense of how large a Hamiltonian matrix H can grow, even for a relatively small molecule like nitrogen (N_2), depending on how accurately you want to simulate it. In the following example, we chose the number of spatial orbitals, spin orbitals, and electrons of N_2 to use in our calculations.

Using the number of spatial orbitals and electrons in the table below, let's compute the number of ways the electrons can occupy the spatial orbitals (spin configurations), which indicates the size of the N_2 molecule Hamiltonian.

	STO-3G	6-31G	cc-pVDZ
Spatial orbitals	(10) **8**	(18) **16**	(28) **26**
Spin orbitals	(20) **16**	(36) **32**	(56) **52**
α-spin electrons	(7) **5**	(7) **5**	(7) **5**
β-spin electrons	(7) **5**	(7) **5**	(7) **5**

Table 1: Number of orbitals and electrons in specified basis sets for N_2

- (#) : number of orbitals and electrons we specifically chose for the purpose of this example.
- # : number of orbitals or electrons when freezing the core orbital (*1s*) and reducing the number of electrons and spatial orbitals each by 2.

⚠️ **Note:** In this example we treat the $1s$ (core) orbital as chemically inactive. This means we can freeze the core orbital and save 2 electrons and 2 spatial orbitals (i.e. 4 spin orbitals -> 4 qubits). This is another useful approximation technique you will frequently see in actual chemistry simulations. Applying this technique, please make sure to use the numbers in **bold** for calculating for all possible electron configurations for the N_2 Hamiltonian.

As you can tell, solving for all possible spin configurations is essentially a combination problem. Let's take a look at the math to calculate how many ways the α**-spin (spin-up)** electrons and β**-spin (spin-down)** electrons can each occupy given spatial orbitals. For the total spin configurations we simply need to multiply them together.

Total electron configurations = (total α-spin configurations) × (total β-spin configurations)

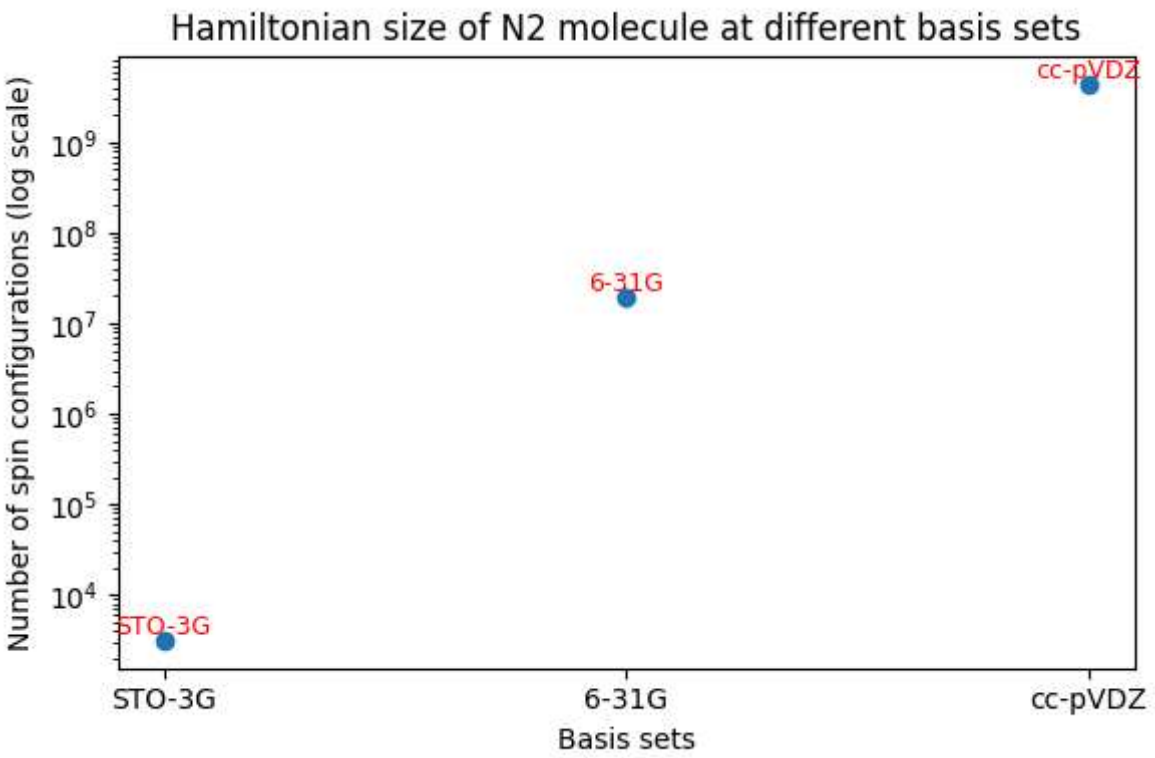
$$nCn_\alpha \times nCn_\beta = \frac{n!}{n_\alpha!(n-n_\alpha)!} \times \frac{n!}{n_\beta!(n-n_\beta)!}$$

Where n: number of spatial orbitals, n_α: number of α-spins, n_β: number of β-spins

Now that we know how to obtain the total spin configurations, let's calculate this in each basis set and plot the results to see how the size of the Hamiltonian grows with more accuracy.

```
# Number of possible spin configurations
# Example: N2 molecule in STO-3G, 6-31G, and cc-pVDZ basis sets
# 14 electrons, 20 spin orbitals (from 10 spatial orbitals × 2)

# Calculate total electron configurations for each basis set
y1 = comb(8, 5) * comb(8, 5)       # STO-3G
y2 = comb(16, 5) * comb(16, 5)     # 6-31G
y3 = comb(26, 5) * comb(26, 5)     # cc-pVDZ

# Data
y = [y1, y2, y3]
x = list(range(len(y)))
labels = ['STO-3G', '6-31G', 'cc-pVDZ']

# Plot with logarithmic y-scale
plt.figure(figsize=(6, 4))
plt.plot(x, y, 'o')

plt.yscale('log')
plt.xticks(x, labels)
plt.xlabel('Basis sets')
```

```
    plt.ylabel('Number of spin configurations (log scale)')
    plt.title('Hamiltonian size of N2 molecule at different basis sets')

    # Add labels above points
    for i in range(len(x)):
        plt.text(x[i], y[i], f'{labels[i]}', fontsize=9, ha='center', va='bottom', color='red

    plt.tight_layout()
    plt.show()
```



Note that the Y-axis above is in logarithmic scale. You can see how the number of spin configurations grows exponentially with a basis set choice of better approximation.

Exercise 1: Measure the size of the O_2 Hamiltonian

Now let's calculate the size of oxygen O_2 in the `6-31G` basis. Oxygen has the same number of orbitals as N_2 but has 2 additional α**-spin (spin-up)** electrons. The numbers you need to use in this exercise are provided in the table below.

Exercise 1: Measure the size of the O_2 Hamiltonian

Using the number of spatial orbitals and electrons in the table below, compute the total number of electron spin configurations for the oxygen O_2 molecule in the `6-31G` basis. **You may write your own code or calculate by hand to get the answer.**

	STO-3G	6-31G	cc-pVDZ
Spatial orbitals	(10) **8**	(18) **16**	(28) **26**
Spin orbitals	(20) **16**	(36) **32**	(56) **52**

	STO-3G	6-31G	cc-pVDZ
α-spin electrons	(9) **7**	(9) **7**	(9) **7**
β-spin electrons	(7) **5**	(7) **5**	() **5**

Table 1: Number of orbitals and electrons in specified basis sets for O_2

- (#) : number of orbitals or electrons based on actual atomic structure in the specified basis set.
- # : number of orbitals or electrons when freezing the core orbital (*1s*) and reducing the number of electrons and spatial orbitals each by 2.

⚠️ **Note:** As we already saw in the previous example, we will treat the $1s$ (core) orbital as chemically inactive in this exercise. This means we can freeze the core orbital and save 2 electrons and 2 spatial orbitals (i.e. 4 spin orbitals -> 4 qubits). This is another useful approximation technique you will frequently see in actual chemistry simulations. Applying this technique, please make sure to use the numbers in **bold** for calculating for all possible electron configurations for the O_2 Hamiltonian.

```
# Exercise 1: Number of possible spin configurations
# Example: O2 molecule in 6-31G basis
# 16 electrons, 20 spin orbitals (from 10 spatial orbitals × 2)

# Calculate all valid electron configurations
# Hint: This is a combinatorial problem. You can calculate by hand and provide the answer

# ---- TODO : Task 1 ---
### Provide your code below to calculate the total configurations

α_config = comb(16, 7)
β_config = comb(16, 5)
total_config = (α_config)*(β_config)
# --- End of TODO ---

print(f"Total physical configurations for O2 in the given basis : {α_config:} x {β_config
```

```
Total physical configurations for O2 in the given basis : 11440 x 4368 = 49969920
```

```
# total_config = 49969920 #provide your answer here if calculated by hand. Then uncomment


# Submit your answer using following code

grade_lab3_ex1(total_config) # Expected result type: integer
```

```
Submitting your answer. Please wait...
Congratulations 🎉! Your answer is correct and has been submitted.
```

After submitting your answer calculated in the 6-31G basis, try calculating the same in other basis sets (e.g. STO-3G, and cc-pVDZ) for comparison. Feel free to plot your results to see how

the size grows.

2. Variational Quantum Eigensolver

So far, we've learned about the Schrödinger Equation, basis sets, and how the Hamiltonian of a physical system can grow exponentially in size depending on how accurately you want to simulate your system. Now let's talk about algorithms.

One of the well-known algorithms among computational chemists who work with near-term quantum computers is perhaps the **Variational Quantum Eigensolver (VQE)**.

The Variational Quantum Eigensolver (VQE) is a hybrid quantum-classical algorithm used to optimize a cost function Hamiltonian based on the variational principle. Although this lab's main focus is on learning about a different kind of hybrid quantum-classical algorithm called Sample-based Quantum Diagonalization (SQD), it would be useful to briefly review the components of VQE as they form some of the important building blocks for SQD as well.

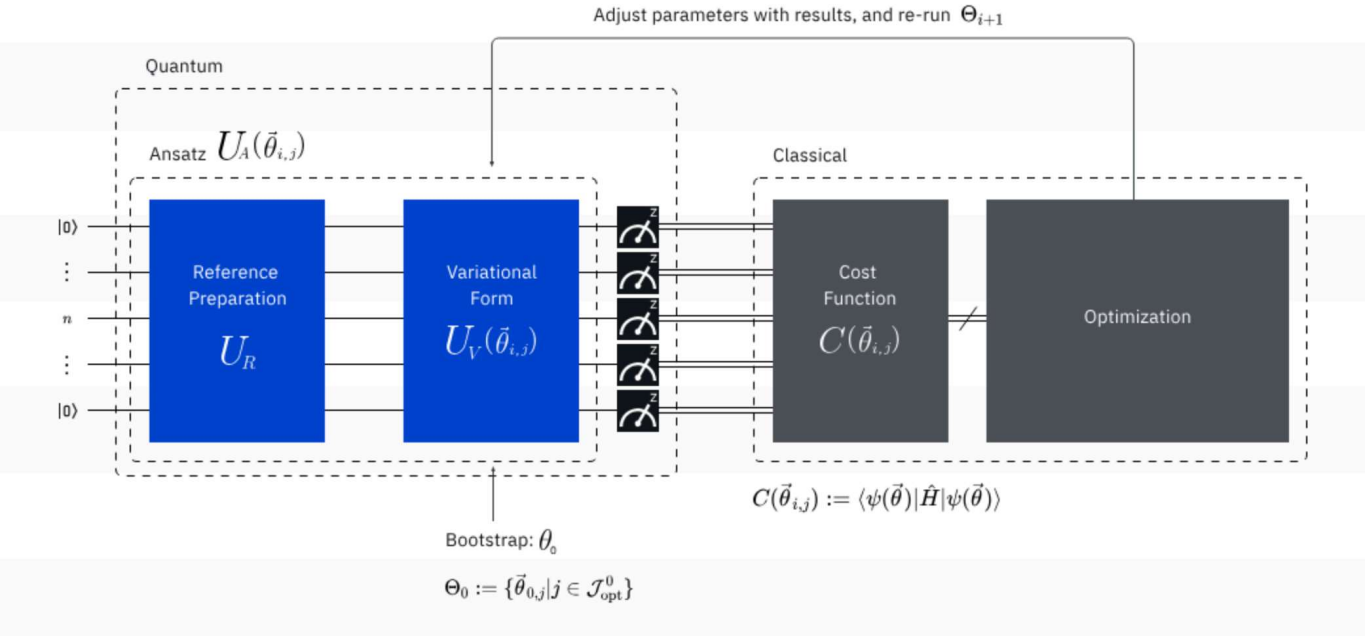


Fig 2. A diagram that shows the computational workflow of VQE

Summary of Computational Steps in VQE:

1. Prepare the quantum state $|\psi(\theta)\rangle$(Quantum).
2. Measure expectation values $\langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle$(Quantum).
3. Compute the cost function $E(\theta)$(Classical).
4. Adjust the parameter θ using classical optimization (Classical).
5. Repeat the steps until cost function $E(\theta)$ converges.

As you can tell from the above, the computational steps for executing VQE is an 'iterative' process - you need to repeat these steps many, many, times. This is a computationally costly

procedure and becomes a limitation for an algorithm like VQE to scale and work with larger molecules.

3. What is Sample-based Quantum Diagonalization (SQD)?

While VQE has been widely explored since its original proposal in 2014, it has certain limitations. As the Hamiltonian grows larger, obtaining the expectation value through the variational method becomes increasingly resource-intensive, and the algorithm does not scale well for larger molecules. This challenge motivates the study and implementation of Sample-based Quantum Diagonalization (SQD).

SQD was inspired by a hybrid quantum-classical method called quantum-selected configuration interaction (QSCI) introduced and published in this paper. Similar to QSCI, SQD is also a hybrid quantum-classical algorithm in which a quantum computer is used to generate electronic configurations by sampling them from a quantum circuit, and then those configurations are used to form a subspace in which to project and diagonalize the electronic Hamiltonian.

This subspace will have dimensions that are smaller than the original Hamiltonian making it much easier to classically diagonalize.

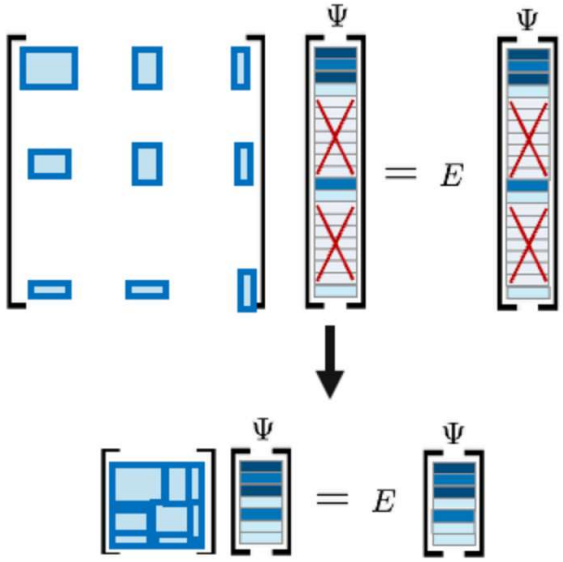


Fig 3. An illustration of the original Hamiltonian shown as an exponentially large matrix in the Hilbert space of N qubits then reduced to a smaller size after configuration recovery and subsampling.

Let's say we have three atomic orbitals 1s, 2p, 3d, and two electrons that can occupy these orbitals that are ranked from low to high energy. Low energy configurations mean most electrons occupy the lower orbitals while high energy configurations will have electrons in the higher orbitals. One of the important assumptions of the SQD algorithm is that high-energy

configurations will have little contributions to the ground state. This allows us to remove these less relevant configurations so we can focus on a subspace of relevant configurations only - in our case configurations that show occupancies of electrons in the lower orbitals.

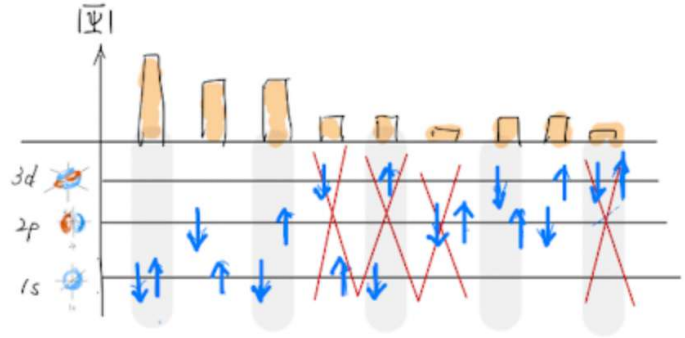


Fig 4. Removing electron configurations that are not relevant to the ground state allows us to reduce the size of the matrix (Hamiltonian)

This reconstruction of Hamiltonian to a smaller size significantly reduces the time to arrive at the solution.

Choosing relevant configurations

For identifying the relevant electron configuration, we use a quantum circuit (ansatz) denoted by $|\psi\rangle$ in this illustration, which upon measurement in the computational basis will give us a probability distribution over the Hilbert space to sample necessary bitstrings from.

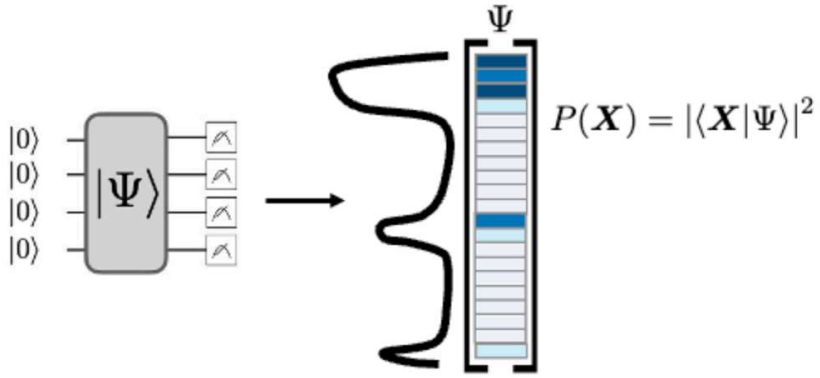
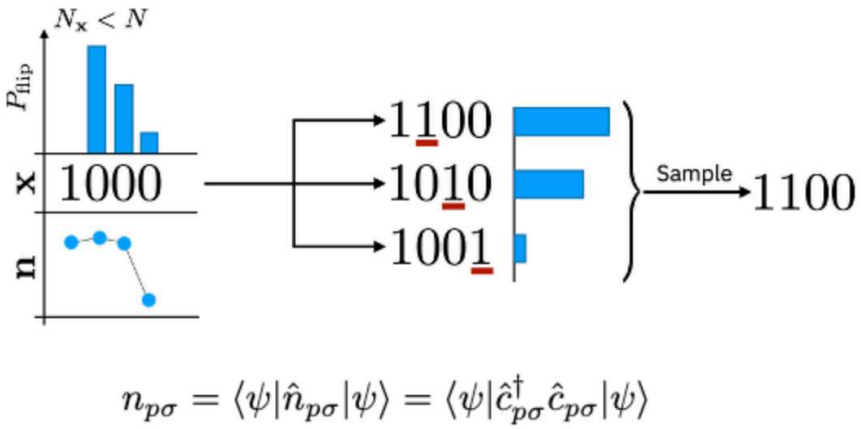


Fig 5. A quantum 'ansatz' circuit will produce the bitstrings we sample from

Dealing with the effects of noise with configuration recovery

Before we subsample from the pool of bitstrings generated by the quantum circuit, we need to deal with 'noisy quantum samples' that may affect the accuracy of our energy calculations. This is where configuration recovery comes into play. Let's say we have a bitstring that is missing one electron from its configuration. This is easy to identify as the hamming weight is wrong. We

can also exploit the expectation of occupancies in the different orbitals (given by 'n' in the illustration below) to determine which bit to flip to correct the bitstrings.



$$n_{p\sigma} = \langle\psi|\hat{n}_{p\sigma}|\psi\rangle = \langle\psi|\hat{c}^\dagger_{p\sigma}\hat{c}_{p\sigma}|\psi\rangle$$

Fig 6. By looking at the electron numbers and occupancy expectation, we can determine which bit to flip

Having reconstructed your Hamiltonian with good samples, you now have a reduced size matrix to diagonalize. Upon diagonalizing the Hamiltonian in the subspace, what the diagonalization subroutine does is assign a non-zero wavefunction amplitude to the computational basis states that contribute to the ground state of the problem. The same routine will assign a zero weight to those bit strings that do not represent the ground state. From the results of all batches, the SQD algorithm obtains the orbital occupancy by electrons and lowest energy estimation and updates the data for the next configuration recovery loop.

4. How to simulate an N_2 molecule with SQD

In this section, we will demonstrate how to post-process noisy quantum samples to find an approximation to the ground state of a chemistry Hamiltonian: the N_2 molecule at equilibrium in the 6-31G basis set. We will follow a SQD approach to process samples taken from a 36-qubit quantum circuit ansatz (in this case, an LUCJ circuit). In order to account for the effect of quantum noise, the configuration recovery technique is used.

Molecular Hamiltonian

The properties of molecules are largely determined by the structure of the electrons within them. As fermionic particles, electrons can be described using a mathematical formalism called second quantization. The idea is that there are a number of *orbitals*, each of which can be either empty or occupied by a fermion. A system of N orbitals is described by a set of fermionic annihilation operators $\{\hat{a}_p\}_{p=1}^N$ that satisfy the fermionic anticommutation relations,

$$\hat{a}_p\hat{a}_q + \hat{a}_q\hat{a}_p = 0,$$
$$\hat{a}_p\hat{a}_q^\dagger + \hat{a}_q^\dagger\hat{a}_p = \delta_{pq}.$$

The adjoint $\hat{a}_p^\dagger$ is called a creation operator.

So far, our exposition has not accounted for spin, which is a fundamental property of fermions. When accounting for spin, the orbitals come in pairs called *spatial orbitals*. Each spatial orbital is composed of two *spin orbitals*, one that is labeled "spin-α" and one that is labeled "spin-β". We then write $\hat{a}_{p\sigma}$ for the annihilation operator associated with the spin-orbital with spin σ ($\sigma \in \{\alpha, \beta\}$) in spatial orbital p. If we take N to be the number of spatial orbitals, then there are a total of $2N$ spin-orbitals. The Hilbert space of this system is spanned by 2^{2N} orthonormal basis vectors labeled with two-part bitstrings $|z\rangle = |z_\beta z_\alpha\rangle = |z_{\beta,N} \cdots z_{\beta,1} z_{\alpha,N} \cdots z_{\alpha,1}\rangle$.

The Hamiltonian of a molecular system can be written as

$$\hat{H} = \sum_{\substack{pr \\ \sigma}} h_{pr}\, \hat{a}_{p\sigma}^\dagger \hat{a}_{r\sigma} + \frac{1}{2} \sum_{\substack{prqs \\ \sigma\tau}} h_{prqs}\, \hat{a}_{p\sigma}^\dagger \hat{a}_{q\tau}^\dagger \hat{a}_{s\tau} \hat{a}_{r\sigma}\,,$$

where the h_{pr} and h_{prqs} are complex numbers called molecular integrals that can be calculated from the specification of the molecule using a computer program. In this tutorial, we compute the integrals using the PySCF software package.

For details about how the molecular Hamiltonian is derived, consult a textbook on quantum chemistry (for example, *Modern Quantum Chemistry* by Szabo and Ostlund).

Local unitary cluster Jastrow (LUCJ) ansatz

SQD requires a quantum circuit ansatz to draw samples from. In this lab, we'll use the local unitary cluster Jastrow (LUCJ) ansatz [1] due to its combination of physical motivation and hardware-friendliness.

The LUCJ ansatz is a specialized form of the general unitary cluster Jastrow (UCJ) ansatz, which has the form

$$|\Psi\rangle = \prod_{\mu=1}^{L} e^{\hat{K}_\mu} e^{i\hat{J}_\mu} e^{-\hat{K}_\mu} |\Phi_0\rangle$$

where $|\Phi_0\rangle$ is a reference state, often taken to be the Hartree-Fock state, and the $\hat{K}_\mu$ and $\hat{J}_\mu$ have the form

$$\hat{K}_\mu = \sum_{pq,\sigma} K_{pq}^\mu\, \hat{a}_{p\sigma}^\dagger \hat{a}_{q\sigma}\ ,\ \hat{J}_\mu = \sum_{pq,\sigma\tau} J_{pq,\sigma\tau}^\mu\, \hat{n}_{p\sigma} \hat{n}_{q\tau}\ ,$$

where we have defined the number operator

$$\hat{n}_{p\sigma} = \hat{a}_{p\sigma}^\dagger \hat{a}_{p\sigma}.$$

The operator $e^{\hat{K}_\mu}$ is an orbital rotation, which can be implemented using known algorithms in linear depth and using linear connectivity. Implementing the $e^{i\mathcal{J}_k}$ term of the UCJ ansatz requires either all-to-all connectivity or the use of a fermionic swap network, making it challenging for noisy pre-fault-tolerant quantum processors that have limited connectivity. The

idea of the *local* UCJ ansatz is to impose sparsity constraints on the $\mathbf{J}^{\alpha\alpha}$ and $\mathbf{J}^{\alpha\beta}$ matrices which allow them to be implemented in constant depth on qubit topologies with limited connectivity. (Here, $\mathbf{J}^{\alpha\alpha} = J^1_{pq,\alpha\alpha}$ and $\mathbf{J}^{\alpha\beta} = J^1_{pq,\alpha\beta}$.)

The IBM hardware has a heavy-hex lattice qubit topology, in which case we can adopt a "zigzag" pattern, depicted below. In this pattern, orbitals with the same spin are mapped to qubits with a line topology (red and blue circles), and a connection between orbitals of different spin is present at every 4th spatial orbital, with the connection being facilitated by an ancillary qubit (purple circles). In this case, the index constraints are

$$\mathbf{J}^{\alpha\alpha} : \{(p, p+1)\ ,\ p = 0, \ldots, N-2\}$$
$$\mathbf{J}^{\alpha\beta} : \{(p, p)\ ,\ p = 0, 4, 8, \ldots,\ p \leq N-1\}$$

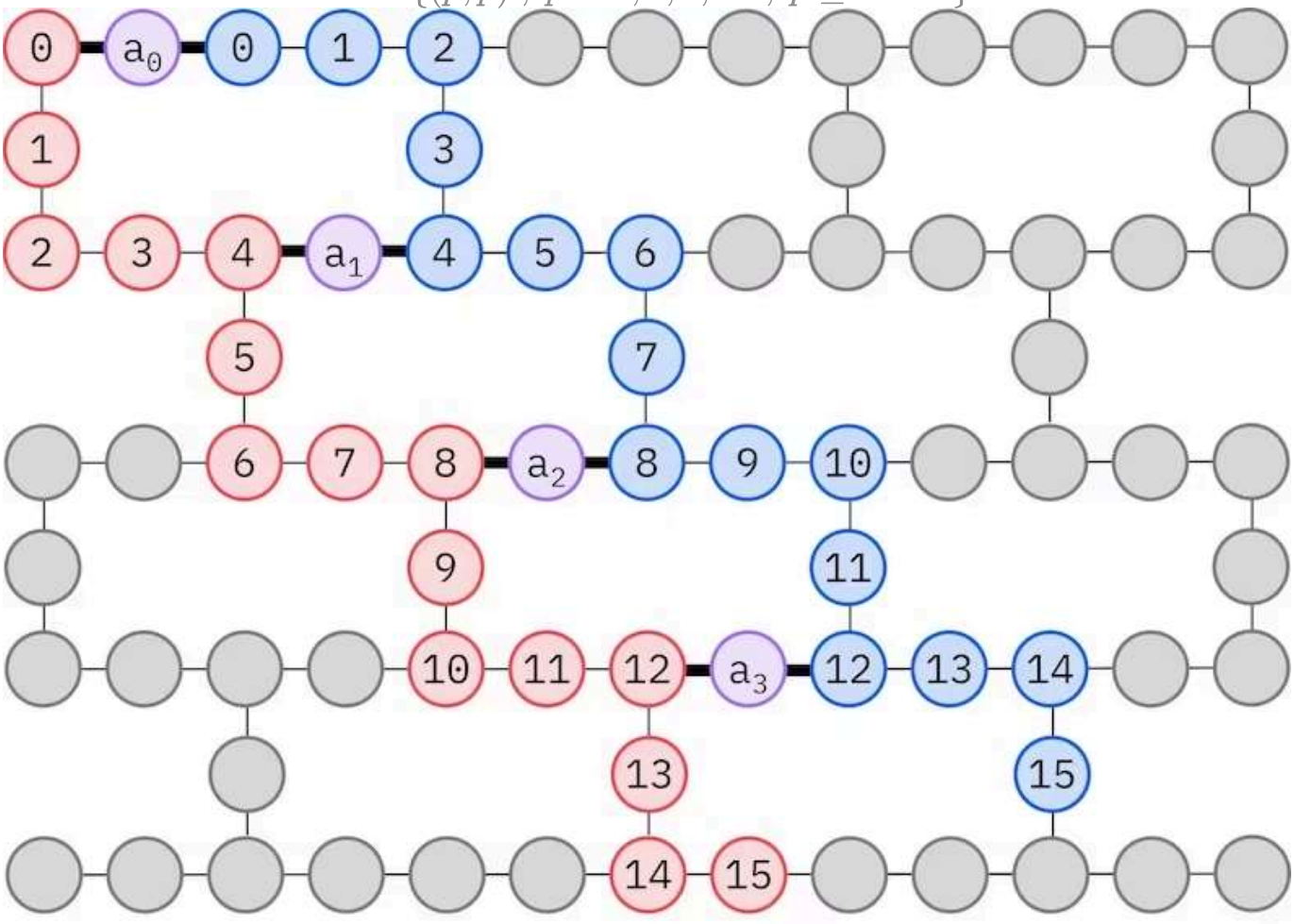


Fig 7. IBM's quantum hardware with a heavy-hex lattice qubit topology

Sample-based Quantum Diagonalization (SQD)

The self-consistent configuration recovery procedure is designed to extract as much signal as possible from noisy quantum samples. The procedure is run in a loop, and each iteration has the following steps:

1. **Recover the configuration**: For each bitstring that violates the specified symmetries, flip its bits with a probabilistic procedure designed to bring the bitstring closer to the current

estimate of the average orbital occupancies, to obtain a new bitstring.

2. **Subsample**: Collect all of the old and new bitstrings that satisfy the symmetries, and subsample subsets of a fixed size, chosen in advance.
3. **Diagonalize in subspace**: For each subset of bitstrings, project the Hamiltonian into the subspace spanned by the corresponding basis vectors and compute a ground state estimate of the projected Hamiltonian on a classical computer.
4. **Find the lowest energy**: Update the estimate of the average orbital occupancies with the ground state estimate with the lowest energy.

The SQD workflow is depicted in the following diagram:

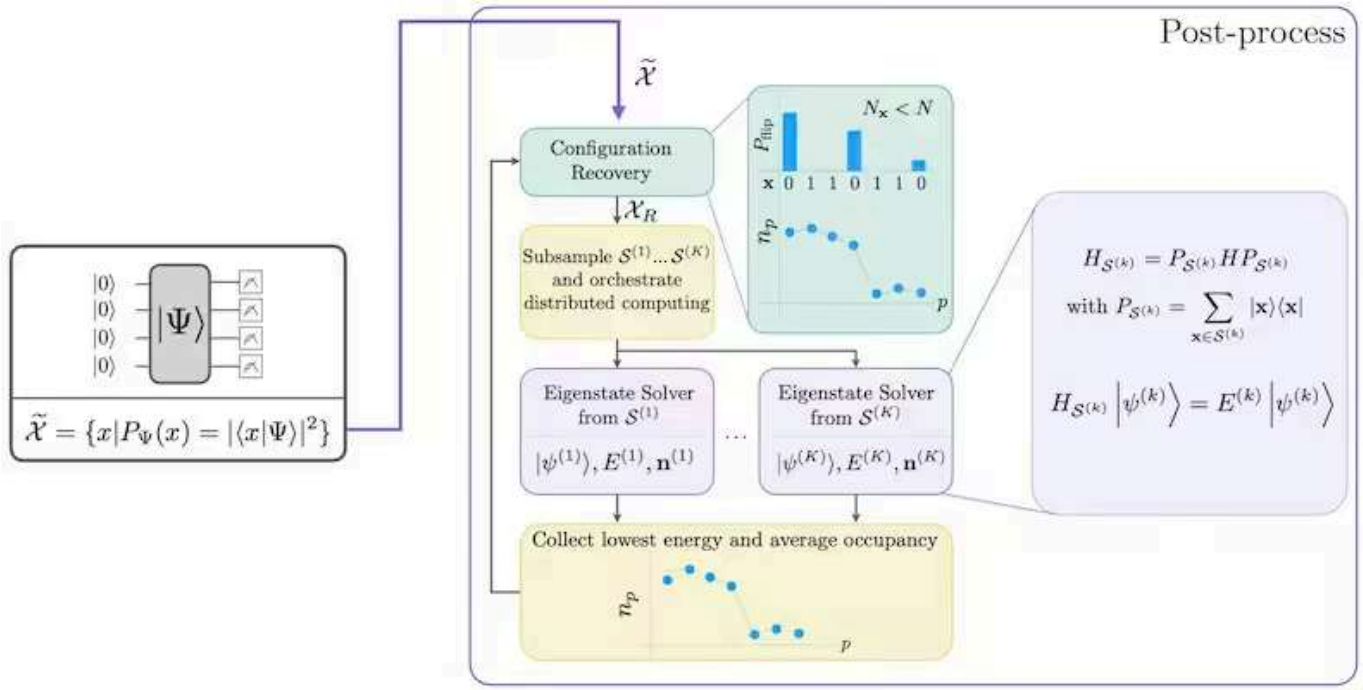


Fig 8. A diagram showing the SQD workflow

SQD is known to work well when the target eigenstate is sparse: the wave function is supported in a set of basis states $\mathcal{S} = \{|x\rangle\}$ whose size does not increase exponentially with the size of the problem.

Configuration recovery loop 1: Recover the configuration

The first step in the configuration recovery loop is to recover the configuration. In other words, error mitigation is performed in this part.

If you run the quantum circuit that is the LUCJ ansatz built above on a quantum computer, you will get a bitstring and its count number as shown in the figure on the lower left. Because current noisy quantum computers give the result with the errors. Since the particle number of "spin-α" and "spin-β" of the molecule in problem is fixed, we will correct the error by flipping a part of the resulting bitstring so that the particle number is stored correctly.

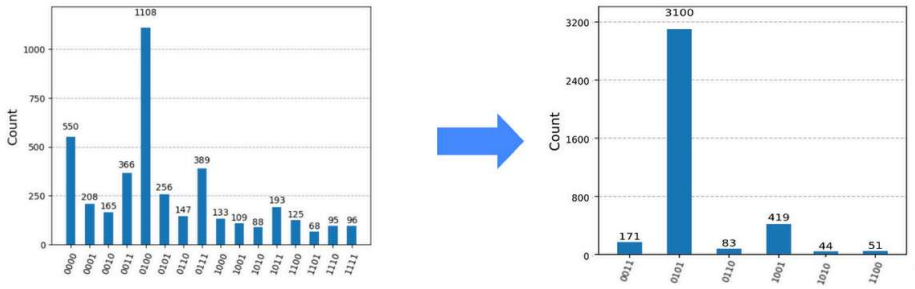


Fig 9. Bitstrings that are initially produced by the LUCJ ansatz circuit (left) and the corrected bitstrings (right)

For example, let's think about a problem that considers up to four spin orbitals for a molecule with two "spin-α" and two "spin-β". If the orbitals are filled from the bottom of energy levels, the quantum state is |0011 0011> as shown in the figure on the lower left. If the result of running the quantum computer contains |1110 0011> then three "spin-β" is too many, so one of the bits is flipped from 1 to 0 so that there are two. Also, if |0011 0001> is observed, one "spin-α" is missing, so one of these three electrons is inverted from 0 to 1.

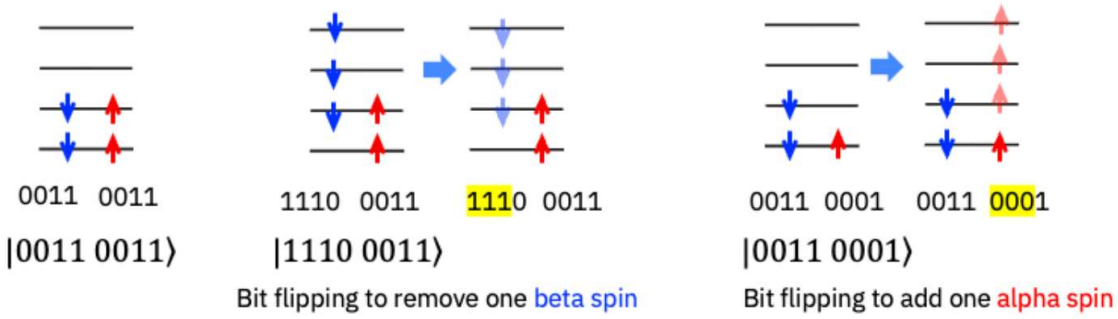


Fig 10. The strings are being corrected based on electron numbers and occupancy expectation with maxim weighted probability

At this time, when choosing which electron to be flipped, the average orbital occupancy is used and flip the bit with the maximum weighted probability of flipping. The average orbital occupancy is calculated at the end of the configuration recovery loop. So, in the first loop, we don't do this because we don't have an average orbital occupancy, and we discard the sample with the wrong particle numbers.

Configuration recovery loop 2: Subsample

The next step is to select some from the corrected samples. This time, we have 100,000 shots, so for example, we randomly select 50 samples from these 5 times, that is, 5 batches. When handling large molecules in batches, this part can be computed in parallel on a supercomputer.

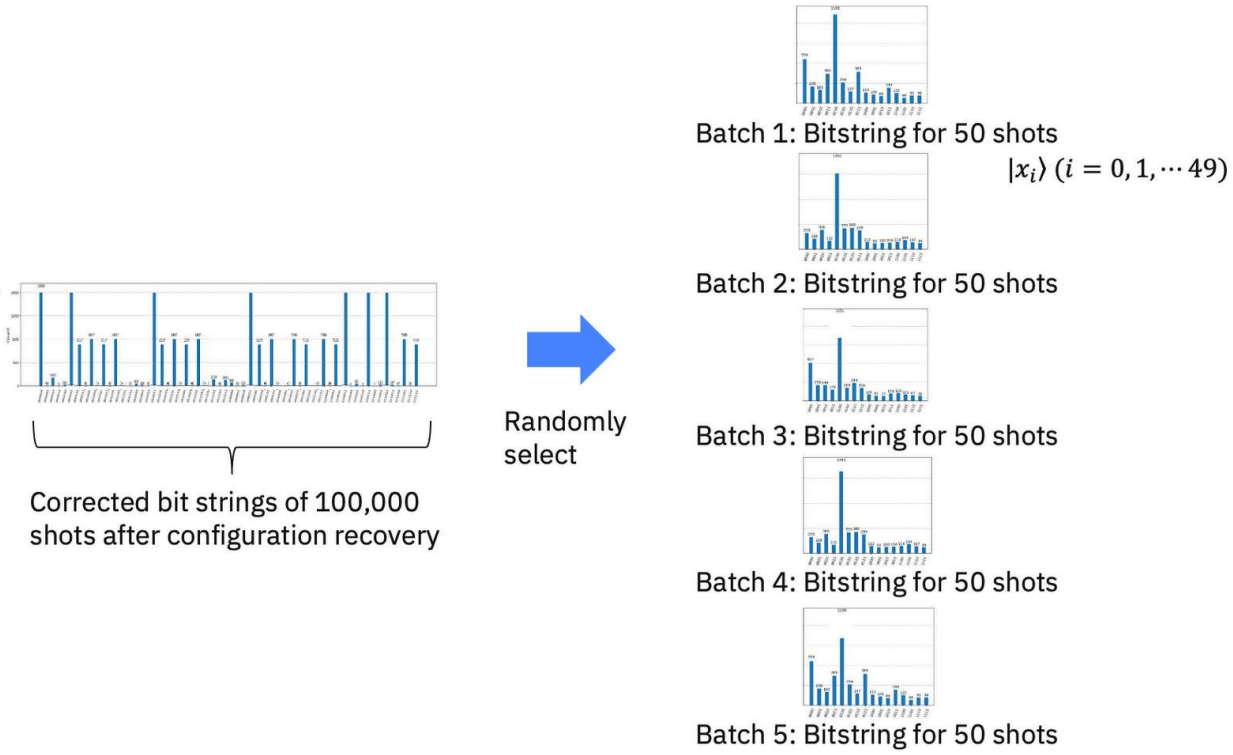


Fig 11. Subsampling from corrected samples

Configuration recovery loop 3: Diagonalize in subspace

Next is diagonalizing in subspace. In the previous example, the original Hamiltonian H, $2^{32} \times 2^{32}$ matrix, is projected into the subspace created by 50 bitstrings, $|x_i\rangle$, and diagonalized. This projection matrix P_S can have 2^{32} different measured bitstrings, from 0...0 to 1...1, but a maximum of 50 bitstrings can be measured. P_S is a matrix with only 50 diagonal components and the rest of the matrix with 0. Therefore, by multiplying P_S by the original Hamiltonian H, the projected Hamiltonian H_s has only 50×50 components and all other elements are 0. So, we can diagonalize only 50×50 matrix instead of the original huge $2^{32} \times 2^{32}$ matrix.

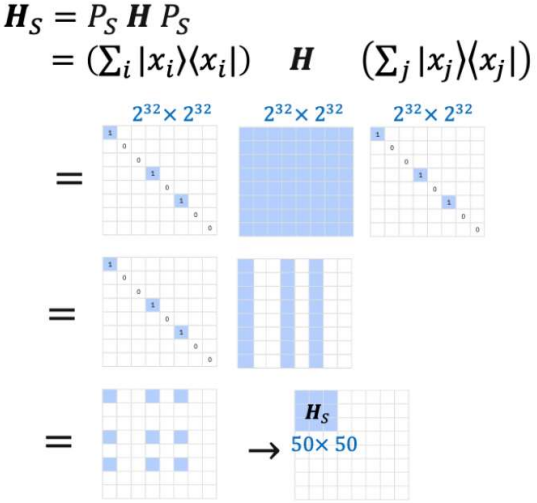


Fig 12. Matrix diagonalization in the subspace

Configuration recovery loop 4: Find the lowest energy

Finally, from the results of all batches, we obtain an estimate of the average orbital occupancy, and the lowest energy as an estimate of the ground state. And update these data.

Qiskit patterns

We implement a Qiskit patterns showing how SQD process:

1. **Step 1: Map to quantum problem**
    - Generate an ansatz for estimating the ground state
2. **Step 2: Optimize the problem**
    - Transpile the ansatz for the backend
3. **Step 3: Execute experiments**
    - Draw samples from the ansatz using the `Sampler` primitive
4. **Step 4: Post-process results**
    - Self-consistent configuration recovery loop
        - Post-process the full set of bitstring samples, using prior knowledge of particle number and the average orbital occupancy calculated on the most recent iteration.
        - Probabilistically create batches of subsamples from recovered bitstrings.
        - Project and diagonalize the molecular Hamiltonian over each sampled subspace.
        - Save the minimum ground state energy found across all batches and update the avg orbital occupancy.

Step 1: Map classical inputs to a quantum problem

We will find an approximation to the ground state of the molecule at equilibrium in the 6-31G basis set. First, we specify the molecule and its properties.

```
warnings.filterwarnings("ignore")

# Specify molecule properties
open_shell = False
spin_sq = 0

# Build N2 molecule
mol = pyscf.gto.Mole()
mol.build(
```

```
    atom=[["N", (0, 0, 0)], ["N", (1.0, 0, 0)]],
    basis="6-31g",
    symmetry="Dooh",
)

# Define active space
n_frozen = 2
active_space = range(n_frozen, mol.nao_nr())

# Get molecular integrals
scf = pyscf.scf.RHF(mol).run()
num_orbitals = len(active_space)
n_electrons = int(sum(scf.mo_occ[active_space]))
num_elec_a = (n_electrons + mol.spin) // 2
num_elec_b = (n_electrons - mol.spin) // 2
cas = pyscf.mcscf.CASCI(scf, num_orbitals, (num_elec_a, num_elec_b))
mo = cas.sort_mo(active_space, base=0)
hcore, nuclear_repulsion_energy = cas.get_h1cas(mo)
eri = pyscf.ao2mo.restore(1, cas.get_h2cas(mo), num_orbitals)

# Compute exact energy
exact_energy = cas.run().e_tot
```

```
converged SCF energy = -108.835236570775
CASCI E = -109.046671778080  E(CI) = -32.8155692383187  S^2 = 0.0000000
```

Before constructing the `LUCJ` ansatz circuit, we first perform a CCSD calculation in the following code cell. The t_1 and t_2 amplitudes from this calculation will be used to initialize the parameters of the ansatz.

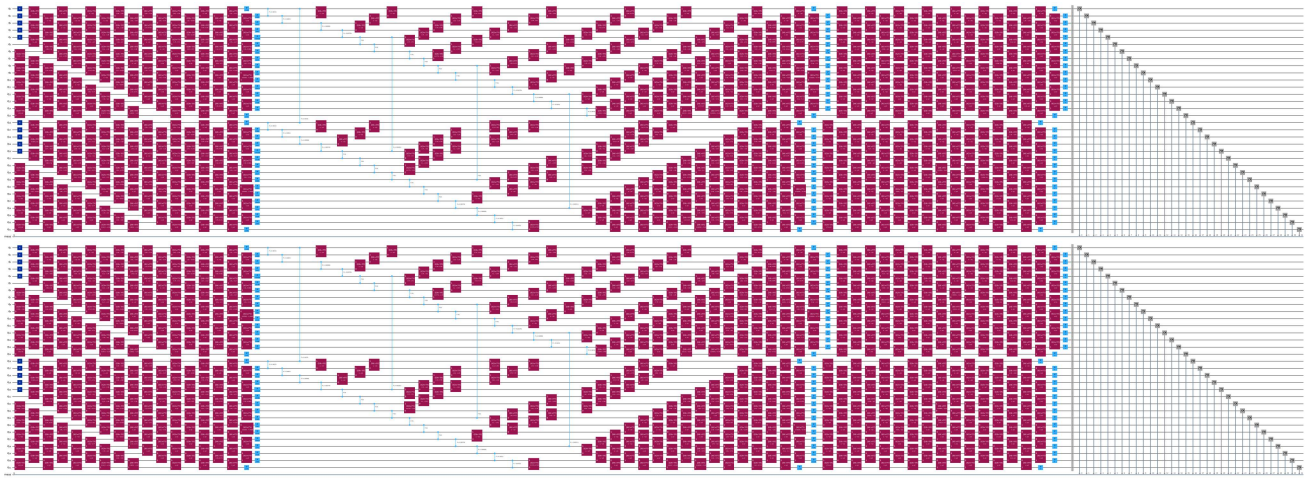
```
# Get CCSD t2 amplitudes for initializing the ansatz
ccsd = pyscf.cc.CCSD(scf, frozen=[i for i in range(mol.nao_nr()) if i not in active_space
t1 = ccsd.t1
t2 = ccsd.t2
```

```
E(CCSD) = -109.0398256929734  E_corr = -0.2045891221988305
```

Now, we use ffsim to create the ansatz circuit. Since our molecule has a closed-shell Hartree-Fock state, we use the spin-balanced variant of the UCJ ansatz, UCJOpSpinBalanced. We pass interaction pairs appropriate for a heavy-hex lattice qubit topology.

```
n_reps = 1
alpha_alpha_indices = [(p, p + 1) for p in range(num_orbitals - 1)]
alpha_beta_indices = [(p, p) for p in range(0, num_orbitals, 4)]

ucj_op = ffsim.UCJOpSpinBalanced.from_t_amplitudes(
    t2=t2,
    t1=t1,
    n_reps=n_reps,
    interaction_pairs=(alpha_alpha_indices, alpha_beta_indices),
)
```

```
nelec = (num_elec_a, num_elec_b)

# create an empty quantum circuit
qubits = QuantumRegister(2 * num_orbitals, name="q")
circuit = QuantumCircuit(qubits)

# prepare Hartree-Fock state as the reference state and append it to the quantum circuit
circuit.append(ffsim.qiskit.PrepareHartreeFockJW(num_orbitals, nelec), qubits)

# apply the UCJ operator to the reference state
circuit.append(ffsim.qiskit.UCJOpSpinBalancedJW(ucj_op), qubits)
circuit.measure_all()

circuit.decompose().decompose().draw("mpl", fold =-1)
```



Step 2: Optimize problem for quantum execution

Next, we optimize the circuit for a target hardware. We'll use the 127-qubit `ibm_brisbane` QPU.

```
service = QiskitRuntimeService(name="qgss-2025")
backend = service.backend("ibm_brisbane")
```

We recommend the following steps to optimize the ansatz and make it hardware-compatible.

- Select physical qubits (`initial_layout`) from the target hardware that adheres to the zig-zag pattern described above. Laying out qubits in this pattern leads to an efficient hardware-compatible circuit with less gates.
- Generate a staged pass manager using the generate_preset_pass_manager function from qiskit with your choice of `backend` and `initial_layout` .
- Set the `pre_init` stage of your staged pass manager to `ffsim.qiskit.PRE_INIT` . `ffsim.qiskit.PRE_INIT` includes qiskit transpiler passes that decompose gates into orbital rotations and then merges the orbital rotations, resulting in fewer gates in the final circuit.
- Run the pass manager on your circuit.

```
spin_a_layout = [0, 14, 18, 19, 20, 33, 39, 40, 41, 53, 60, 61, 62, 72, 81, 82]
spin_b_layout = [2, 3, 4, 15, 22, 23, 24, 34, 43, 44, 45, 54, 64, 65, 66, 73]
initial_layout = spin_a_layout + spin_b_layout

pass_manager = generate_preset_pass_manager(
    optimization_level=3, backend=backend, initial_layout=initial_layout
)

# We will use the circuit generated by this pass manager for hardware execution
pass_manager.pre_init = ffsim.qiskit.PRE_INIT
isa_circuit = pass_manager.run(circuit)
print(f"Gate counts (w/ pre-init passes): {isa_circuit.count_ops()}")
```

```
Gate counts (w/ pre-init passes): OrderedDict([('rz', 2466), ('sx', 2125), ('ecr', 73
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
cd /content/drive/MyDrive/QGSS-2025/lab-3
```

```
/content/drive/MyDrive/QGSS-2025/lab-3
```

Step 3: Execute using Qiskit Primitives

After optimizing the circuit for hardware execution, we are ready to run it on the target hardware and collect samples for ground state energy estimation.

⚠️ **Note:** We have commented out the code for running the circuit on a QPU and left it for the user's reference. Instead of running on real hardware in this walkthrough, we will just read in 100k samples drawn from `ibm_brisbane` at an earlier time.

```
# from qiskit_ibm_runtime import SamplerV2 as Sampler

# sampler = Sampler(mode=backend)
# job = sampler.run([isa_circuit], shots=10_000)
# primitive_result = job.result()
# pub_result = primitive_result[0]
# bit_array = pub_result.data.meas

bit_array = np.load('utils/N2_device_bitarray.npy', allow_pickle=True).item()
```

Step 4: Post-process and return result to desired classical format

Recall that the self-consistent configuration recovery is an iterative procedure that runs in a loop. In the following code cell, the first iteration of the loop simply uses the raw samples (after post-selection on symmetries) as input to the diagonalization procedure to obtain an estimate of the average orbital occupancies. Later iterations of the loop use these occupancies to generate new configurations from raw samples that violate the symmetries. These configurations are collected and then subsampled to produce batches of configurations, which are then used to project the Hamiltonian and compute a ground state estimate with an eigenstate solver.

There are a few user-controlled options which are important for this technique:

- `max_iterations` : Number of iterations of the self-consistent recovery loop.
- `num_batches` : Number of batches of configurations to subsample (this will be the number of separate calls to the eigenstate solver)
- `samples_per_batch` : Number of unique configurations to include in each batch
- `max_cycles` : Maximum number of Davidson cycles run by the eigenstate solver

Warning: 5 minutes needed

When running the code below it will take up to 5 minutes (depending on your computer) to execute and will block this notebook for this time.

```
%%time
# SQD options
energy_tol = 1e-3
occupancies_tol = 1e-3
max_iterations = 5

# Eigenstate solver options
num_batches = 5
samples_per_batch = 50
symmetrize_spin = True
carryover_threshold = 1e-4
max_cycle = 200
rng = np.random.default_rng(24)


# Pass options to the built-in eigensolver. If you just want to use the defaults,
# you can omit this step, in which case you would not specify the sci_solver argument
# in the call to diagonalize_fermionic_hamiltonian below.
sci_solver = partial(solve_sci_batch, spin_sq=0.0, max_cycle=max_cycle)


# List to capture intermediate results
result_history = []

def callback(results: list[SCIResult]):
    result_history.append(results)
    iteration = len(result_history)
    print(f"Iteration {iteration}")
```

```
        for i, result in enumerate(results):
            print(f"\tSubsample {i}")
            print(f"\t\tEnergy: {result.energy + nuclear_repulsion_energy}")
            print(f"\t\tSubspace dimension: {np.prod(result.sci_state.amplitudes.shape)}")

    result = diagonalize_fermionic_hamiltonian(
        hcore,
        eri,
        bit_array,
        samples_per_batch=samples_per_batch,
        norb=num_orbitals,
        nelec=nelec,
        num_batches=num_batches,
        energy_tol=energy_tol,
        occupancies_tol=occupancies_tol,
        max_iterations=max_iterations,
        sci_solver=sci_solver,
        symmetrize_spin=symmetrize_spin,
        carryover_threshold=carryover_threshold,
        callback=callback,
        seed=rng,
    )
```

```
                    Energy: -108.84150457670017
                    Subspace dimension: 10000
                Subsample 3
                    Energy: -107.91290084871596
                    Subspace dimension: 9409
                Subsample 4
                    Energy: -108.84031227231543
                    Subspace dimension: 9216
         Iteration 3
                Subsample 0
                    Energy: -108.89577578545783
```

```
                    Energy: -108.9546155305682
                    Subspace dimension: 20736
            Subsample 4
                    Energy: -108.90034365029865
                    Subspace dimension: 21316
    Iteration 5
            Subsample 0
                    Energy: -108.96268770420869
                    Subspace dimension: 29929
            Subsample 1
                    Energy: -108.9688180929162
                    Subspace dimension: 32041
            Subsample 2
                    Energy: -108.96611554852339
                    Subspace dimension: 32400
            Subsample 3
                    Energy: -108.96736672345207
                    Subspace dimension: 30976
            Subsample 4
                    Energy: -108.96877604604418
                    Subspace dimension: 30276
    CPU times: user 4min 17s, sys: 1.63 s, total: 4min 18s
    Wall time: 3min 43s
```

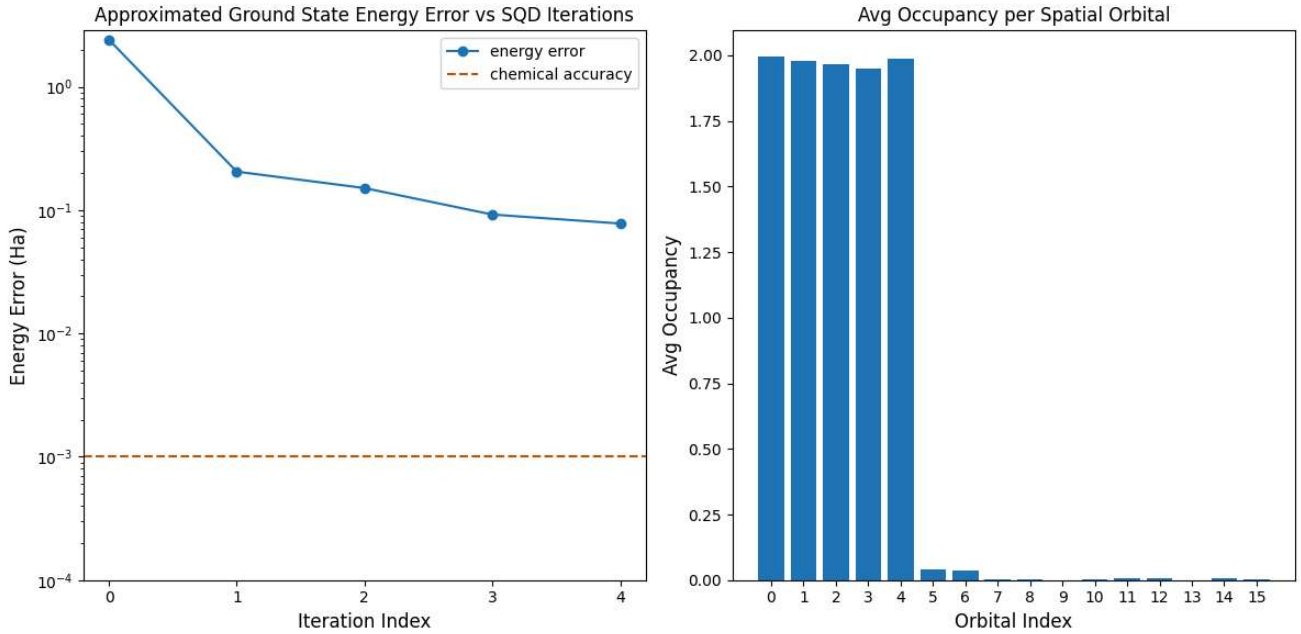
Visualize the results

To see the result, we first create a function `plot_energy_and_occupancy` .

```
def plot_energy_and_occupancy(result_history, exact_energy):

    # Data for energies plot
    x1 = range(len(result_history))
    min_e = [
        min(result, key=lambda res: res.energy).energy + nuclear_repulsion_energy
        for result in result_history
    ]
    e_diff = [abs(e - exact_energy) for e in min_e]
    yt1 = [1.0, 1e-1, 1e-2, 1e-3, 1e-4]

    # Chemical accuracy (+/- 1 milli-Hartree)
    chem_accuracy = 0.001

    # Data for avg spatial orbital occupancy
    y2 = np.sum(result.orbital_occupancies, axis=0)
    x2 = range(len(y2))

    fig, axs = plt.subplots(1, 2, figsize=(12, 6))

    # Plot energies
    axs[0].plot(x1, e_diff, label="energy error", marker="o")
    axs[0].set_xticks(x1)
    axs[0].set_xticklabels(x1)
    axs[0].set_yticks(yt1)
    axs[0].set_yticklabels(yt1)
    axs[0].set_yscale("log")
```

```
    axs[0].set_ylim(1e-4)
    axs[0].axhline(y=chem_accuracy, color="#BF5700", linestyle="--", label="chemical
    axs[0].set_title("Approximated Ground State Energy Error vs SQD Iterations")
    axs[0].set_xlabel("Iteration Index", fontdict={"fontsize": 12})
    axs[0].set_ylabel("Energy Error (Ha)", fontdict={"fontsize": 12})
    axs[0].legend()

    # Plot orbital occupancy
    axs[1].bar(x2, y2, width=0.8)
    axs[1].set_xticks(x2)
    axs[1].set_xticklabels(x2)
    axs[1].set_title("Avg Occupancy per Spatial Orbital")
    axs[1].set_xlabel("Orbital Index", fontdict={"fontsize": 12})
    axs[1].set_ylabel("Avg Occupancy", fontdict={"fontsize": 12})

    print(f"Exact energy: {exact_energy:.5f} Ha")
    print(f"SQD energy: {min_e[-1]:.5f} Ha")
    print(f"Absolute error: {e_diff[-1]:.5f} Ha")
    plt.tight_layout()
    plt.show()


plot_energy_and_occupancy(result_history, exact_energy)
```

```
Exact energy: -109.04667 Ha
SQD energy: -108.96882 Ha
Absolute error: 0.07785 Ha
```



The first plot shows that after a couple of iterations we estimate the ground state energy within `~100 mH` (chemical accuracy is typically accepted to be `1 kcal/mol ≈ 1.6 mH`). The energy can be improved by drawing more samples from the circuit or increasing the number of samples per batch.

The second plot shows the average occupancy of each spatial orbital after the final iteration. We can see that both the spin-up and spin-down electrons occupy the first five orbitals with high probability in our solutions.

Exercise 2: Flip a bit by configuration recovery

In a problem in which an N_2 molecule is prepared using the STO-3G basis set, we perform configuration recovery. When the average orbital occupancy n is as follows, we will correct the bitstring x as follows. What bitstring is most likely to be modified to?

$n = [0.007, 0.029, 0.029, 0.995, 0.976, 0.976, 0.993, 0.997, 0.007, 0.029, 0.029, 0.995, 0.997]$

$x = [1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0]$

A weighted probability of flipping $w(y)$ using a modified ReLU function is calculated as follows from [2].

$$w(y) = \begin{cases} \delta \cdot \frac{y}{h} & \text{if } y \leq h \\ \delta + (1-\delta) \cdot \frac{y-h}{1-h} & \text{if } y > h \end{cases}$$

Here, y is a probability of flipping, and defined as $y[i] = |x[i] - n[i]|$ for the i -th spin orbital. h defines the location of the "corner" of the ReLU function, and the parameter δ defines the value of the ReLU function at the corner. We use $\delta = 0.01$, same as [2], and h =number of alpha(or beta) particles/number of alpha(or beta) spin orbitals$= N/M$ (filling factor).

In the actual configuration recovery, a bit is randomly inverted with a weight of $w(y)$. In this exercise, answer the result of inverting the bit i with the largest $w(y[i])$ as the bitstring with the highest probability of obtaining.

Note:

- The right half of a bitstring represents spin-up orbitals, and the left half represents spin-down orbitals. A `1` means the orbital is occupied by an electron, and a `0` means the orbital is empty.
- Please refer to the section "4.1 Configuration recovery overview" in Lesson 4: SQD application of "Quantum Diagonalization Algorithm" of IBM Quantum Learning.
- In this case, one more beta particle is needed, so if the i-th orbital is already occupied and need not to be flipped, you set its y_beta[i] to 0.

```
n = [0.007, 0.029, 0.029, 0.995,
     0.976, 0.976, 0.993, 0.997,
     0.007, 0.029, 0.029, 0.995,
     0.976, 0.976, 0.993, 0.997]

x = [1, 0, 0, 0,  1, 0, 1, 1,  1, 0, 1, 1,  1, 1, 0, 0]


x = np.array(x)
n = np.array(n)

# ---- TODO : Task 2 ---
```

```
# Divide into alpha spin and beta spin
x_alpha = x[8:]
x_beta = x[:8]

# probability of flipping
y = 1 - 2 * n
y_alpha = y[8:]
y_beta = y[:8]

# In this case, one more beta particle is needed, so set y_beta[i] to 0 if x_beta[i] is a
for i in range(len(y_beta)):
    if x_beta[i] == 1:
        y_beta[i] = 0

# --- End of TODO ---

print(y_beta)
```

```
[ 0.     0.942  0.942 -0.99   0.    -0.952  0.     0.   ]
```

```
h = 5/8
delta = 0.01
w = np.zeros(len(y_beta))

# find the maximum w
# ---- TODO : Task 2 ---
max_w = -1.0
max_index = -1
for i in range(len(y_beta)):
    w[i] = y_beta[i]

max_index = np.argmax(w)

# --- End of TODO ---
print(max_index, max_w)
```

```
1 -1.0
```

```
# Flip the bit of the index with the largest w
# ---- TODO : Task 2 ---
corrected_x_beta = x_beta.copy()
for i in range(len(y_beta)):
    # Check if the current index is the one we want to flip
    if i == max_index:
        # Flip the bit (0 becomes 1, 1 becomes 0)
        corrected_x_beta[i] = 1 - corrected_x_beta[i]

x = np.concatenate([corrected_x_beta, x_alpha])
corrected_x = x.tolist()
# --- End of TODO ---
# print(corrected_x)
```

```
corrected_x_beta
```

```
array([1, 0, 0, 1, 1, 0, 1, 1])
```

```
# Submit your answer using following code

grade_lab3_ex2(corrected_x) # Expected result type: list
```

```
Submitting your answer. Please wait...
Congratulations 🎉! Your answer is correct and has been submitted.
```

5. Improve the ansatz

The more accurate the sampling from the quantum circuit, the better the results. Therefore, how to make the ansatz for sampling is one of the key points in SQD performance. Depending on how the ansatz is set, different bit strings are sampled, which affects the accuracy of the energy estimate value that can be obtained. Here, we will explore the different ansatz to improve the result.

5.1 Change basis set

First, let's try to model the molecule in a more natural way by changing the basis set of the molecule to account for more electron orbitals. Incorporating more orbitals into the calculation will increase the computational complexity, but should lower the value of the energy estimation.

Exercise 3: Change basis set

Change the basis set from `6-31G` to `cc-pvdz`. How many qubits are needed if we use the LUCJ ansatz with **6** ancillary qubits in case we use `ibm_torino` as a backend?
Note: Because of the geometric limitation of `ibm_torino`, the number of ancillary qubits is limited to 6 here.

```
warnings.filterwarnings("ignore")

# Specify molecule properties
open_shell = False
spin_sq = 0

# Build N2 molecule
mol = pyscf.gto.Mole()
mol.build(
    atom=[["N", (0, 0, 0)], ["N", (1.0, 0, 0)]],
    # ---- TODO : Task 3 ---
    basis= "cc-pvdz", #"6-31G" ### input your code here ###,
    # --- End of TODO ---
    symmetry="Dooh",
```

```
)

# Define active space
n_frozen = 2
active_space = range(n_frozen, mol.nao_nr())

# Get molecular integrals
scf = pyscf.scf.RHF(mol).run()
num_orbitals = len(active_space)
n_electrons = int(sum(scf.mo_occ[active_space]))
num_elec_a = (n_electrons + mol.spin) // 2
num_elec_b = (n_electrons - mol.spin) // 2
cas = pyscf.mcscf.CASCI(scf, num_orbitals, (num_elec_a, num_elec_b))
mo = cas.sort_mo(active_space, base=0)
hcore, nuclear_repulsion_energy = cas.get_h1cas(mo)
eri = pyscf.ao2mo.restore(1, cas.get_h2cas(mo), num_orbitals)

print(num_orbitals)
```

```
converged SCF energy = -108.929838385609
26
```

```
# ---- TODO : Task 3 ---
n_qubits = 2 * num_orbitals + 6  # 2 qubits per orbital + 6 ancillary qubits ### input yc
# --- End of TODO ---



n_qubits
```

```
58
```

```
# Submit your answer using following code

grade_lab3_ex3(n_qubits) # Expected result type: integer
```

```
Submitting your answer. Please wait...
Congratulations 🎉! Your answer is correct and has been submitted.
```

.2 Select the best layout

Now that we have obtained the number of qubits, the next step is to determine the layout of the physical qubits. To use a larger basis, we will use a Heron device with better performance.

Exercise 4: Select the best layout

Which qubits should we choose as the initial placement to get the best results? To select the qubits, you need to check the errors of each qubit. In the following map, qubits with a readout error greater than 0.1 are shown in black, and edges with a CZ error greater than 0.1 are shown in white. Answer the best `initial_layout`, which is used as an argument of pass_manger to create the ISA circuit.

We will use `ibm_torino` as a backend. Because of the geometric limitation of `ibm_torino`, the number of ancillary qubits limit to 6 here.

Select the best initial qubit layout to get better sampling by following:

1. List the qubits with a readout error of 0.1 or more as `bad_readout_qubits` from `backend_target`.
2. List of edges with a CZ error of 0.1 or more as `bad_czgate_edges` from `backend_target`.
3. Display the coupling map with `bad_readout_qubits` in black and `bad_czgate_edges` in white.
4. Select the best initial qubit layout.

⚠️ **Note: You need to use the preloaded pickle file `backend_target_v20.pkl` or `backend_target_v21.pkl` as `backend_target` for backend information in order to pass the grader of this exercise**. In Lab 2, we used `backend.properties()`, `backend.target`, etc., but we won't use them this time. We have commented out the code for a real backend a for the user's reference.

```
# from qiskit_ibm_runtime import QiskitRuntimeService
# service = QiskitRuntimeService(name="qgss-2025")
# backend = service.backend('ibm_torino')
# backend_target = backend.target
```

```
backend = service.backend('ibm_torino')
```

```
import qiskit
print(f'Qiskit: {qiskit.__version__}')
```

```
Qiskit: 2.1.1
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
cd /content/drive/MyDrive/QGSS-2025/lab-3
```

```
/content/drive/MyDrive/QGSS-2025/lab-3
```

⚠️ **Note:** If you are using Qiskit version 2.1.x, open `backend_target_v21.pkl` in the next cell.

```
# for Qiskit version 2.0.x users
# with open("utils/backend_target_v20.pkl", "rb") as f:
# for Qiskit version 2.1.x users
```

```
with open("utils/backend_target_v21.pkl", "rb") as f:
    backend_target = pickle.load(f)
```

```
backend_target
```

```
<qiskit.transpiler.target.Target at 0x1d57e780>
```

```
backend_target.qubit_properties
```

```
frequency=None),
 QubitProperties(t1=0.0001610333523823885, t2=2.9086374094505125e-05,
frequency=None),
 QubitProperties(t1=0.0001081238543244775 3, t2=5.9459982774397853e-05,
frequency=None),
 QubitProperties(t1=0.0002038076931334559, t2=0.0001354668248433232 8,
frequency=None),
 QubitProperties(t1=0.0001518735457212574 8, t2=9.439975896148208e-05,
frequency=None),
```

Note: Please refer "Instruction properties" part of "Get QPU information with Qiskit" to get the properties like "measure" and "cz" from `backend_target`.

```
BAD_READOUT_ERROR_THRESHOLD = 0.1
BAD_CZGATE_ERROR_THRESHOLD = 0.1
backend_num_qubits = 133

# ---- TODO : Task 4 ---
# List qubits with readout error above the threshold
bad_readout_qubits = []
for qubit in range(backend_num_qubits):
    try:
        props = backend_target['measure'][(qubit,)]
        if hasattr(props, 'error') and props.error >= BAD_READOUT_ERROR_THRESHOLD:
            bad_readout_qubits.append(qubit)
    except Exception:
        continue ### build your code here ###

# List edges with a CZ gate error above the threshold
bad_czgate_edges = []
cmap = backend_target.build_coupling_map().get_edges()
for edge in cmap:
    try:
        props = backend_target['cz'][(edge[0], edge[1])]
        if hasattr(props, 'error') and props.error >= BAD_CZGATE_ERROR_THRESHOLD:
            bad_czgate_edges.append(edge)
    except KeyError:
        continue

'''# Iterate through all coupled qubit pairs (edges) on the device
for q1, q2 in backend_target.build_coupling_map().get_edges():
    # Check the error for the 'cz' gate on that specific edge
    if backend_target.gate_error("cz", [q1, q2]) > BAD_CZGATE_ERROR_THRESHOLD:
        bad_czgate_edges.append((q1, q2))''' ### build your code here ###
# --- End of TODO ---
print("Bad readout qubits:", bad_readout_qubits)
print("Bad CZ gates:", bad_czgate_edges)
```

```
Bad readout qubits: [12, 53, 115, 126, 131]
Bad CZ gates: [(100, 101), (101, 100)]
```

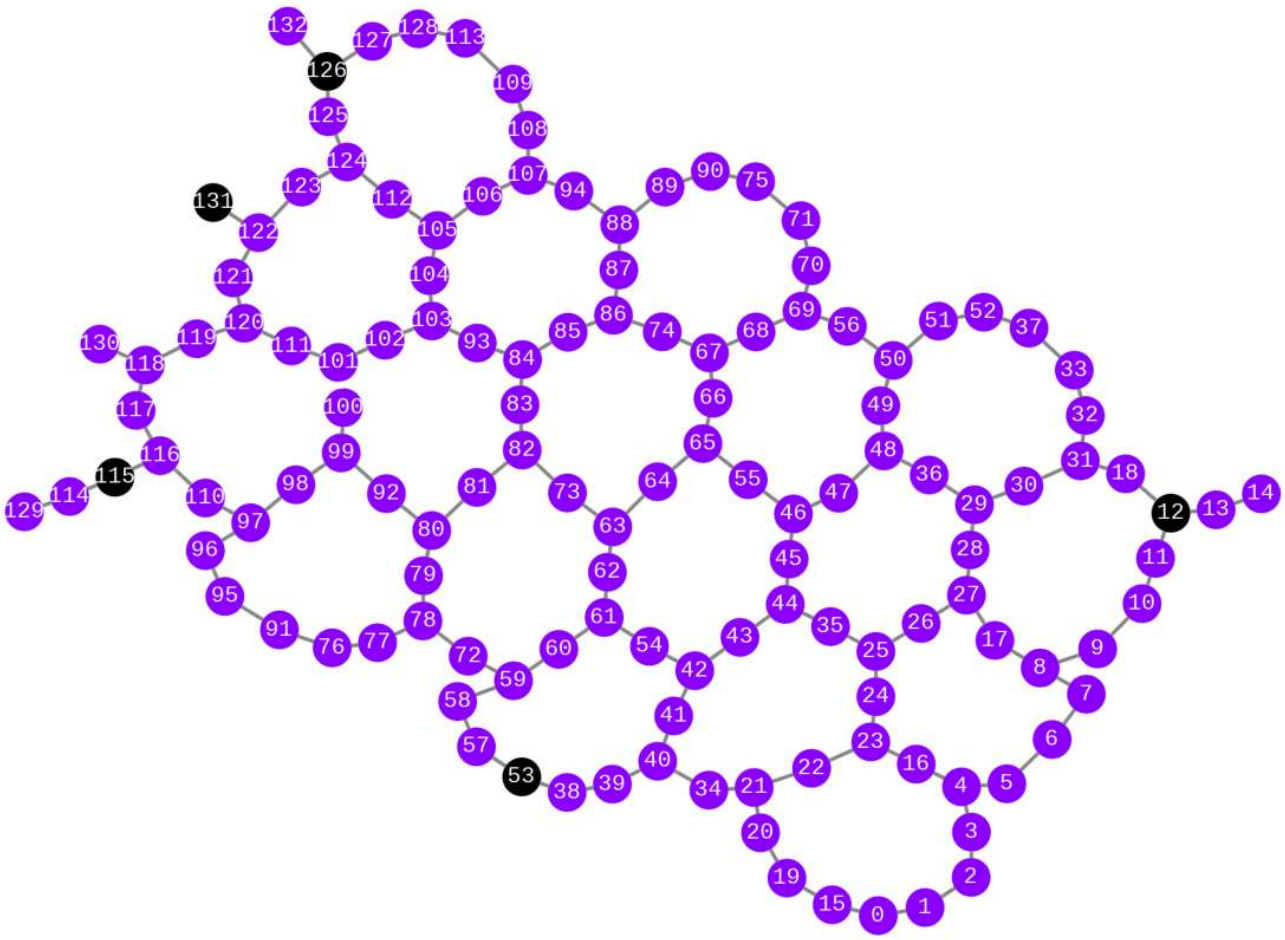
Warning: Graphviz Library needed

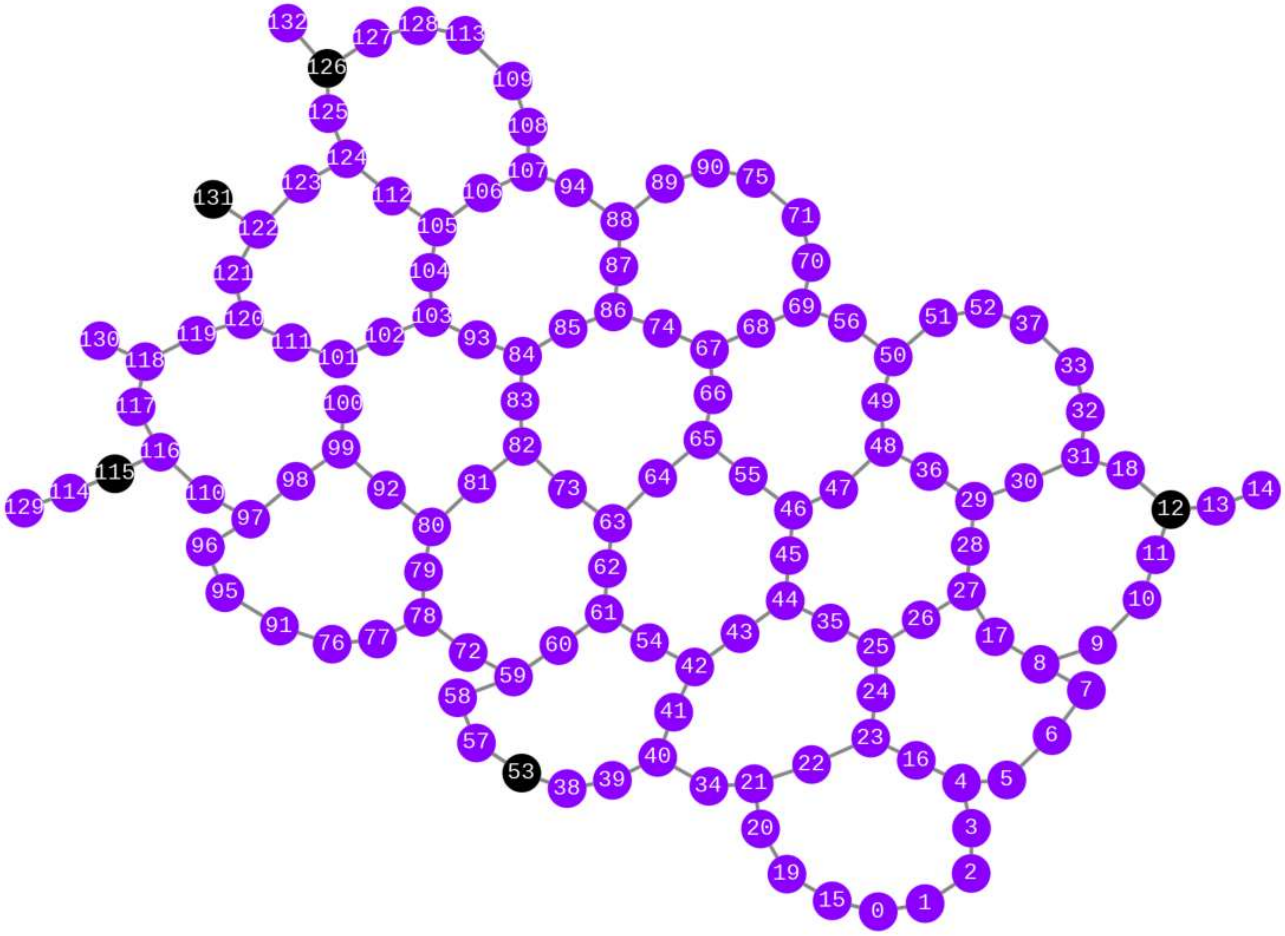
The 'Graphviz' library is required to use 'plot_coupling_map'. To install, follow the instructions at:

https://graphviz.org/download

If you dont want to install it, you can just skip the next block of code, it is only needed for visualization.

```
qubit_color = []
for i in range(backend_num_qubits):
    if i in bad_readout_qubits:
        qubit_color.append("#000000") #black
    else:
        qubit_color.append("#8c00ff") #purple
line_color = []
for e in backend_target.build_coupling_map().get_edges():
    if e in bad_czgate_edges:
        line_color.append("#ffffff") #white
    else:
        line_color.append("#888888") #gray
plot_gate_map(backend, qubit_color=qubit_color, line_color=line_color, qubit_size=50, fon
```






```
# select the initial layout
# ---- TODO : Task 4 ---
spin_a_layout = ### add your qubits list ###
```

```
spin_a_layout = ### add your qubits list ###
spin_b_layout = ### add your qubits list ###
# --- End of TODO ---
initial_layout = spin_a_layout + spin_b_layout
```

```
bad_undirected
```

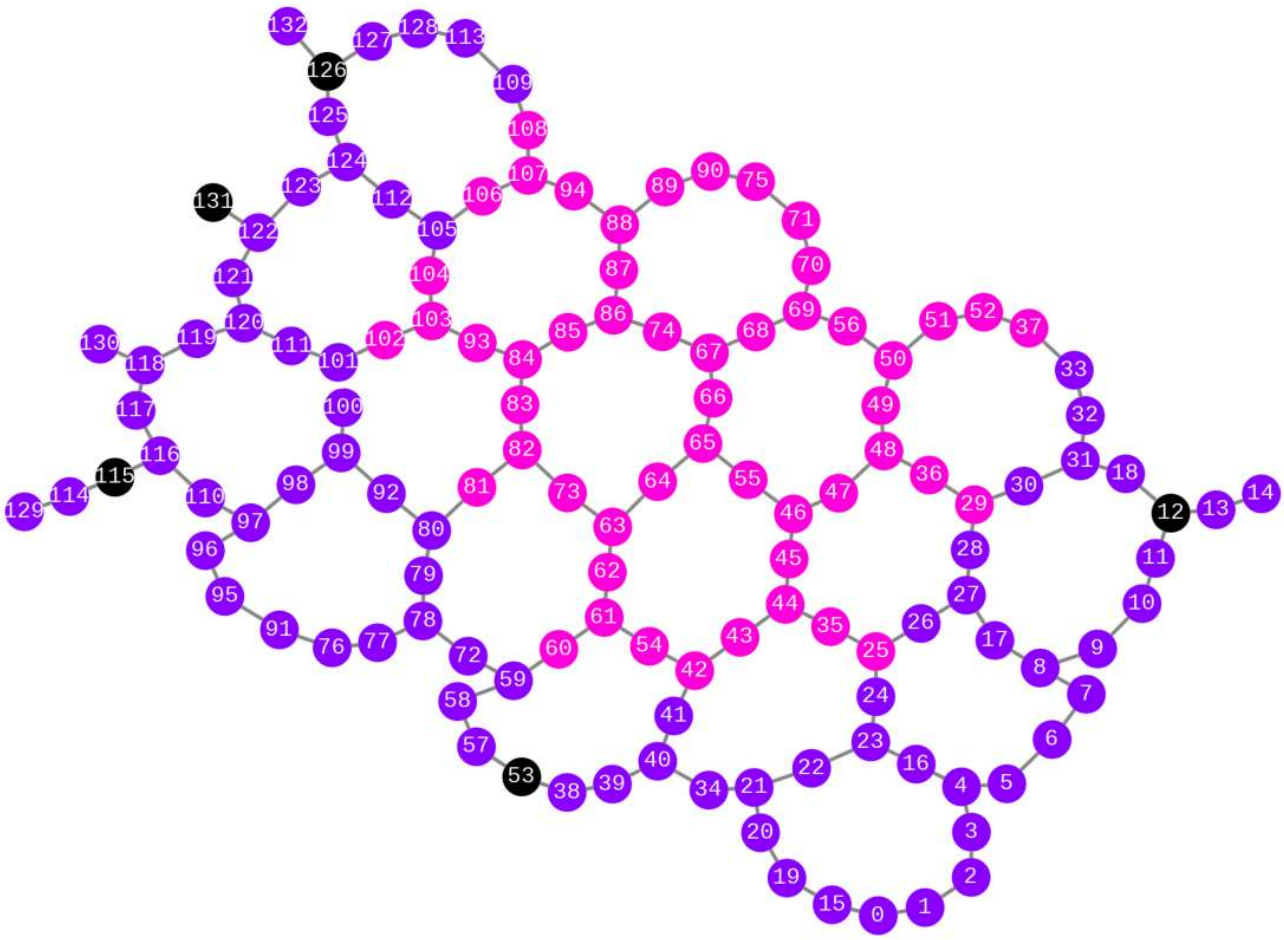
```
{(100, 101)}
```

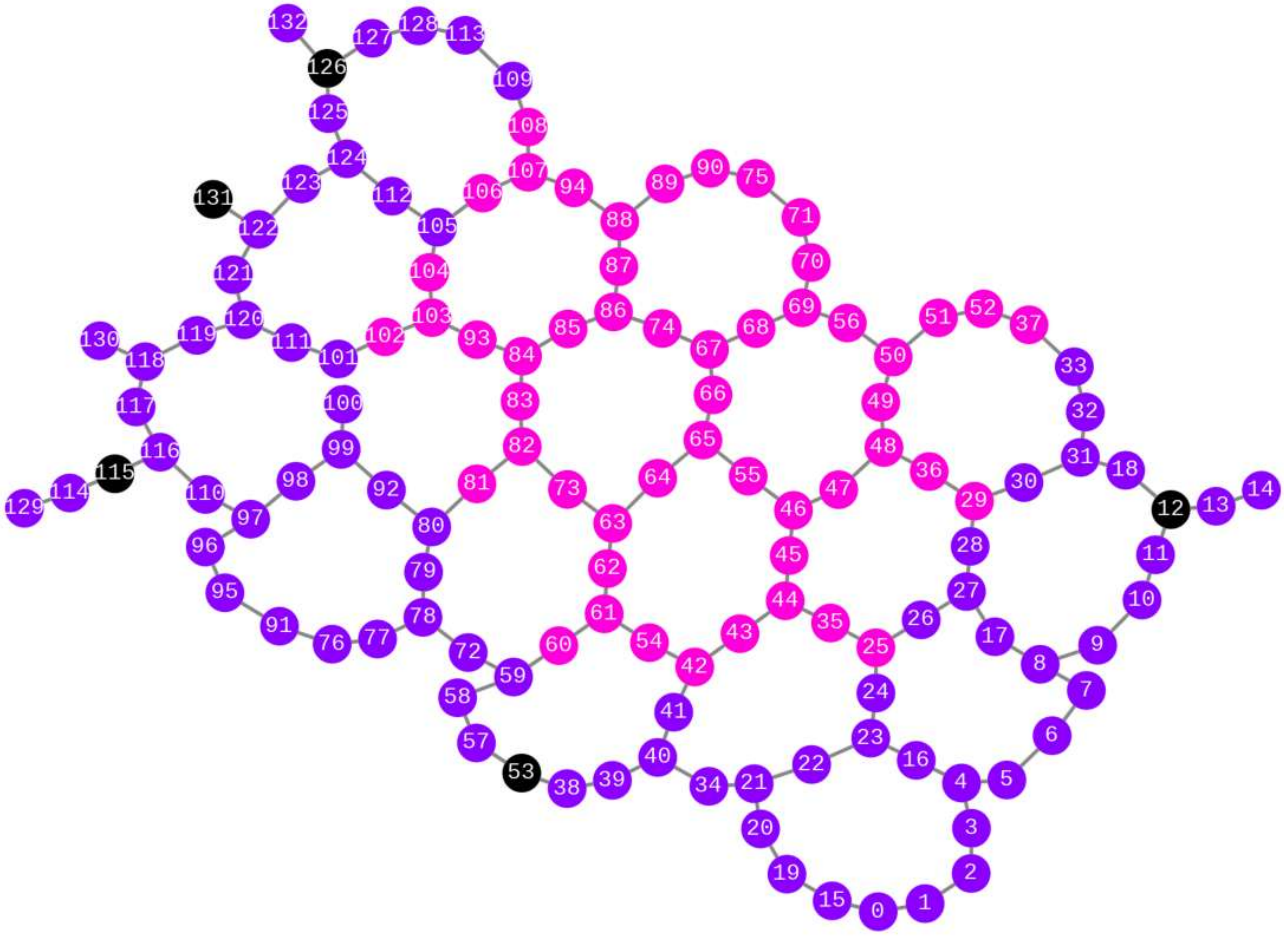
```
bad_readout_qubits
```

```
[12, 53, 115, 126, 131]
```

Check your layout here:

```
qubit_color = []
for i in range(backend_num_qubits):
    if i in bad_readout_qubits:
        qubit_color.append("#000000") #black
    elif i in initial_layout:
        qubit_color.append("#ff00dd") #pink
    else:
        qubit_color.append("#8c00ff") #purple
line_color = []
for e in backend_target.build_coupling_map().get_edges():
    if e in bad_czgate_edges:
        line_color.append("#ffffff") #white
    else:
        line_color.append("#888888") #gray
plot_gate_map(backend, qubit_color=qubit_color, line_color=line_color, qubit_size=50
```





Start coding or generate with AI.

```
Bad readout qubits: [12, 53, 115, 126, 131]
Bad CZ gates: [(100, 101), (101, 100)]
Submitting your answer. Please wait...
Oops 😟! Sorry, your answer is incorrect. Please try again.
Please review your answer and try again.
```

```
len(g)
```

```
128
```

```
for x in backend_target:
  print(x)
```

```
sx
rz
id
measure
cz
x
delay
reset
```

```
Start coding or generate with AI.
```

```
# Submit your answer using following code

grade_lab3_ex4(initial_layout) # Expected result type: lists
```

```
Submitting your answer. Please wait...
Oops 😟! Sorry, your answer is incorrect. Please try again.
Please review your answer and try again.
```

5.3 Add more interaction to LUCJ ansatz

The LUCJ ansatz used in section 4 has the form

$$|\Psi\rangle = \prod_{\mu=1}^{L} e^{\hat{K}_\mu} e^{i\hat{J}_\mu} e^{-\hat{K}_\mu} |\Phi_0\rangle$$

where $|\Phi_0\rangle$ is a reference state, often taken to be the Hartree-Fock state, and the $\hat{K}_\mu$ and $\hat{J}_\mu$ have the form

$$\hat{K}_\mu = \sum_{pq,\sigma} K^\mu_{pq}\, \hat{a}^\dagger_{p\sigma} \hat{a}_{q\sigma} \ , \ \hat{J}_\mu = \sum_{pq,\sigma\tau} J^\mu_{pq,\sigma\tau}\, \hat{n}_{p\sigma} \hat{n}_{q\tau} \ ,$$

where we have defined the number operator

$$\hat{n}_{p\sigma} = \hat{a}^\dagger_{p\sigma} \hat{a}_{p\sigma}.$$

The IBM hardware has a heavy-hex lattice qubit topology, and it yields the following index constraints on the $\mathbf{J}$ matrices:

$$\mathbf{J}^{\alpha\alpha} : \{(p, p+1)\ ,\ p = 0, \ldots, N-2\}$$
$$\mathbf{J}^{\alpha\beta} : \{(p, p)\ ,\ p = 0, 4, 8, \ldots,\ p \leq N-1\}$$

Here, $\mathbf{J}^{\alpha\alpha} = J^1_{pq,\alpha\alpha}$ and $\mathbf{J}^{\alpha\beta} = J^1_{pq,\alpha\beta}$.

In this case, $\mathbf{J}^{\alpha\alpha}$ only interacts with adjacent spins on the α or β orbit. This time, to model close to nature, let's change $\mathbf{J}^{\alpha\alpha}$ so that the interaction also works with the next adjacent spin.

$$\mathbf{J}^{\alpha\alpha} : \{(p, p+1, p+2)\ ,\ p = 0, \ldots, N-3\}$$
$$\mathbf{J}^{\alpha\beta} : \{(p, p)\ ,\ p = 0, 4, 8, \ldots,\ p \leq N-1\}$$

Exercise 5: Add more interaction to LUCJ ansatz

In the LUCJ ansatz, modify the code of `alpha_alpha_indices` so that the $\mathbf{J}$ matrix interacts not only with adjacent spins on the α or β orbit, but also with spins two steps ahead. Submit the list of `alpha_alpha_indices`.

```
# Get CCSD t2 amplitudes for initializing the ansatz
ccsd = pyscf.cc.CCSD(scf, frozen=[i for i in range(mol.nao_nr()) if i not in active_space
t1 = ccsd.t1
t2 = ccsd.t2

n_reps = 1
# ---- TODO : Task 5 ---
alpha_alpha_indices = ### input your code here ###
# --- End of TODO ---
alpha_beta_indices = [(p, p) for p in range(0, num_orbitals, 4)]

ucj_op = ffsim.UCJOpSpinBalanced.from_t_amplitudes(
    t2=t2,
    t1=t1,
    n_reps=n_reps,
    interaction_pairs=(alpha_alpha_indices, alpha_beta_indices),
)

nelec = (num_elec_a, num_elec_b)

# create an empty quantum circuit
qubits = QuantumRegister(2 * num_orbitals, name="q")
circuit = QuantumCircuit(qubits)

# prepare Hartree-Fock state as the reference state and append it to the quantum circuit
circuit.append(ffsim.qiskit.PrepareHartreeFockJW(num_orbitals, nelec), qubits)

# apply the UCJ operator to the reference state
circuit.append(ffsim.qiskit.UCJOpSpinBalancedJW(ucj_op), qubits)
circuit.measure_all()
circuit.decompose().decompose().draw("mpl", fold=-1)


# Submit your answer using following code
```

Congratulations! You finished this lab. The next session is a bonus session and does not count towards your score.

```
# Check your submission status with the code below
from qc_grader.grader.grade import check_lab_completion_status

check_lab_completion_status("qgss_2025")
```

```
Lab 0: 2/2 exercises completed (100%)
    ✅ 2509 participants have completed this lab
Lab 1: 9/9 exercises completed (100%)
    ✅ 2114 participants have completed this lab
Lab 2: 7/7 exercises completed (100%)
    ✅ 1411 participants have completed this lab
Lab 3: 3/5 exercises completed (60%)
    ✅ 1256 participants have completed this lab
Lab 4: 0/6 exercises completed (0%)
    ✅ 1222 participants have completed this lab
Functions Labs: 0/8 exercises completed (0%)
    ✅ 6 participants have completed this lab
```

Bonus: Real hardware execution with error mitigation

Finally, let's run the modified ansatz circuit on a real device and run the configuration recovery loop to find the energy. In doing so, we will use error mitigation to reduce the effect of errors as much as possible.

```
service = QiskitRuntimeService(name="qgss-2025")
backend = service.backend('ibm_torino')
```

```
pass_manager = generate_preset_pass_manager(
    optimization_level=3, backend=backend, initial_layout=initial_layout
)

# We will use the circuit generated by this pass manager for hardware execution
pass_manager.pre_init = ffsim.qiskit.PRE_INIT
isa_circuit = pass_manager.run(circuit)
print(f"Gate counts (w/ pre-init passes): {isa_circuit.count_ops()}")
```