MLQD: A package for machine learning-based quantum dissipative dynamics

Arif Ullah$^1$ and Pavlo O. Dral$^2$

$^1$School of Physics and Optoelectronic Engineering, Anhui University, Hefei 230601, China
$^2$State Key Laboratory of Physical Chemistry of Solid Surfaces, Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Department of Chemistry, and College of Chemistry and Chemical Engineering, Xiamen University, Xiamen 361005, China

(*Electronic mail: ua2024@xmu.edu.cn)
(*Electronic mail: dral@xmu.edu.cn)

(Dated: 20 April 2023)

Machine learning has emerged as a promising paradigm to study the quantum dissipative dynamics of open quantum systems. To facilitate the use of our recently published ML-based approaches for quantum dissipative dynamics, here we present an open-source Python package MLQD (https://github.com/Arif-PhyChem/MLQD), which currently supports the three ML-based quantum dynamics approaches: (1) the recursive dynamics with kernel ridge regression (KRR) method, (2) the non-recursive artificial-intelligence-based quantum dynamics (AIQD) approach and (3) the blazingly fast one-shot trajectory learning (OSTL) approach, where both AIQD and OSTL use the convolutional neural networks (CNN). This paper describes the features of the MLQD package, the technical details, optimization of hyperparameters, visualization of results, and the demonstration of the MLQD’s applicability for two widely studied systems, namely the spin-boson model and the Fenna–Matthews–Olson (FMO) complex. To make MLQD more user-friendly and accessible, we have made it available on the Python Package Index (PyPi) platform and it can be installed via pip install mlqd. In addition, it is also available on the XACS cloud computing platform (https://XACScloud.com) via the interface to the MLATOM package (http://MLatom.com).

I. INTRODUCTION

The basic time-dependent Schrödinger equation describes the unitary dynamics of an isolated quantum system. However, isolated quantum systems are an idealistic approximation with many limitations as in real world, systems are always coupled to an environment. Thus, to study quantum systems in reality, it is important to incorporate the effects of environment which gives rise to dephasing and dissipation. Systems with dephasing and dissipation (open quantum systems or quantum dissipative systems) are ubiquitous and can be exploited in environment-assisted quantum transport, chemical and biological systems, quantum information processing and quantum computing, defect tunneling in solids, quantum electrodynamics, superconducting junctions, and quarkonium transport in a hot nuclear environment. Exact solution of Schrödinger equation for open quantum systems is a daunting task and in most cases is not feasible because of exponential growth in Hilbert space dimension and a large number of environment degrees of freedom. Thus, many approximations are adopted such as averaging out environment degrees of freedom and classical description of the system and/or environment.

In the past three decades, great progress has been made in the development of theoretical approaches for open quantum systems. These approaches include the perturbative Redfield equation, the exact Nakajima–Zwanzig formalism and its kernel-based expansions, the quantum-classical and fully classical approaches, Green’s function formalism, the transfer tensor method and its extension, the multi-configuration time-dependent Hartree (MCTDH) the pseudo-mode approach, the reaction coordinate (RC) approach, the quantum Monte Carlo (QMC), the time-dependent numerical renormalization group (tDMRG), the density matrix renormalization group (tDMRG), the polaron transformation, the time evolving density matrix using the orthogonal polynomial algorithm (TEDOPA), the quasidiabatic propagator path integral (QuAPI), the numerical variational method (NVM), the automated compression of environment (ACE) method, the hierarchy equations of motion (HEOM), the dissipation equation of motion (DEOM), and the stochastic equation of motion (SEOM). These methods on their own are successful attempts to solve the complex multibody problem of open quantum systems, however, the prohibitive increase in their computational cost with the system size limits their applicability.

In the past two decades, data proliferation has led to the advent of machine learning (ML) methods. ML has been described as a fourth pillar in Science next to experiment, theory and simulation. Without discussing the broad use of ML, in recent years, ML has seen a surge in the field of quantum dynamics in general and, in particular, quantum dissipative dynamics. To be specific, ML has been used to predict molecular configurations in the four-dimensional space, the relaxation dynamics of a two-state system the excitation energy transfer in Fenna–Matthews–Olson light-harvesting complex and average exciton
transfer times and transfer efficiencies, parameters of Hamiltonian, evolution of the proton density in a potential well and vibronic Hamiltonians as a direct function of time. Machine learning has also been extended to Meyer–Miller mapping based symmetrical quasi-classical and fewest-switches surface hopping dynamics.

The rapid development of new ML methods for quantum dissipative dynamics led to so-far not well-organized and scattered software implementations. In this article, we present an open-source software package MLQD, version 1, which provides a framework for ML-based quantum dissipative dynamics implementations. MLQD is incorporated with kernel ridge regression (KRR) and convolutional neural networks (CNN) models and a user can train and predict dynamics following both recursive and non-recursive approaches. We follow our recently published recursive KRR-based approach on the relaxation dynamics of the spin-boson model and non-recursive CNN-based AIQD and OSTL approaches. MLQD also supports hyperparameter optimization using MLATOM’s grid search functionality for KRR and Bayesian methods with Tree-structured Parzen Estimator (TPE) for CNN models via the HYPEROPT package. In addition, we also incorporate the visualization of results by auto-plotting. MLQD has also been interfaced with the MLATOM package which allows the user to run MLQD on the XACS cloud computing platform.

In the following, we provide an overview of the MLQD package, the theory of implemented approaches, technical details, optimization of hyperparameters, visualization of results, and the demonstration of MLQD’s applicability for two widely studied systems, namely the spin-boson model and the Fenna–Matthews–Olson (FMO) complex.

II. MLQD PACKAGE OVERVIEW

MLQD package is written in Python language and provides the implementation of our recently proposed ML-based approaches for quantum dissipative dynamics. This section lays out the concise documentation of theory, code design, use and implementation. Coming to the code design, we provide a simplified flowchart of MLQD architecture in Fig. 1. MLQD comes with two main features, createQDmodel which trains a QD model and useQDmodel which uses the already trained QD model for dynamics propagation. MLQD has also the feature of preparing the training data X and Y considering the training grid search functionality in the backend) and HYPEROPT library in the case of AIQD and OSTL approaches.

A. ML-based quantum dynamics approaches

ML-based quantum dynamics approaches can be divided into two main categories: recursive approaches and non-recursive.

1. Recursive approaches

In recursive approaches, the future dynamics depends on its past dynamics which in nature is the same as in traditional quantum dynamics approaches. In these approaches, an initial shot-time dynamics of time-length \( t_m \) is used as an input to predict system dynamics at the next time step \( t_{m+1} \), i.e., \( \rho_s(t_{m+1}) = f_{\text{ML}}(\rho_s(t_0), \rho_s(t_1), \ldots, \rho_s(t_m)) \) where \( \rho_s(t) \) is the reduced density matrix (RDM) of the system at time \( t \) (see Section III) and \( f_{\text{ML}} \) is ML function. To incorporate the recursiveness of the dynamics, in the next step, the predicted dynamics at \( t_{m+1} \) is appended to the end of the input vector while the value at the start of the input vector is dropped which leads to a new input of the same size as the old input. The new input is used to predict system dynamics at the next time step \( t_{m+2} \), i.e., \( \rho_s(t_{m+2}) = f_{\text{ML}}(\rho_s(t_1), \rho_s(t_2), \ldots, \rho_s(t_{m+1})) \). To predict system dynamics at \( t_{m+3} \), the predicted dynamics at \( t_{m+2} \) is included with the drop of time step at the first end and this process continues till the last time step. In Fig. 2, we have elaborated on the transformation of the training trajectories into the training data for machine learning. Following Ref. 91, in MLQD, the recursive approach is adopted only for the KRR model and we will refer to this approach as the KRR approach (see Subsection II B 1).

2. Non-recursive approaches

Non-recursive approaches have the downside that they should be run sequentially, one step at a time, which is intrinsically computationally costly. An additional downside is the potential for error accumulation at each time step. To develop approaches that are free of these downsides, we have recently proposed two non-recursive approaches AIQD (artificial intelligence-based quantum dynamics) and OSTL (one-shot trajectory learning). Non-recursive approaches are based on neural networks, details are given in Subsection II B 2. In the following, we briefly describe both approaches.
FIG. 1. A simplified flowchart of the MLQD package. MLQD package comes with six main python modules: `evolution.py`, `train_ml.py`, `ml_dyn.py`, `prep_input.py`, `hyperopt_optim.py` and `cnn.py`. The `evolution.py` module is the headquarter of the MLQD package which controls all the operations including, training a QD model and using the already trained QD model for dynamics propagation–calling `train_ml.py` and `ml_dyn.py` with the keywords `createQDmodel` and `useQDmodel`, respectively. Calling `train_ml.py` in return calls the `cnn.py` and MLAtom package \cite{108-110} to train a CNN model and a KRR model, respectively. MLQD has also the feature of preparing the training data X and Y (using `prep_input.py`) considering the training trajectories are given in the same format as in our QDSET-1. \cite{114} MLQD can also optimize the hyperparameters using the grid search for KRR model (utilizing MLAtom in the backend) and HYPEROPT library (using `hyperopt_optim.py`) \cite{112} in the case of AIQD and OSTL approaches. MLQD’s architecture allows for future developments with independent modules easily incorporated in the `evolution.py` module.
\( \hat{\rho}_i(t) \) of \( N \times N \) dimensions

\[
\begin{bmatrix}
\rho_{11}^{(i)}(t_0) & \ldots & \rho_{N1}^{(i)}(t_0) \\
\rho_{12}^{(i)}(t_0) & \ldots & \rho_{N2}^{(i)}(t_0) \\
\vdots & \ddots & \vdots \\
\rho_{1N}^{(i)}(t_0) & \ldots & \rho_{NN}^{(i)}(t_0)
\end{bmatrix}
\begin{bmatrix}
\rho_{11}^{(i)}(t_1) & \ldots & \rho_{N1}^{(i)}(t_1) \\
\rho_{12}^{(i)}(t_1) & \ldots & \rho_{N2}^{(i)}(t_1) \\
\vdots & \ddots & \vdots \\
\rho_{1N}^{(i)}(t_1) & \ldots & \rho_{NN}^{(i)}(t_1)
\end{bmatrix}
\begin{bmatrix}
\rho_{11}^{(i)}(t_m) & \ldots & \rho_{N1}^{(i)}(t_m) \\
\rho_{12}^{(i)}(t_m) & \ldots & \rho_{N2}^{(i)}(t_m) \\
\vdots & \ddots & \vdots \\
\rho_{1N}^{(i)}(t_m) & \ldots & \rho_{NN}^{(i)}(t_m)
\end{bmatrix}
\begin{bmatrix}
\rho_{11}^{(i)}(t_{m+1}) & \ldots & \rho_{N1}^{(i)}(t_{m+1}) \\
\rho_{12}^{(i)}(t_{m+1}) & \ldots & \rho_{N2}^{(i)}(t_{m+1}) \\
\vdots & \ddots & \vdots \\
\rho_{1N}^{(i)}(t_{m+1}) & \ldots & \rho_{NN}^{(i)}(t_{m+1})
\end{bmatrix}
\begin{bmatrix}
\rho_{11}^{(i)}(t_T) & \ldots & \rho_{N1}^{(i)}(t_T) \\
\rho_{12}^{(i)}(t_T) & \ldots & \rho_{N2}^{(i)}(t_T) \\
\vdots & \ddots & \vdots \\
\rho_{1N}^{(i)}(t_T) & \ldots & \rho_{NN}^{(i)}(t_T)
\end{bmatrix}
\]

The number of training points is equal to \((t_T - t_m)/\Delta t\)

\[
\begin{bmatrix}
X_{t_0,1,\ldots,n} \\
X_{t_1,2,\ldots,n+1} \\
X_{t_2,3,\ldots,n+2} \\
\vdots \\
X_{t_m,\ldots,n-1,n,n+1}
\end{bmatrix}
\begin{bmatrix}
Y_{t_0+1} \\
Y_{t_0+2} \\
Y_{t_0+3} \\
\vdots \\
Y_{t_m+1}
\end{bmatrix}
\]

\[
\begin{align*}
\begin{bmatrix}
\hat{\rho}_{11}^{(i)}(t_k) & \ldots & \hat{\rho}_{N1}^{(i)}(t_k) \\
\hat{\rho}_{12}^{(i)}(t_k) & \ldots & \hat{\rho}_{N2}^{(i)}(t_k) \\
\vdots & \ddots & \vdots \\
\hat{\rho}_{1N}^{(i)}(t_k) & \ldots & \hat{\rho}_{NN}^{(i)}(t_k)
\end{bmatrix}
& = \\
\begin{bmatrix}
\hat{\rho}_{11}^{(i)}(t_{k+1}) & \ldots & \hat{\rho}_{N1}^{(i)}(t_{k+1}) \\
\hat{\rho}_{12}^{(i)}(t_{k+1}) & \ldots & \hat{\rho}_{N2}^{(i)}(t_{k+1}) \\
\vdots & \ddots & \vdots \\
\hat{\rho}_{1N}^{(i)}(t_{k+1}) & \ldots & \hat{\rho}_{NN}^{(i)}(t_{k+1})
\end{bmatrix}
\end{align*}
\]

**FIG. 2.** Transformation of the quantum dynamics trajectories into the training data for the recursive approach on an example of the RDM terms.

### a. AIQD approach

In the AIQD approach,\(^9^8\) the dynamics is predicted as a continuous function of time, i.e., dynamics property (system’s state) can be predicted at an arbitrary time without step-wise dynamics propagation. Unlike recursive approaches, AIQD does not need to use short-time dynamics as an input. Input, in addition to time, includes simulation parameters such as temperature \(T\), characteristic frequency \(\gamma\) and system-bath coupling strength \(\lambda\). The dynamics corresponding to these parameters is used as a target vector (for training) and predicted by the model when used for inference. In addition, as all time steps are independent of each other, parallel computation of all time steps is possible. Fig. 3 shows data preparation for the AIQD approach where each trajectory transforms into the training points equal to the number of time steps. We also note that to make predictions in asymptotic limit and cover different time regions with similar accuracy, the time input is mapped into a multi-dimensional vector after normalization with the logistic function.

### b. OSTL approach

Similar to the AIQD approach, in the OSTL approach\(^9^9\) dynamics property (system’s state) can be predicted without step-wise dynamics propagation. However, in contrast to the AIQD, OSTL predicts entire trajectory in one
ith trajectory of RDM $\tilde{\rho}_i(t)$ with $N \times N$ dimensions

\[
\begin{bmatrix}
\tilde{\rho}_{s_{i1}}^{(t)}(t_0) & \ldots & \tilde{\rho}_{s_{i1}}^{(t)}(t_0) \\
\tilde{\rho}_{s_{i2}}^{(t)}(t_0) & \ldots & \tilde{\rho}_{s_{i2}}^{(t)}(t_0) \\
\vdots & \vdots & \vdots \\
\tilde{\rho}_{s_{iN}}^{(t)}(t_0) & \ldots & \tilde{\rho}_{s_{iN}}^{(t)}(t_0)
\end{bmatrix}
\begin{bmatrix}
\tilde{\rho}_{s_{i1}}^{(t_1)}(t_1) & \ldots & \tilde{\rho}_{s_{i1}}^{(t_1)}(t_1) \\
\tilde{\rho}_{s_{i2}}^{(t_1)}(t_1) & \ldots & \tilde{\rho}_{s_{i2}}^{(t_1)}(t_1) \\
\vdots & \vdots & \vdots \\
\tilde{\rho}_{s_{iN}}^{(t_1)}(t_1) & \ldots & \tilde{\rho}_{s_{iN}}^{(t_1)}(t_1)
\end{bmatrix}
\begin{bmatrix}
\tilde{\rho}_{s_{i1}}^{(t_M)}(t_M) & \ldots & \tilde{\rho}_{s_{i1}}^{(t_M)}(t_M) \\
\tilde{\rho}_{s_{i2}}^{(t_M)}(t_M) & \ldots & \tilde{\rho}_{s_{i2}}^{(t_M)}(t_M) \\
\vdots & \vdots & \vdots \\
\tilde{\rho}_{s_{iN}}^{(t_M)}(t_M) & \ldots & \tilde{\rho}_{s_{iN}}^{(t_M)}(t_M)
\end{bmatrix}
\]

\[
\mathcal{Y}(t_0) \quad \mathcal{Y}(t_1) \quad \ldots \quad \mathcal{Y}(t_M)
\]

The number of training points is equal to number of time steps

\[
\begin{align*}
\lambda^{(i)} & \quad \gamma^{(i)} & \quad T^{(i)} & \quad \{f_j(t_0)\} & \quad \mathcal{Y}(t_0) \\
\lambda^{(i)} & \quad \gamma^{(i)} & \quad T^{(i)} & \quad \{f_j(t_1)\} & \quad \mathcal{Y}(t_1) \\
\vdots & \quad \vdots & \quad \vdots & \quad \vdots & \quad \vdots \\
\lambda^{(i)} & \quad \gamma^{(i)} & \quad T^{(i)} & \quad \{f_j(t_k)\} & \quad \mathcal{Y}(t_k) \\
\lambda^{(i)} & \quad \gamma^{(i)} & \quad T^{(i)} & \quad \{f_j(t_{k+1})\} & \quad \mathcal{Y}(t_{k+1}) \\
\vdots & \quad \vdots & \quad \vdots & \quad \vdots & \quad \vdots \\
\lambda^{(i)} & \quad \gamma^{(i)} & \quad T^{(i)} & \quad \{f_j(t_M)\} & \quad \mathcal{Y}(t_M)
\end{align*}
\]

\[
\mathcal{Y}(t) = \tilde{\rho}_{s_{i1}}^{(t)}(t), \ldots, \tilde{\rho}_{s_{iN}}^{(t)}(t), \tilde{\rho}_{s_{i1}}^{(t)}(t), \ldots, \tilde{\rho}_{s_{iN}}^{(t)}(t), \ldots, \tilde{\rho}_{s_{iN-1}}^{(t)}(t), \tilde{\rho}_{s_{iN}}^{(t)}(t)
\]

FIG. 3. Transformation of the ith trajectory of RDM $\tilde{\rho}_i(t)$ in the AIQD approach. The $\tilde{\rho}_i(t)$ at each time step transforms into a vector $\mathcal{Y}(t)$ with dimension $M = \text{number of sites} + (2 \times \text{number of the upper off-diagonal terms})$. As in RDM $\tilde{\rho}_{\text{nn}}(t) = \tilde{\rho}_{\text{nn}}(t)$ (n $\neq m$), only the upper off-diagonal terms are learned. In addition, the real and imaginary parts of each off-diagonal term are separated. Simulation parameters $\lambda^{(i)}$, $\gamma^{(i)}$, and $T^{(i)}$ are the reorganization energy, characteristic frequency, and temperature of the ith trajectory in their respective order. The $\{f(t)\}$ is a set of logistic functions normalizing the dimension of time, i.e., $f_j(t) = a/(1 + b \exp(-(t + c_j)/d))$ where $a$, $b$, $d$ are fixed constants while $c_j = 5j - 1$ having $j$ as a natural number, i.e., $j \in \{0, 1, 2, 3, \ldots\}$.

shot for a discretized set of time steps. OSTL also includes the simulation parameters (i.e., $\lambda$, $\gamma$, $T$) as an input, while time is not included. As shown in Fig. 4, in the OSTL approach, each trajectory transforms into a single training point thus significantly reducing the cost of training. The full-time dynamics is predicted using the multi-output feature which obviates invoking the entire ML structure for each time step, thus leading to a significant speed up in dynamics prediction too.

B. Machine learning models

1. Kernel ridge regression

In kernel ridge regression (KRR), for a given input vector $\mathbf{r}$, a function $f(\mathbf{r})$ is approximated with the following expansion\cite{108,115}

\[
f(\mathbf{r}) = \sum_{i}^{N_r} \eta_i K(\mathbf{r}, \mathbf{r}_i),
\]
ith trajectory of the RDM $\hat{\rho}_i(t)$ with $N \times N$ dimensions

\[
\begin{bmatrix}
\hat{\rho}^{(i)}_{00}(t_0) & \ldots & \hat{\rho}^{(i)}_{10}(t_0) \\
\hat{\rho}^{(i)}_{01}(t_0) & \ldots & \hat{\rho}^{(i)}_{11}(t_0) \\
\vdots & \vdots & \vdots \\
\hat{\rho}^{(i)}_{0N_1}(t_0) & \ldots & \hat{\rho}^{(i)}_{1N_1}(t_0) \\
\end{bmatrix} \rightarrow
\begin{bmatrix}
\hat{\rho}^{(i)}_{00}(t_1) & \ldots & \hat{\rho}^{(i)}_{10}(t_1) \\
\hat{\rho}^{(i)}_{01}(t_1) & \ldots & \hat{\rho}^{(i)}_{11}(t_1) \\
\vdots & \vdots & \vdots \\
\hat{\rho}^{(i)}_{0N_1}(t_1) & \ldots & \hat{\rho}^{(i)}_{1N_1}(t_1) \\
\end{bmatrix} \rightarrow
\begin{bmatrix}
\hat{\rho}^{(i)}_{00}(t_M) & \ldots & \hat{\rho}^{(i)}_{10}(t_M) \\
\hat{\rho}^{(i)}_{01}(t_M) & \ldots & \hat{\rho}^{(i)}_{11}(t_M) \\
\vdots & \vdots & \vdots \\
\hat{\rho}^{(i)}_{0N_1}(t_M) & \ldots & \hat{\rho}^{(i)}_{1N_1}(t_M) \\
\end{bmatrix}
\]

A single training point

\[\mathcal{Y}(t) = \hat{\rho}_{00}^{(i)}(t), \ldots, \hat{\rho}_{10}^{(i)}(t), \hat{\rho}_{01}^{(i)}(t), \ldots, \hat{\rho}_{11}^{(i)}(t), \ldots, \hat{\rho}_{0N_1}^{(i)}(t), \hat{\rho}_{1N_1}^{(i)}(t)\]

FIG. 4. Transformation of the ith trajectory of RDM $\hat{\rho}_i(t)$ in the OSTL approach. The $\hat{\rho}_i(t)$ at each time step transforms into a vector $\mathcal{Y}(t)$ with dimension $M = \text{number of sites} + (2 \times \text{number of the upper off-diagonal terms})$. As in RMD $\hat{\rho}_i(t) = \hat{\rho}_{i}^{(n)}(t)$, only the upper off-diagonal terms are learned. In addition, the real and imaginary parts of each off-diagonal term are separated. Simulation parameters $\lambda^{(i)}$, $\gamma^{(i)}$ and $T^{(i)}$ are the reorganization energy, characteristic frequency, and temperature of the ith trajectory in their respective order.

where $N_r$ is the number of training points, $\eta = \{ \eta_i \}$ is a vector of regression coefficients and $K(r, r')$ is a kernel function measuring the similarity between two vectors $r$ and $r'$. The very common kernel is the Gaussian kernel\textsuperscript{108,116}

\[K(r, r') = \exp \left( -\frac{\|r - r'\|^2}{2\sigma^2} \right), \]

(2)

where $\sigma$ is a hyperparameter defining the length scale. It is worth emphasizing that many other kernel functions $K(r, r')$ such as Matérn and exponential kernels\textsuperscript{117,118} can also be used, however, based on our previous studies,\textsuperscript{91,92} these kernels do not outperform the Gaussian kernel, thus in MLQD, we only use the Gaussian kernel.

To find the regression coefficients $\eta$ in Eq. (1), MLQD uses MLATOM package\textsuperscript{108–110} in the backend and solves the following equation

\[(K + \lambda I) \eta = y, \]

(3)

where $K$ is the kernel matrix, $I$ is the identity matrix, $y$ is the vector of target values, and $\lambda$ represents a non-negative regularization hyperparameter.

2. Neural networks

Carefully-constructed neural networks (NN) models, consisting of neurons organized in layers, can be considered as universal approximators of any continuous function.\textsuperscript{119–121} There is a large group of NN models that can be used to learn and predict quantum dynamics,\textsuperscript{92} however here we will restrict ourselves to the convolutional neural network (CNN) model as it is the only NN model implemented in the MLQD package. As shown in Fig. 5, the CNN model in MLQD package comprises of an input layer, two or three convolutional layers, a maximum pooling layer, a flatten layer, two or three dense layers and an output layer. In CNN, convolutional layers extract the features such as time dependence, the maximum pooling layer decreases the size of the feature map, the flatten layer transforms the output from the maximum pooling layer into one-dimensional vector and then we have the dense layers which are the common feed-forward neural networks.
FIG. 5. A flowchart of CNN model in MLQD which consists of an input layer, convolutional layers, a maximum pooling layer (not shown), a flatten layer, dense layers and an output layer.

C. Optimization of hyperparameters

In the MLQD package, we provide a set of default values for hyperparameters, however, the hyperparameter optimization is also possible. In the case of KRR approach, MLQD uses MLATOM’s grid search functionality and optimizes its both hyperparameters $\sigma$ and $\lambda$. In the case of AIQD and OSTL approaches, MLQD uses the HYPEROPT library\(^\text{112}\) for the optimization of CNN structure. HYPEROPT uses Bayesian optimization with the parallel infrastructure for a fast search of best hyperparameters in a defined multidimensional space. We optimize the number of filters, kernels size, the number of neurons, learning rate and the number of batches in a predefined multidimensional space. The number of hidden convolutional layers and hidden dense layers are optimized between two numbers, i.e., \{2, 3\}.

D. Plotting

For a better understanding of any approach, visualization of results is necessary and in many cases such as cloud computing, auto-plotting provides complete mouse-click computing. We incorporate this functionality in MLQD, where the predicted dynamics is plotted against the reference trajectory providing a clear visualization of the predicted dynamics.

III. APPLICATIONS

In this section, we present two case studies to highlight the applications of the MLQD package. We consider the two widely studied systems, namely the spin-boson model and the 8-site Fenna–Matthews–Olson (FMO) complex.

Before that, we briefly overview the general theory behind open quantum systems. A quantum system coupled to its outside environment (bath) is regarded as an open quantum system with the dynamics governed by the following Hamiltonian

$$\hat{H} = \hat{H}_s + \hat{H}_b + \hat{H}_{sb},$$

where $\hat{H}_s$ and $\hat{H}_b$ represent the Hamiltonian for the system and the outside environment (bath), respectively. The last term $\hat{H}_{sb}$ incorporates the interaction between the system and the environment. To propagate quantum dynamics, Liouville–von Neumann equation can be employed

$$\dot{\rho}(t) = -i[\hat{H}, \rho(t)],$$

where $\rho(t)$ is the density matrix at time $t$ and $\hbar$ is set to 1. In system-bath approaches,\(^{51,52,64,70}\) calculations are usually simplified by considering system and environment uncorrelated at $t = 0$, i.e., $\rho(0) = \rho_s \rho_b$ where $\rho_s$ is the density matrix of the system and $\rho_b$
denotes density matrix of the environment. As we are interested only in the system, we can take a partial trace over environment degrees of freedom
\[
\hat{\rho}_k = \text{Tr}_b [U(t, 0)\rho(0)U^\dagger(t, 0)],
\]
where \(\hat{\rho}_k\) is the density matrix of the reduced system (reduced density matrix (RDM)) and \(U(t, 0)\) \((U^\dagger(t, 0))\) is the propagation operator forward (backward) in time and \(\text{Tr}_b\) is the partial trace over environment degrees of freedom. In reality, the term open quantum system is applicable to most of the systems, however, because of the curse of dimensionality, not all of them are easy to be theoretically handled. In the following, we present a brief theory of two broadly studied pedagogical systems, the two-states spin-boson model and the FMO complex.

### A. Case study 1: Relaxation dynamics of spin-boson model

As a first case study, we consider the relaxation dynamics of excited state \(|e\rangle\) in spin-boson model. The spin-boson model is a two-state system coupled with an environment of an infinite number of non-interacting harmonic oscillators. The Hamiltonian of the composite system (two-states system + environment) is expressed as
\[
H = \varepsilon (|e\rangle \langle e| - |g\rangle \langle g|) + \Delta (|e\rangle \langle g| + |g\rangle \langle e|) + \sum_{k=1}^{\infty} \omega_k b_k^\dagger b_k + (|e\rangle \langle e| - |g\rangle \langle g|) \sum_{k=1}^{\infty} c_k (b_k^\dagger + b_k),
\]
where \(|e\rangle\) and \(|g\rangle\) denote the two states of the system, \(\varepsilon\) is the energy difference between the two states and \(\Delta\) is the tunneling splitting. The \(b_k^\dagger (b_k)\) denotes the creation (annihilation) operator in the environment Hilbert space and \(\omega_k\) is the frequency corresponding to \(k\) mode. The last term in Eq. (7) incorporates the interaction between the system and environment with \(c_k\) as the coupling strength between the system’s operator and \(k\) environment mode. The effects of the environment on system dynamics are described by the spectral density of the environment
\[
J(\omega) = \sum_k \alpha_k \delta(\omega - \omega_k),
\]
where \(\alpha_k = \frac{\pi}{2} \frac{c_k^2}{m_k \delta_k}\). Here we adopt the Ohmic spectral density function with the Drude–Lorentz cut-off
\[
J(\omega) = 2\lambda \frac{\gamma \omega}{\omega^2 + \gamma^2},
\]
where \(\lambda\) is the reorganization energy and \(\gamma\) is the characteristic frequency or the inverse of environment relaxation time, i.e., \(\gamma = 1/\tau\).

For our example, we use the spin-boson data set from our recently published QD3SET-1 database.\(^{114}\) The mentioned data set consists of 1000 trajectories generated for each possible combination of the following parameters; \(\tilde{\varepsilon} = \varepsilon/\Delta = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}\), \(\tilde{\gamma} = \gamma/\Delta = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}\), and \(\tilde{\beta} = \beta/\Delta = \{0.1, 0.25, 0.5, 0.75, 1.0\}\), where the tunneling matrix element \(\Delta\) is set as an energy unit. Data is generated with the HEOM method implemented in the QuTiP software package\(^{123}\) and each trajectory is propagated up to \(t\Delta = 20\) with the time step \(dt\Delta = 0.05\).

In this case study, we use all three available approaches (i.e., KRR, AIQD and OSTL) to predict the relaxation dynamics in the two possible cases, namely the symmetric case \(\tilde{\varepsilon} = 0\) and the asymmetric case \(\tilde{\varepsilon} = 1.0\). Before training, we divide the spin-boson data set into the training set (400 trajectories for each case) and the test set (100 trajectories for each case). The division is based on farthest-point sampling which selects the most distant points in a three-dimensional Euclidean space\(^{98}\) \((\lambda, \tilde{\gamma}, \tilde{\beta})\), thus efficiently covering the parameter space in comparison to random sampling.\(^{108}\) Keeping in mind the high computational cost of KRR, we sample the data for training with comparatively larger time step \(dt\Delta = 0.1\).

For KRR, a short-time trajectory of \(t_{\text{m}}\Delta = 4.0\) is used as an input and following the algorithm described in Fig. 2,\(^{91}\) we transform the trajectories beyond \(t_{\text{m}}\Delta = 4.0\) into target values. We train separate KRR models for each diagonal element of the RDM. As \(\hat{\rho}_{12} = \hat{\rho}_{11}^\dagger\), we learn only the upper off-diagonal term where we train a separate KRR model for real and imaginary parts. After training, we provide a reference short-time trajectory of time-length \(t_{\text{m}}\Delta = 4.0\) to initiate recursive propagation with the trained KRR model (the recursive propagation is beyond this short-time trajectory dynamics). In Fig. 6(A), we show population and coherence for symmetric case \(\tilde{\varepsilon} = 0\) while results for asymmetric case \(\tilde{\varepsilon} = 1\) are shown in Fig. S1(A).

In the case of AIQD and OSTL approaches, we prepare training data following Figs. 3 and 4 in their respective order. In both approaches, we use \(\tilde{\gamma}_{\text{max}} = 10.0\) for \(\tilde{\gamma}\) and 1.0 for the remaining simulation parameters as normalization factors. To normalize the dimension of time, in AIQD we use a set of 10 logistic functions, i.e., \(f_j(t) = a/(1 + b \exp(-t + e_j)/d)\) where \(j = 0, 1, 2, \ldots, 9\). We set \(a = 1.0, b = 15.0, d = 1.0\) and \(c = 5j - 1\). In List. 1, we show an example of MLQD input for creating (training) a CNN model following the OSTL approach. After training, by providing the trained AIQD model, values of the simulation parameters
and time, MLQD predicts the corresponding RDM $\hat{\rho}_s$. In the OSTL approach, the RDM $\hat{\rho}_s$ is predicted for the whole time range, i.e., $t_{m\Delta} = 0, \ldots, t_{m\Delta} = 20$. In Figs. 6(B) and (C), we show the time evolution of RDM’s diagonal (population) and off-diagonal terms (coherence) for a set of test parameters. Results for the asymmetric case $\hat{\varepsilon} = 1$ are given in Figs. S1(B) and (C).

from mlqd.evolution import quant_dyn
param={
    'n_states': 2,
    'QDmodel': 'createQDmodel',
    'QDmodelType': 'OSTL',
    'prepInput': 'True',
    'XfileIn': 'x_data',
    'YfileIn': 'y_data',
    'QDmodelOut': 'OSTL_CNN_SB_model',
    'dataPath': '/path/to/training_trajectories',
    'max_evals': 50,
    'TrEpochs': 1000,
    'OptEpochs': 100,
    'patience': 100,
    'hyperParam': 'True',
    'systemType': 'SB',
    'tempNorm': 1.0,
    'lambNorm': 1.0,
    'gammaNorm': 10.0,
    'DeltaNorm': 1.0,
    'energyNorm': 1.0,
    'YfileIn': 'y_data',
}
quant_dyn(**param)

Listing 1. An example of MLQD input for creating (training) a CNN model following OSTL approach.

B. Case study 2: Excitation energy transfer in FMO complex with AIQD approach

In our second case study, we consider excitation energy transfer in the FMO complex which is described by the Frenkel exciton model with the following Hamiltonian

$$
H = \sum_{n=1}^{N} |n\rangle \epsilon_n \langle n| + \sum_{n,m=1, n\neq m}^{N} |n\rangle J_{nm} \langle m| + \sum_{n=1}^{N} \sum_{k=1}^{N} \left( \frac{1}{2} P_{k,n}^2 + \frac{1}{2} \omega_{k,n}^2 Q_{k,n}^2 \right) - \sum_{n=1}^{N} \sum_{k=1}^{N} |n\rangle c_{k,n} Q_{k,n} \langle n| + \sum_{n=1}^{N} |n\rangle \lambda_{n} \langle n|,
$$

(10)

where $N$ is the number of sites (bacteriochlorophyll molecules), $\epsilon_n$ is the energy of the $n$th site and $J_{nm}$ denotes the inter-site coupling between sites $n$ and $m$. The third term in Eq. (10) describes the environmental part with $P_{k,n}$ as conjugate momentum, $Q_{k,n}$ as coordinate and $\omega_{k,n}$ as a frequency of the corresponding environment mode $k$. $\lambda_{n}$ is the reorganization energy associated with site $n$ and the strength of the coupling between environment mode $k$ and site $n$ is represented by $c_{k,n}$. In the case of FMO complex, we assume that all sites have the same spectral density as described by Eq. (9).

In our example, we consider the 8-site FMO complex where three sites (1, 6 and 8) have an equal probability of getting initially excited while the reaction center is in the vicinity of sites 3 and 4. For the training, we use the FMO-IV data set from the QD3SET-1 database generated for the following Hamiltonian

$$
H_s = \begin{pmatrix}
310 & -80.3 & 3.5 & -4.0 & 4.5 & -10.2 & -4.9 & 21.0 \\
-80.3 & 230 & 23.5 & 6.7 & 0.5 & 7.5 & 1.5 & 3.3 \\
3.5 & 23.5 & 0 & -49.8 & -1.5 & -6.5 & 1.2 & 0.7 \\
-4.0 & 6.7 & -49.8 & 180 & 63.4 & -13.3 & -42.2 & -1.2 \\
4.5 & 0.5 & -1.5 & 63.4 & 450 & 55.8 & 4.7 & 2.8 \\
-10.2 & 7.5 & -6.5 & -13.3 & 55.8 & 320 & 33.0 & -7.3 \\
-4.9 & 1.5 & 1.2 & -42.2 & 4.7 & 33.0 & 270 & -8.7 \\
21.0 & 3.3 & 0.7 & -1.2 & 2.8 & -7.3 & -8.7 & 505
\end{pmatrix}
$$

(11)
FIG. 6. The time evolution of the diagonal (population) and off-diagonal terms (coherence) of the reduced density matrix (RDM) $\tilde{\rho}$, predicted with the recursive (KRR) and non-recursive (AIQD and OSTL) approaches. The calligraphic $R$ and $I$ denote the real and imaginary parts of the off-diagonal terms, respectively. Results are shown for the symmetric spin-boson model ($\varepsilon = 0.0$) with the following set of unseen parameters: $\tilde{\gamma} = 10.0$, $\tilde{\lambda} = 0.3$, $\tilde{\beta} = 1.0$. The predicted results are compared to the reference HEOM method (dots).

with the diagonal offset of 12195 cm$^{-1}$. In the considered data set, exciton dynamics is propagated for the most distant 500 combinations of the following parameters: $\lambda = \{10, 40, 70, \ldots, 520\}$ cm$^{-1}$, $\gamma = \{25, 50, 75, \ldots, 500\}$ cm$^{-1}$, and $T = \{30, 50, 70, \ldots, 510\}$ K. The chosen 500 trajectories are propagated for each possible case of initial excitation (i.e., on sites 1, 6 and 8) with time length $t = 50$ ps and time step $dt = 5$ fs. Calculations are performed with the local thermalizing Lindblad master equation (LTLME) approach,$^{127,128}$ implemented in the QUANTUM_HEOM package.$^{129}$ In order to make it compatible with the Hamiltonians with larger dimensions, the package is modified locally.

We choose our training trajectories (400 for each case of initial excitation) and test trajectories (100 for each case of initial excitation) based on farthest-point sampling. In this case study, we consider only the AIQD and OSTL approaches as training the KRR model (in the current implementation) is not feasible because of its high computational cost. In both approaches (AIQD and OSTL), we use $\lambda_{\text{max}} = 520$, $\gamma_{\text{max}} = 500$ and $T_{\text{max}} = 510$ as normalizing factors for the corresponding simulation parameters (i.e., $\lambda$, $\gamma$ and $T$). In AIQD, we use the same number of logistic functions and the same constants as was adopted for the spin-boson case. After training, we pass a set of unseen simulation parameters and MLQD predicts the corresponding dynamics using the trained CNN models. In the described order, Figs 7 and 8 show the excitation energy transfer (diagonal terms of RDM) and the time evolution of coherent terms (off-diagonal terms of RDM) for a test trajectory. The presented results are with the initial excitation on site 1 and results for the initial excitation on sites 6 and 8 are given in Figs. S2-5. In List. 2, we show an
FIG. 7. The time evolution of the diagonal elements of RDM $\tilde{\rho}$s. The initial excitation is on site 1 and other parameters are $\gamma = 125$, $\lambda = 70$, $T = 30$. The results are compared to the reference LTLME method (dots). In our calculations, $\gamma$ and $\lambda$ are considered in the units of cm$^{-1}$, while $T$ is in the units of K.

Example of the MLQD input for predicting exciton dynamics in the FMO complex using the OSTL approach.

```python
from mlqd.evolution import quant_dyn
param={
    'initState': 8,
    'n_states': 8,
    'time': 50,
    'time_step': 5,
    'QDmodel': 'useQDmodel',
    'QDmodelType': 'OSTL',
    'gamma': 200.0,
    'lamb': 130.0,
    'temp': 330.0,
    'systemType': 'FMO',
    'QDmodelIn': 'OSTL_CNN_FMO_model.hdf5'
}```
FIG. 8. The time evolution of the dominant off-diagonal terms (i.e., $\tilde{\rho}_{mn}, m \neq n$) with the initial excitation on site-1. The calligraphic $R$ and $I$ represent the real and imaginary parts, respectively. The results are compared to the reference LTLME method (dots). The time evolution of diagonal terms and the corresponding simulation parameters are given in Fig. 7.

Listing 2. An example of MLQD input for predicting exciton dynamics in FMO complex using OSTL approach.

IV. CONCLUSIONS AND OUTLOOK

In this article, we have presented MLQD, an open-source Python package for ML-based quantum dissipative dynamics. The package provides a set of recursive (based on KRR method) and non-recursive (AIQD and OSTL) ML approaches with the features of training an ML model, using the trained model to predict dynamics, optimization of hyperparameters and visualization of results. The package has been made available on the XACS cloud computing platform with the interface to the MLATOM package.

To highlight the applications of the MLQD package, we demonstrated the features of the MLQD package on examples of the
2-state spin-boson model and 8-site FMO complex, but it is not restricted to them and can be used for any system if the training data is provided. A comparison of all the different methods implemented in the MLQD package on different systems would be of interest and we are currently working on it. In the future, MLQD will be extended to more realistic systems along with the incorporation of other machine learning-based approaches as well as the traditional quantum dynamics methods such as HEOM and SEOM.

CODE AVAILABILITY STATEMENT

MLQD package is available on https://github.com/Arif-PhyChem/MLQD along with tutorials in Jupyter Notebooks. It can also be installed as pip package, i.e., pip install mlqd. In addition, MLQD is interfaced with the MLATOM@XACS package http://mlatom.com which allows MLQD to be used on the XACS cloud computing platform https://xacs.xmu.edu.cn. A user manual for MLQD on cloud computing is provided at http://mlatom.com/manual/#mlqd.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ACKNOWLEDGMENTS

We acknowledge funding by the National Natural Science Foundation of China (No. 22003051 and funding via the Outstanding Youth Scholars (Overseas, 2021) project), the Fundamental Research Funds for the Central Universities (No. 20720210092), and via the Lab project of the State Key Laboratory of Physical Chemistry of Solid Surfaces. Calculations were performed with high-performance computing resources provided by Xiamen University.

REFERENCES

3V. May and O. Kuhn, Charge and energy transfer dynamics in molecular systems (John Wiley & Sons, 2008).


