# ADAPT-VQE in PyTorch Lightning: A Reproducible LiH Baseline with PennyLane Lightning

Thomas Roustan

Pierre Lapolla
pro@pierrelapolla.com

***Abstract*—We present a lightweight implementation of the adaptive derivative-assembled pseudo-Trotter variational quantum eigensolver (ADAPT-VQE) built with PennyLane, the Lightning backend, and PyTorch Lightning. The repository targets reproducible experimentation on molecular electronic structure problems through explicit molecule specifications, deterministic training loops, and checkpointed optimization. As a first benchmark, we study lithium hydride (LiH) in STO-3G with Jordan-Wigner mapping and an active space of two electrons in four orbitals. The current implementation combines gradient-based ADAPT operator selection with L-BFGS-B pre-optimization and torch L-BFGS refinement. This paper describes the software architecture, algorithmic choices, and a first in-repository baseline to support future scaling studies and ablation experiments.**

***Index Terms*—quantum chemistry, variational quantum eigensolver, ADAPT-VQE, PennyLane, PyTorch Lightning**

## I. Introduction

Accurate ground-state energy estimation for molecular systems is a central target for quantum algorithms in chemistry. Variational quantum eigensolvers (VQEs) remain among the most practical approaches on near-term hardware because they combine parameterized quantum circuits with classical optimization. However, fixed ansatze can be either too shallow to reach chemical accuracy or too deep to optimize reliably.

ADAPT-VQE addresses this tradeoff by constructing the ansatz iteratively. At each iteration, the method selects the excitation operator with the strongest local energy gradient and only then expands the circuit. This adaptive strategy can reduce unnecessary parameters while preserving a physically motivated operator pool.

This repository implements ADAPT-VQE with a software stack focused on rapid experimentation:

- Hamiltonian construction with PennyLane quantum chemistry tools.
- Quantum simulation on the lightning.qubit backend.
- Optimization orchestration, logging, and checkpointing through PyTorch Lightning.
- Configuration-driven molecule and training settings using Pydantic models.

The first target system is LiH, defined in molecules/LiH.xyz and molecules/LiH.yaml with STO-3G basis, Jordan-Wigner mapping, and active-space parameters chosen to yield an 8-qubit problem. The intent of this paper is to document this baseline implementation and establish a clear starting point for broader benchmarks.

## II. Method

### A. Problem Setup

Given a second-quantized molecular Hamiltonian $H$, we seek the ground-state energy $E_0 = \min_\psi (\psi^\dagger H \psi)$. The variational objective is $E(\theta) = \psi(\theta)^\dagger H \psi(\theta)$, where $\psi(\theta)$ is prepared from a Hartree-Fock reference state and a parameterized excitation circuit.

In this codebase, the molecular Hamiltonian is built in src/app/chem/hamiltonians.py from a MoleculeSpec loaded by src/app/chem/molecule_specs.py. The default configuration (src/app/settings.py) uses molecule identifier LiH and backend device lightning.qubit.

### B. ADAPT-VQE Ansatz Growth

The operator pool is generated from spin-adapted single and double excitations produced by qml.qchem.excitations and mapped to wires with qml.qchem.excitations_to_wires. Starting from an empty ansatz, each ADAPT step:

1) Evaluates a finite-difference gradient proxy for every candidate operator.
2) Selects the operator with maximum absolute gradient.
3) Appends it to the current ansatz.
4) Re-optimizes all selected parameters with L-BFGS-B.

For candidate $i$, the gradient proxy is $g_i \approx \frac{E(\theta, +\varepsilon e_i) - E(\theta, -\varepsilon e_i)}{2\varepsilon}$, with epsilon = vqe.adapt.finite_diff_eps (default 1e-3).

The ADAPT loop is controlled by:

- vqe.adapt.max_steps (default 1 in the current baseline).
- vqe.adapt.grad_tol (default 3e-3).
- vqe.adapt.pretrain_maxiter (default 30).
- Optional pool draining (vqe.adapt.drain_pool = true).

### C. Lightning Training Objective

After ansatz construction, the trainable parameters are refined in a PyTorch Lightning module (src/app/adapt_vqe_module.py). The loss is $L(\theta) = |E(\theta) - E_0|$, where $E_0$ is computed by sparse exact diagonalization (eigsh) of the qubit Hamiltonian.

The training loop uses a synthetic step dataset (src/app/steps_data_module.py) to execute a configurable number of optimization steps per epoch, with checkpointing handled by Lightning callbacks (src/app/trainer.py).

## III. Preliminary Baseline

The repository includes an initial single-line summary in Results/adapt_vqe_results_single_line.csv for LiH with one ADAPT expansion step. That run reports:

- Final variational energy: -7.862843036651611 Hartree.
- Reference energy: -7.8644891084057305 Hartree.
- Absolute gap: approximately 1.65e-3 Hartree.

These numbers should be treated as a first reproducibility checkpoint rather than a final performance claim. The immediate next stage is to increase vqe.adapt.max_steps, compare against the fixed UCCSD baseline already implemented in src/app/uccsd_vqe_module.py, and characterize convergence versus optimization budget.

## IV. Implementation Notes

The runtime flow starts in src/app/__main__.py, which instantiates LightningManager (src/app/lightning_manager.py) and launches training. The manager performs:

1) deterministic seeding (seed_everything),
2) construction of the synthetic steps data module,
3) selection of either ADAPT-VQE or UCCSD-VQE based on vqe.algorithm,
4) creation of a PyTorch Lightning trainer with checkpoint and optional W&B logging.

Within ADAPT-VQE (src/app/adapt_vqe_module.py), two optimization phases are used:

1) inner ansatz-growth pre-optimization via SciPy minimize(..., method: "L-BFGS-B"),
2) outer training optimization via torch LBFGS in Lightning.

This separation allows each new ADAPT expansion to start from locally improved parameters before global refinement across training steps.

## V. Reproducibility and Configuration

Experiment behavior is defined by the typed settings model in src/app/settings.py. For the default baseline:

- seed = 2002,
- vqe.algorithm = "adapt",
- train.max_epochs = 1,
- train.steps_per_epoch = 100,
- vqe.adapt.max_steps = 1.

Molecular geometry and chemistry metadata are split between molecules/LiH.xyz and molecules/LiH.yaml, then merged through load_molecule_spec in src/app/chem/molecule_specs.py. This decouples structure from algorithmic hyperparameters and keeps benchmark definitions explicit.

## VI. Limitations and Next Steps

The current implementation is intentionally minimal and provides a stable baseline rather than a fully tuned ADAPT-VQE study. Key limitations are:

- one ADAPT growth step by default,
- finite-difference gradient screening for pool selection (cost scales with pool size),
- CPU-oriented default trainer settings.

Near-term improvements include larger ADAPT depths, alternative operator ranking heuristics, and side-by-side convergence curves against the existing UCCSD module under matched optimization budgets.

## VII. Conclusion

This repository delivers a concise, reproducible ADAPT-VQE baseline for LiH using PennyLane Lightning and PyTorch Lightning. The code organizes chemistry specification, ansatz construction, and training orchestration into separate modules, enabling controlled experiments and straightforward extension to additional molecules and algorithmic variants.

## References