Bayesian Iterative Quantum Amplitude Estimation Simulation Code Documentation

Qilin Li[⋆1], Atharva Vidwans[⋆2], Yazhen Wang[1], and Micheline B. Soley[†2,3,4]

[1]Department of Statistics, University of Wisconsin-Madison, 1300 University Ave Madison, WI 53706 USA
[2]Department of Chemistry, University of Wisconsin-Madison, 1101 University Ave., Madison, WI 53706, USA
[3]Department of Physics, University of Wisconsin-Madison, 1150 University Ave., Madison, WI 53706, USA
[4]Data Science Institute, University of Wisconsin-Madison, 447 Lorch Ct., Madison, WI 53706, USA

Contents

1 Introduction

We introduce Bayesian Iterative Quantum Amplitude Estimation (BIQAE), a method to reduce the cost associated with quantum amplitude estimation by harnessing the information passage of Bayesian inference. This document provides documentation for the `simulations` folder in the BIQAE GitHub repository, which includes (i) scaling analysis of quantum sample complexities and (ii) application of BIQAE to estimate molecular ground-state energies. The repository also contains two additional folders:

- `src`, which includes the source code implementation of BIQAE, and
- `demos`, which provides a usage example of how to run BIQAE in practice.

To cite this code, please include the following reference:

> Qilin Li, Atharva Vidwans, Yazhen Wang, Micheline B. Soley, "Harnessing Bayesian Statistics to Accelerate Iterative Quantum Amplitude Estimation," in preparation.

2 Dependencies

Implementation of the simulations requires Python with libraries listed in `BIQAE/requirements.txt`.

In addition to a local machine, high-throughput computing (HTC) resources and HTCondor are required for acceleration of quantum amplitude estimation (QAE) for both simulations. See https://osg-htc.org/ for HTC resources free of charge as of the compilation of this documentation. Note data files are provided such that visualization can be performed directly on a local machine without access to HTC resources.

Optional PowerShell commands are also included.

3 Instructions

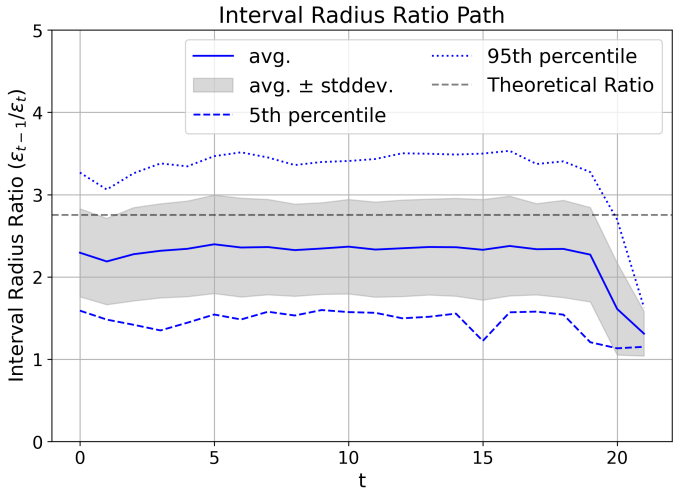
3.1 Quantum Sample Complexity Scaling Analysis

Quantum sample complexity scaling analysis of Beta-BIQAE and IQAE-CP is performed according to the code provided in the folder `BIQAE/simulations/numerical/`, as follows:

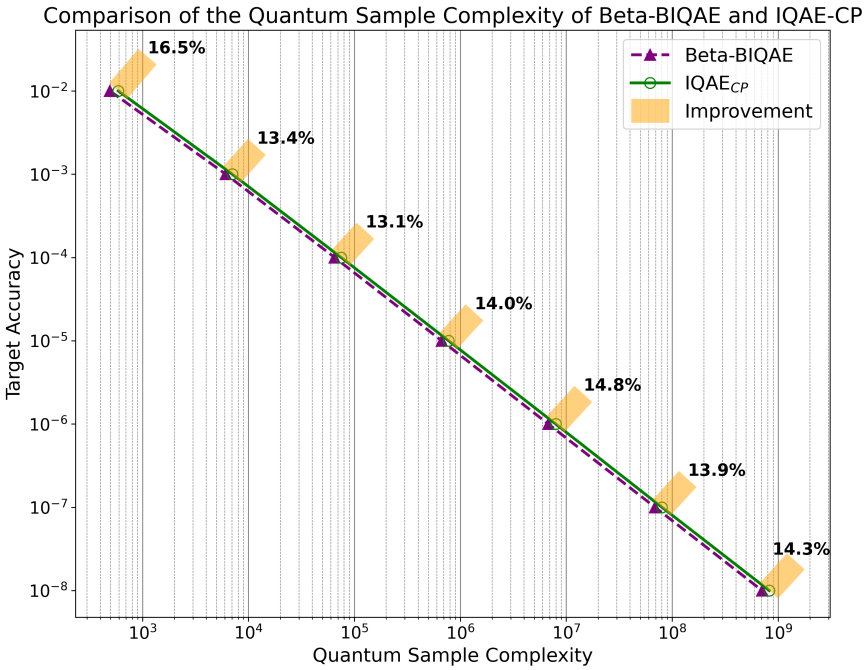
- The θ values of interest (where the amplitude $a = \sin^2\theta$) are stored in `.../numerical/all_angle_results/param_list.txt`.
- Copy the entire GitHub repository to the HTC access node, except for the folder `BIQAE/simulations/molecule/`.
- Submit `simul.sub` to HTC by running

```
condor_submit simul.sub
```

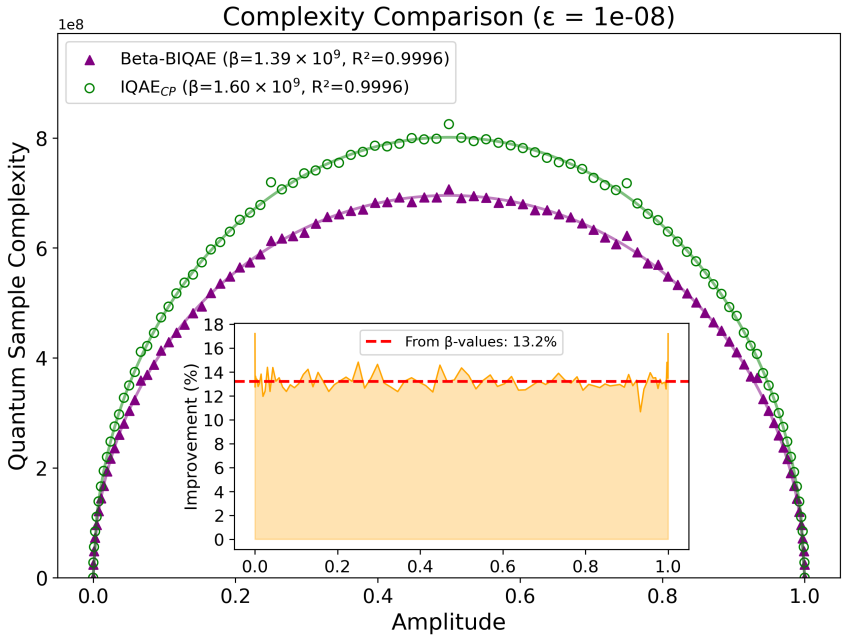
  from the `all_angle_results` directory. When the jobs are done, the `all_angle_results` folder will be populated with data files named in the format `results_THETA.pkl`, where `THETA` indicates that the results correspond to the angle $\theta =$ `THETA`°.
- Copy the folder `all_angle_results` back to the local machine.
- On the local machine, execute *in situ* `radius_ratios_plot.ipynb`, `loglog_plot.ipynb`, and `all_angle_plot.ipynb` from their eponymous folders to create the following images (see additional details in accompanying manuscript):
  - `figs_radius_ratio/radius_ratio_THETA_LEVEL.png` — a plot showing the progression of the radius ratio across stages for angle $\theta =$ `THETA`°, where results are aggregated from 1000 iterations for each amplitude value at target accuracy level $\varepsilon = 10^{-\texttt{LEVEL}}$.



– `loglog_plot_45_on_line.png` — a plot that illustrates the percentage improvement in the quantum sample complexity of Beta-BIQAE relative to IQAE-CP for $a = 0.5$ for accuracy levels ranging from 10^{-2} to 10^{-8}.



– `all_angle_plot_LEVEL.png` — a plot showing the quantum sample complexity of Beta-BIQAE and IQAE-CP against $a = \sin^2\theta,\ \theta = 0°, 5°, \ldots, 90°$ at a given target accuracy level $\varepsilon = 10^{-\texttt{LEVEL}}$.



3.2 Molecular Ground-State Energy Estimation

The folder `BIQAE/simulations/molecular/` contains the code and results for estimating the ground-state energy of molecules (H_2, LiH, HF, and BeH_2). The folder is built through the following steps:

- On the local machine:
  - In the directory `BIQAE/simulations/molecular/`, create a folder named `MOLECULENAME`, where `MOLECULENAME` is a placeholder for one of the following molecule names:`H2`, `LiH`, `HF`, or `BeH2` (see Github repository example(s)). This folder will serve as the main working directory for the selected molecule.
  - Create a file `.../MOLECULENAME/select_BL/selected.csv` that lists the bond lengths of interest for ground-state energy estimation. In `selected.csv`, the first column `BL` contains the bond lengths values in angstroms (Å), and the second column indicates whether the estimation has been completed. Initially, set all indicator values to `0` to represent that the jobs have not yet been executed.
  - Copy `vqe.ipynb` and `vqe_pm.ipynb` from the corresponding folder in the GitHub repository into the local directory `.../MOLECULENAME/`. In `vqe.ipynb`, ensure that `molecule.symbol`, `molecule.coords`, `molecule.charge` and `molecule.multiplicity` are set to the molecule of interest, respectively.
  - Execute `vqe_pm.ipynb`, which first creates the folders named `Processing_BL_BONDLENGTH/` for each bond length specified in `selected.csv`, and then calls `vqe.ipynb` to generate the following files in each each processing folder:
    - `vqe.ipynb` — an executed copy of the notebook `.../MOLECULENAME/vqe.ipynb`.
    - `sorted_pauli_list.pkl` — the Pauli decomposition of the Hamiltonian in descending order of coefficient magnitudes in the form of a list `[[Pauli string, coefficients],...]`.
    - `vqe_results.pkl` — an instance of CustomVQE, a class defined to store the results from the variational quantum eigensolver, including the ground-state energy and the ground-state wavefunction stored in terms of AAnsatz parameters.
  - Copy `all_str_A_Q.ipynb`, `all_str_A_Q_pm.ipynb` and `circuit_constructors.py` from the corresponding folder in the GitHub repository into the local directory `.../MOLECULENAME/BondLengths/`.
  - Execute `all_str_A_Q_pm.ipynb` to run `all_str_A_Q.ipynb` for each processing folder to create and save the circuits $\mathcal{A}$ and $\mathcal{Q}$ of each Pauli string in `.../Processing_BL_BONDLENGTH/all_str_A_Q/circuits/`.
  - **Important:**
    - In our current code, H_2 uses Jordon Weigner mapper whereas the LiH, HF and BeH_2 use Tapper mapping.
    - Using the taper mapping requires reversing the order of Pauli matrices within each individual Pauli code word when performing the Hadamard test. See the difference between the for loop on line 48 of the notebook `circuit_constructors.py` for LiH and the for loop on line 32 of the notebook `circuit_constructors.py` for H_2.

    - If you are not using a Jupyter kernel named `Qiskit`, you must update the `kernel_name` parameter in the notebook `all_str_A_Q_pm.ipynb` to match the name of your current kernel.
  - Create a subfolder named `run_algo` in an arbitrary processing folder and copy the files `run_algo.py`, `submit_run_algo.sh`, and `submit_run_algo.sub` from the corresponding `run_algo` folder in the GitHub repository. The target accuracy can be changed by editing lines 36 and 89 in `run_algo.py`.
  - Copy the created `run_algo` folder to all other processing folders. The following PowerShell script is provided for convenience (note the string `processing_BL_0p60000` must be replaced with the name of the processing folder initially used, and the script must be executed from inside the `BondLengths` directory):

```
Get-ChildItem -Directory -Filter "processing_BL_*" |
Where-Object { $_.Name -ne "processing_BL_0p60000" } |
ForEach-Object {
    Remove-Item "$($_.FullName)\run_algo" -Recurse -Force -
        ErrorAction SilentlyContinue;
    Copy-Item -Path "processing_BL_0p60000\run_algo" -
        Destination "$($_.FullName)" -Recurse -Force
}
```

  - Copy the entire `MOLECULENAME` folder to the HTC access node.

- On HTC:
  - Submit `submit_run_algo.sub` for all the processing folders manually or automatically by running the Shell script `run_submit_run_algo.sh` using the following bash commands executed from the `Bondlengths` directory. (This submission script can be found in the corresponding `Bondlengths` directory in the GitHub repository.)

```
# Run this if the script is not yet executable:
# chmod +x run_submit_run_algo.sh
 ./run_submit_run_algo.sh
```

    When the jobs are done, the `run_algo` folder will be populated with data files named in the format `results_INDEX.pkl`. `INDEX` stands for the `INDEX`th Pauli string.
  - Create a subfolder named `aggregate_results` in an arbitrary processing folder and copy the files `aggregate_results.py`, `submit_aggregate_results.sh`, and `submit_aggregate_results.sub` from the corresponding `.../aggregate_results/` folder in the GitHub repository.
  - Copy the created `aggregate_results` folder to all other processing folders. The following bash script is provided for convenience: (Replace `processing_BL_0p60000` with the name of the processing folder you initially used. Run this script from the `BondLengths` directory.)

```
for dir in processing_BL_*; do
    [ "$dir" != "processing_BL_0p60000" ] &&
    rm -rf "$dir/aggregate_results" &&
    cp -r processing_BL_0p60000/aggregate_results "$dir/";
done
```

- Submit `submit_aggregate_results.sub` for all the processing folders manually or automatically by running the Shell script `run_submit_aggregate_results.sh` using the following bash commands executed from the `Bondlengths` directory. (This submission script can be found in the corresponding `Bondlengths` directory in the GitHub repository.)

```
# Run this if the script is not yet executable:
# chmod +x run_submit_run_algo.sh
./run_submit_aggregate_results.sh
```

  When the jobs are done, the results are saved in the data file `.../aggregate_results/all_results.pkl`.

- Create a folder `.../MOLECULENAME/exact_curve` and copy the files `exact_curve.py`, `exact_curve.sh`, and `exact_curve.sub` from the corresponding folder in the GitHub repository. Submit the job on HTC and the results are saved as `.../exact_curve/exact_curve.csv`.

- Copy the entire `MOLECULENAME` folder from the HTC access node back to the local machine.

- On the local machine:

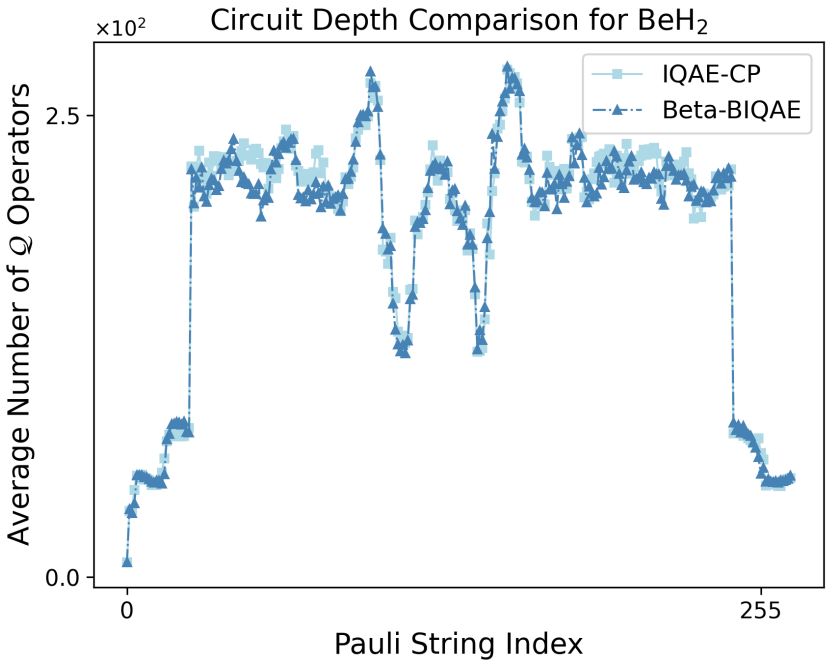
  - Rename `processing_*` folders as `completed_*` folders. The following PowerShell script may be used: (Run this script from the `BondLengths` directory.)

```
Get-ChildItem -Directory -Name |
Where-Object { $_ -like "processing_*" } |
ForEach-Object {
    Rename-Item -Path $_ -NewName ($_.Replace("processing_",
        "completed_"))
}
```

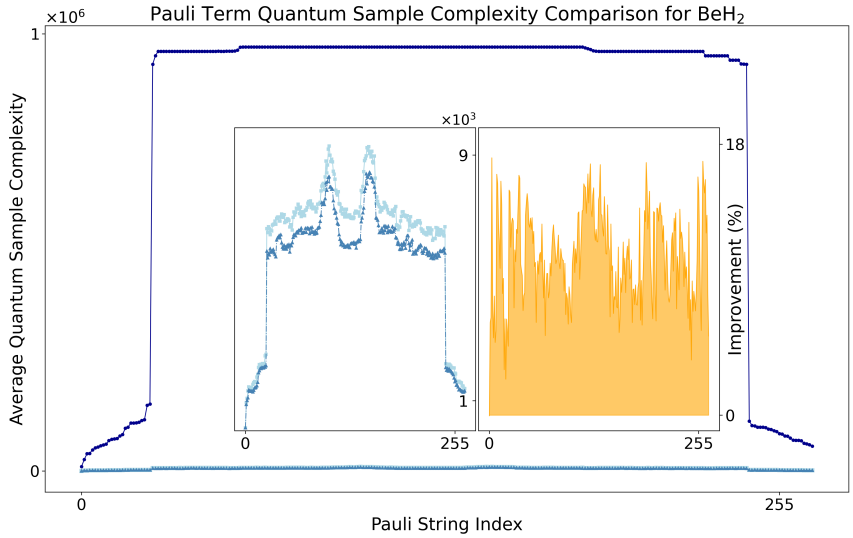
  - Copy `analysis.ipynb`, `analysis_pm.ipynb` from the corresponding folder in the GitHub repository into the local directory `.../MOLECULENAME/BondLengths/`.

  - Execute `analysis_pm.ipynb` to call `analysis.ipynb` to analyze the results and generates the following files for each `completed_BL_BONDLENGTH` folder:

    * `analysis_out.ipynb` — an executed copy of the notebook `.../BondLengths/analysis.ipynb`.
    * `dep.png` — a plot comparing the circuit depths of Beta-BIQAE and IQAE-CP in terms of the number of $\mathcal{Q}$ operators for all Pauli strings at a given bond length. The following is an example for BeH_2 at the equilibrium distance:



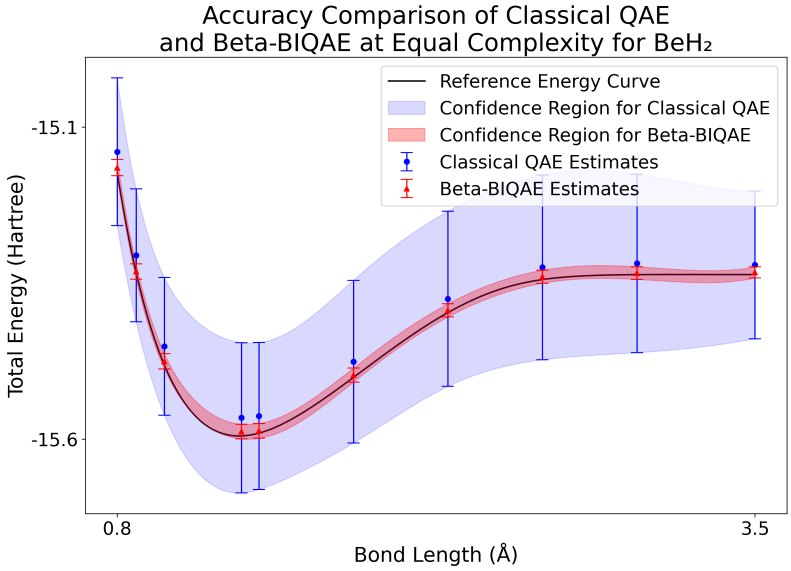
* `sc.png` — a plot comparing the quantum sample complexities of Classical QAE, IQAE-CP, and Beta-BIQAE at the fixed target accuracy for all Pauli strings at a given bond length. The inset shows the percentage improvement of Beta-BIQAE over IQAE-CP. The following is an example for BeH_2 at the equilibrium distance:



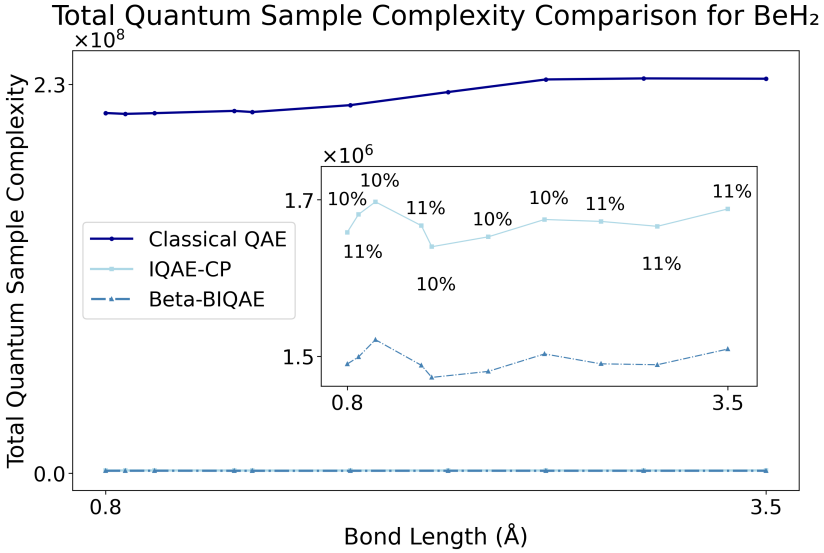
This step also generates `.../BondLength/plot/summary.csv`, which contains the statistics used to plot the aggregated results across different bond lengths.

– Copy `plot.ipynb` from the corresponding folder in the GitHub repository into the local directory `.../BondLengths/plot/` (note each molecule necessitates a tailored version of `plot.ipynb`) and execute it to generate the following plots (see additional details in the accompanying manuscript) in the directory `.../BondLengths/plot/`:

* `compare_acc.png` — a plot comparing the accuracy of Classical QAE and Beta-BIQAE under equal quantum sample complexity, averaged over 200 repetitions.[1] The following is an example for BeH_2:



* `total_sc.png` — a plot comparing the total quantum sample complexity of Classical QAE, IQAE-CP, and Beta-BIQAE at the fixed accuracy with an inset that highlights the percentage improvement of Beta-BIQAE relative to IQAE-CP. The result is averaged over 200 repetitions. The following is an example for BeH_2:



– Set the "Completed" column to be 1 for all corresponding bond lengths in `seleted.csv`.

[1] Note the number of repetitions may be modified in `aggregate_results.py` and `submit_run_algo.sh` for Classical QAE and Beta-BIQAE, respectively.

4 Brief Overview of the `BIQAE/simulations` Folder in the GitHub Repository

`numerical/`
Files for quantum sample complexity analysis with Beta-BIQAE and IQAE-CP

- `all_angle_results/`
  Programs and scripts for quantum sample complexity analysis with collocated ouput
  - `param_list.txt`
    List of parameters θ for which $a = \sin^2\theta$ is to be estimated
  - `simul.py`
    Python script for running experiments and computing the quantum sample complexities
  - `simul.sh`
    Shell script for executing ***`simul.py`*** *on the HTC execution node*
  - `simul.sub`
    HTC Submit file for ***`simul.sh`*** *and simul.py*
  - `results_INDEX.pkl`
    Data file storing the results for the ***`INDEX`****th Pauli string, generated on HTC*
- `all_angle_plot/`
  Files associated with plotting quantum sample complexity against $a = \sin^2\theta,\ \theta = 0°, 5°, \ldots, 90°$ for varying accuracy levels
- `loglog_plot/`
  Files associated with creating scaling plot of the quantum sample complexity of Beta-BIQAE and IQAE-CP for $a = 0.5$
- `figs_radius_ratio/`
  Files associated with plotting the evolution of the radius ratio during Beta-BIQAE stages for the specified angle θ and accuracy ε

`MOLECULENAME/`
Folder for molecular ground-state energy estimation with Beta-BIQAE and IQAE-CP

- `select_BL/`
  Folder associated with specification of bond lengths of interest
  - `selected.csv`
    List of bond lengths of interest
- `vqe_pm.ipynb`
  Jupyter notebook to run ***`vqe.ipynb`*** *for each processing folder*
- `vqe.ipynb`
  Jupyter notebook to determine the ground-state wavefunction at a single bond length
- `BondLengths/`
  Folder associated with simulations for all bond lengths
  - `completed_BL_BONDLENGTH`
    Each completed folder contains the code, circuits, results and plots for each bond length
  - `all_str_A_Q_pm.ipynb`
    Jupyter notebook to run ***`all_str_A_Q.ipynb`*** *for each processing folder lengths*
  - `all_str_A_Q.ipynb`
    Jupyter notebook to construct the quantum circuit for all Pauli strings at a single bond length
  - `circuit_constructors.py`
    Python module imported by ***`all_str_A_Q.ipynb`*** *to construct circuits $\mathcal{A}$ and $\mathcal{Q}$*
  - `analysis_pm.ipynb`
    Jupyter notebook run ***`analysis.ipynb`*** *for each processing folder*
  - `analysis.ipynb`
    Jupyter notebook to analyze and visualize results for a single bond length
  - `run_submit_aggregate_results.sh`
    Auto-submission Shell script to submit ***`submit_aggregate_results.sub`*** *for each processing folder on HTC*

MOLECULENAME/ (continued)

- BondLengths/ (continued)
  - run_submit_run_algo.sh
    Auto-submission Shell script to submit ***submit_run_algo.sub*** *for each processing folder on HTC*
  - plot/
    Folder of associated plots: compare_acc.png *and* total_sc.png
- exact_curve/
  Folder associated with generation of the reference potential energy curve
  - exact_curve.py
    Python file to generate the reference potential energy curve and save the results to exact_ curve.csv
  - submit_exact_curve.sh
    Shell script for executing ***exact_curve.py*** *on the HTC execution node*
  - submit_exact_curve.sub
    HTC Submit file for ***submit_exact_curve.sh*** *and* ***exact_curve.py***
  - exact_curve.csv
    CSV file containing bond lengths, nuclear repulsion energy, and electronic energy used for plotting the reference potential energy curve

5 License

MIT License

A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.