Excited-state dynamics with machine learning

Lina Zhang, Arif Ullah, Max Pinheiro Jr, Pavlo Dral, Mario Barbatti

Excited-state dynamics with machine learning

Lina Zhang,[a] Arif Ullah,[a] Max Pinheiro Jr,[b] Pavlo O. Dral,[a*] Mario Barbatti[b,c*]

[a]*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*

[b]*Aix Marseille University, CNRS, ICR, Marseille, France*

[c] *Institut Universitaire de France, 75231 Paris, France*

E-Mail: dral@xmu.edu.cn

E-mail: mario.barbatti@univ-amu.fr

Abstract

This chapter will briefly introduce how machine learning can assist excited-state dynamics. We start with an overview of two traditional excited-state dynamics, a two-state system-bath approach in full quantum dynamics and the trajectory surface hopping approach in mixed quantum-classical dynamics. Then, we introduce a combination of machine learning with each of these two approaches. Finally, we present case studies in a tutorial format, enabling readers to perform simple dynamics simulations on model systems and realistic molecules.

Introduction

Electronically excited states are all around in the photophysics and photochemistry of molecules and materials [1-3]. Theoretical studies of excited states are vital in many different research fields, including photocatalysis [4-6], photosynthesis [7-9], photolysis [10-11], photoprotection [12-14], photoreception [15-17], and photovoltaics [18-20]. These studies may be performed based either on static electronic structure calculations or dynamics, or a combination of both [1]. Dynamics research elucidates temporal evolution processes of molecules using molecular dynamics (MD) approaches. MD encompasses a vast family of methods from pure classical dynamics [21-25] (using either single Born–Oppenheimer potential energy surfaces or force fields) to full quantum dynamics [26-30], with many approaches in between (see Chapter *Basics of dynamics*). In order to reveal the temporal evolution of molecules after light absorption and thus provide information about important

geometries visited, decay channels, branching ratios, and lifetimes of excited states, we need to use excited-state dynamics approaches, which are usually referred to as nonadiabatic MD (NAMD) because dynamics may need to account for nonadiabatic effects occurring at nuclear conformations where the Born–Oppenheimer approximation fails [1,31].

A large number of NAMD methods have been developed. Ideally, when performing NAMD, we want to solve the full time-dependent quantum mechanical problem with no approximations, i.e., to carry out *full quantum dynamics* for either an isolated or an open system interacting with its environment. In the past decades, many full quantum dynamics methods were developed, such as the multiconfiguration time-dependent Hartree (MCTDH) [26], time-dependent DMRG (TD-DMRG) [32], hierarchical equations of motion (HEOM) [30], and many others [27,33-40]. Though full quantum dynamics methods are exact, their computational cost and complexity restrict them to only model systems. To overcome these challenges, mixed quantum-classical methods are developed, such as the Ehrenfest dynamics [41] and the popular trajectory surface hopping (TSH) [42], which propagates classical nuclear trajectories and correct them for nonadiabatic effects with *ad hoc* approximations. Further approximations such as quasiclassical dynamics methods, like the Miller–Meyer–Stock–Thoss formalism (MMST) [43], ab initio multiple spawning (AIMS) [44], and others [45-46] are built by truncating the full quantum problem in the lowest powers of the Planck constant or mapping the quantum coordinates onto the classical phase space. Some of the imaginary-time path-integral-based approaches, such as ring-polymer molecular dynamics (RPMD) [47] and centroid molecular dynamics (CMD) [48], are also worth noting.

Whenever a NAMD method is based on classical trajectories, as in AIMS and TSH, we say it is mixed quantum-classical dynamics (MQC) [49]. This class of methods holds most of the applied investigations of realistic molecules, ranging from small isolated molecules [50-53] to nanoscopic clusters [54-56]. However, the expensive quantum chemical calculations needed for these simulations are still a major limiting step for making these methods a routine practice in quantum chemistry. Take TSH as an example; Westermayr et al. [3] estimated that propagating 100 fs dynamics for a tiny molecule like $CH_2NH_2^+$ using highly accurate quantum-chemical properties (computed with MR-CISD/aug-cc-pVDZ) would take about 20 h using one CPU (2x Intel Xeon E5-2650 v3 CPU). It may not sound much, but considering that to reach statistical convergence of 2% error in the sample proportions (this margin of error for a 95% confidence interval is approximately $1/\sqrt{N}$, where N is the number of trajectories), we

may need 2000 trajectories or 4.5 years of CPU time. Consider yet that even with some of the most advanced MRCI algorithms, the calculations still scale with $N^7 \times M$ [57], where N is the number of active orbitals and M the number of orbitals, and you can grasp the prohibitive costs of performing dynamics. TSH is a local theory where quantum effects are estimated only at the classical position. If, however, non-local quantum effects are required (for instance, for the description of tunnelling or quantum interference), we must use higher-level dynamics methods, like the full quantum dynamics methods mentioned above. Naturally, such non-local quantum methods using information from the entire configuration space are computationally expensive and not easily extensible to large systems. For instance, the computational cost and memory requirements of HEOM, which delivers non-local quantum information for a system-bath model Hamiltonian, increase exponentially with the decrease in temperature.

Fortunately, machine learning (ML) has proven to make relatively accurate predictions for chemical properties at a meagre cost [58-61], thus, is seen as an auspicious way to accelerate NAMD. The ML modelling of potential energy surfaces (PESs), couplings, and other quantities we may need for dynamics is still very challenging. Take ML-assisted TSH as an example. First, great care is required when generating the training set to include as many important configurations as possible while keeping the training set at a minimum. Moreover, extra attention should be paid when picking points around critical regions like conical intersections since the PES will show removable discontinuities, and nonadiabatic coupling vectors (NACs) may show a singularity around these regions.

Despite these difficulties, much progress has been made in ML-assisted NAMD. Westermayr et al. [3] showed that it would just cost 24 s to simulate 100 fs surface hopping dynamics for the same $CH_2NH_2^+$ through SchNarc (an approach combining SchNet and SHARC) using one CPU (2x Intel Xeon E5-2650 v3 CPU). Lopez et al. [62] have performed a 10 ns simulation for trans-hexafluoro-2-butene in 2 days, using the machine learning *ab initio* molecular dynamic package PyRAI2MD. Akimov [63] has recently applied ML to propagate nanostructures dynamics for 20 ps. Table 1 surveys several of such works. It is, of course, a dated and incomplete picture of the field; however, it may help debutant researchers find out the type of method that is the most suited for their application.

While most applications of ML-assisted NAMD have been made with TSH, significant progress has also been made in ML-assisted system-bath quantum dynamics. Rodríguez and Kananenka [64] have studied the excitation energy transfer in a dimer using a convolutional

neural network (CNN). Lin et al. [65] have used bootstrap-based long short-term memory recurrent neural networks to predict the quantum dissipative dynamics of a two-state system. Ullah and Dral [66] have studied the dynamics of the well-known spin-boson model based on the non-parametric kernel ridge regression (KRR) model. In all these studies, a reference short-time trajectory is provided as an input, and the trained ML model predicts the long-time quantum dynamics iteratively with very low computational cost. In a recent preprint, Ullah and Dral used ML to propagate quantum dynamics up to the asymptotic limit of infinite time [67] non-iteratively without using a reference short-time trajectory as input.

This chapter delivers a practical discussion of major concepts and issues from traditional excited-state dynamics to machine learning-assisted dynamics and shows how to perform such simulations.

Table 1. ML-NAMD within TSH for molecular systems reported in the literature (stand on June 2021). See explanation about ZNSH and DC-FSSH in *Methods*.

No.	Ref.	System (number of atoms/dimensions)	ML algorithm descriptor single- or multi-state model	Number of QC calculations	Hopping
1	Lan et al., JPCL, 2018[68]	6-aminopyrimidine (12 atoms)	KRR with Gaussian kernel Coulomb matrix single-state	Total: 201 008; Training: 65 316; Dynamics: CASSCF calculation accounts for 3.9%	ZNSH
2	Dral, Barbatti, Thiel, JPCL, 2018[69]	Spin-boson Hamiltonian (1 and 33 dimensions)	KRR with Matérn kernel coordinates single-state	Training: 128 for 1-D A-SBH model; 1,000 & 10,000 for 33-D A-SBH model Dynamics: A-SBH calculations accounts for 13–16%	DC-FSSH
3	Cui et al., JPCL, 2018[70]	CH_2NH (5 atoms)	NN-based DPMD model local frame descriptor single-state	Training: 90 000 Test: 26 522	ZNSH
4	Cui et al.,JPCL, 2019[71]	$CH_3N{=}NCH_3$ and five waters (15 atoms for waters)	NN-based DPMD model local frame descriptor chromophore is not treated by ML models, ML only used for water monomer, dimer, and trimer energies	Training: 144,000 for monomer, 100,000 for dimer, and 60,000 for trimer	ZNSH
5	Marquetand et al., CS, 2019[72]	$CH_2NH_2^+$ (6 atoms)	feed-forward NNs inverse distance descriptor multi-state	Final training set: 4000; Test set: 770	SHARC
6	Marquetand et al., MLST, 2020[73]	$CH_2NH_2^+$ (6 atoms)	①feed-forward NNs FCHL	Training set: 4000 Test set: 770	SHARC

			inverse distance descriptor single-state and multi-state ②KRR FCHL single-state and multi-state		
7	Marquetand et al., JPCL, 2020[74]	$CH_2NH_2^+$ and CSH_2 (6 and 4 atoms)	NN-based SchNet model automatically determined representation derived from interatomic distances and nuclear charge multi-state	Training set: 3000 for $CH_2NH_2^+$; 4703 for CSH_2	SHARC
8	Brorsen et al., JPCA, 2020[75]	Spin-boson Hamiltonian system (1 and 3 dimensions)	KRR with Matérn kernel Coordinates ML used only to fit the nonadiabatic coupling single-state	Training: 17 points for 1D A-SBH model; 4913 points for 3D A-SBH model	FSSH
9	Lopez et al., CS, 2021[62]	Trans-hexafluoro-2-butene; Norbornyl hexacyclodiene (12 atoms; 25 atoms)	feed-forward NN inverse distance descriptor multi-state	Total: Trans-hexafluoro-2-butene 6207; Norbornyl hexacyclodiene 6267	FSSH; ZNSH

Methods

This section outlines traditional and ML-assisted NAMD methods with a focus on HEOM and TSH, which will be later considered in the case studies.

Hierarchical equation of motion (HEOM)

In full quantum dynamics methods, nuclei and electrons' time evolutions are described quantum mechanically. In the Chapter *Basics of dynamics*, the time-dependent Schrödinger equation has been expanded in a Born–Huang basis, leading to electrons-nuclei coupled equations of motion. Here, we describe the system-bath approach, where the system is divided into two parts: electrons are considered the system of interest, which are coupled to the nuclei, treated as an environment (bath); note that other definitions of system and bath are possible. Many methods are based on a system-bath approach, e.g., QuAPI [27], HEOM [30], and the stochastic equation of motion (SEOM) [33-35], and in this chapter, we restrict ourselves to HEOM and a two-state system with one ground state and one excited state, which is also coupled to external (nuclear) degrees of freedom (bath). The bath is modeled as a reservoir of an infinite number of noninteracting harmonic oscillators. To describe our composite, total system, we use the spin-boson model (we set $\hbar = 1$ and $\beta = 1/(k_B T)$ where k_B is Boltzmann constant and T is the temperature):

$$\hat{H}_{\mathrm{T}} = \hat{H}_{\mathrm{S}} + \hat{H}_{\mathrm{B}} + \hat{H}_{\mathrm{SB}},$$
$$\hat{H}_{\mathrm{T}} = \frac{1}{2}\varepsilon\hat{\sigma}_z + \frac{1}{2}\Delta\hat{\sigma}_x + \sum_k \omega_k \hat{b}_k^{\dagger}\hat{b}_k + \hat{\sigma}_z \sum_k c_k\left(\hat{b}_k^{\dagger} + \hat{b}_k\right) \quad (1)$$

where $\hat{H}_{\mathrm{S}}$, $\hat{H}_{\mathrm{B}}$, and $\hat{H}_{\mathrm{SB}}$ are respectively system Hamiltonian (S), bath Hamiltonian (B), and system-bath interaction Hamiltonian term (SB). $\hat{\sigma}_x$ and $\hat{\sigma}_z$ are Pauli matrices, ε is the energy bias of the two states, and Δ is the tunneling matrix element of the two states. $\hat{b}_k^{\dagger}\left(\hat{b}_k\right)$ is the creation (annihilation) operator and ω_k is the frequency of the bath mode k. In Eq. (1), the last term denotes the system-bath interaction where c_k represents the interaction coefficients. The influence of bath on the dynamics of the two-state system is described by a two-time correlation function of the operator $\hat{F}$, i.e., $C(t) = \langle\hat{F}(t)\hat{F}(0)\rangle$ where $\hat{F} = \sum_k c_k\left(\hat{b}_k^{\dagger} + \hat{b}_k\right)$. In the case of isolated bath, $C(t)$ can be written as [76]

$$C(t) = \frac{1}{\pi}\int_0^{\infty} \mathrm{d}\,\omega J(\omega)\left(\coth\left(\frac{\omega\beta}{2}\right)\cos(\omega t) - i\sin(\omega t)\right) \quad (2)$$

where $J(\omega)$ is the spectral density $J(\omega) = \sum_k c_k^2\delta(\omega - \omega_k)$, and here we consider it in Debye form

$$J(\omega) = 2\lambda\frac{\omega\omega_{\mathrm{c}}}{\omega^2 + \omega_{\mathrm{c}}^2} \quad (3)$$

where $\omega_{\mathrm{c}}$and λ denote the characteristic frequency and the reorganization energy, respectively. The correlation function in Eq. (2) can be expressed as a sum of exponential decay functions of time

$$C(t > 0) = \sum_{k=0}^{\infty}\eta_k e^{-\gamma_k t} \quad (4)$$

with Matsubara frequencies $\gamma_0 = \omega_c$ and $\gamma_k = 2k\pi/\beta$ for $k \geq 1$. The coefficients η_k are given by

$$\eta_0 = \lambda\omega_c\left[cot\left(\frac{\omega_c\beta}{2}\right) - i\right] \quad (5)$$

$$\eta_k = \frac{4\lambda\omega_c}{\beta}\frac{\gamma_k}{\gamma_k^2 - \omega_c^2} \quad \text{for } k \geq 1 \quad (6)$$

We assume that the system is initially in the excited state, and the bath is in thermal equilibrium. We have a product initial state $\rho_{\mathrm{T}}(0) = \rho_{\mathrm{S}}(0) \otimes e^{-\beta \widehat{H}_{\mathrm{B}}}/Z_{\mathrm{B}}$ where ρ_{T}, ρ_{S} and Z_{B} are respectively the density matrix of the composite system, the density matrix of the system of interest, and the partition function of the isolated bath $Z_{\mathrm{B}} = \mathrm{Tr}[e^{-\beta \widehat{H}_{\mathrm{B}}}]$ (Tr denotes the trace over bath degrees of freedom). This condition is justified in situations when the bath relaxation is slower compared to the changes in the system after the initial excitation (methods to deal with the situation when this approximation is no longer valid also exist [77-78]). Using the path integral formalism, the HEOM for Eq. (1) can be written as [30]

$$\frac{\partial}{\partial t}\rho_{\boldsymbol{n}} = -\left(i\mathcal{L} + \sum_k n_k \gamma_k\right)\rho_{\boldsymbol{n}} + -i\left[\hat{\sigma}_z, \sum_k \rho_{n_k^+}\right] - i\sum_k n_k\left(\eta_k \hat{\sigma}_z \rho_{\boldsymbol{n}_k^-} - \eta_k^* \rho_{\boldsymbol{n}_k^-}\hat{\sigma}_z\right) \quad (7)$$

where $\mathcal{L} * = [\widehat{H}_{\mathrm{S}}, *]$ represents the Liouville superoperator. The labels $\boldsymbol{n} = \{n_{\mathrm{k}}\} = \{n_0, n_1, n_2, \ldots\}$ are integers, and for each Matsubara term k, each index runs from zero to infinity. The null labels $\rho_{[0,0,0,\ldots]}$ define the reduced density matrix (RDM) of the system (density matrix of the system of interest after tracing out the bath degrees of freedom). The $\rho_{\boldsymbol{n}}$ with $\boldsymbol{n} \neq 0$ represent the auxiliary density matrices (ADMs) which encode the influence of the bath on the system. The $\boldsymbol{n}_k^{\pm}$ refers to the change in $\boldsymbol{n}$ at index k by 1, i.e., $\boldsymbol{n}_k^{\pm} = \{n_0, n_1, \ldots, n_k \pm 1, \ldots\}$. In Eq. (4), as $k \to \infty$, $\gamma_k \to \infty$ and $\eta_k \to 0$, thus we can introduce a cutoff K, neglecting the terms with large Matsubara frequency ($k > K$) or treating them using the Markov approximation (replaced them by Dirac's delta function $\delta(t)$). At high temperatures, we may need a small cutoff K, but it should be increased with a decrease in temperature. Each matrix in Eq. (7) belongs to a hierarchy level L defined as $L = \sum_{k=1}^{K} n_k$. At $L = 0$ hierarchy level, we only evaluate RDM ignoring the effects of the bath. All ADMs in Eq. (7) occupy higher hierarchy levels such as ADMs with $\boldsymbol{n} = \{1,0,0,0\}$ and $\boldsymbol{n} = \{1,0,0,1\}$ belong to $L = 1$ and $L = 2$, respectively. As evaluating an infinite number of ADMs is not feasible, a hierarchy truncation level should be fixed, setting a limit on L. The hierarchy truncation level should be increased for non-Markovian cases (strong system-bath coupling) as the contribution of higher-level ADMs becomes significant. With given K and L, the total number of ADMs is given by $(L + K)!/(L!\,K!)$. For a two-state system with $K = 2$ and $L = 30$, 496 matrices (each of size 2×2) need to be evaluated at each time-step. In the HEOM method, the number of ADMs that have to be evaluated rapidly increases with an increase in K and L, which may therefore lead to a very high computational cost.

ML-assisted HEOM

ML can substantially reduce the propagation cost with the expensive full quantum methods such as HEOM. A strategy proposed recently [64,66] trains ML on multiple trajectories and then propagates new trajectories for different conditions. Before training ML models, unsupervised training trajectories (unlabeled data with no explicit vector **Y**, see Figure 1) are transformed into supervised data (labeled data with explicit vector **Y**). For this, an input for ML is formed from a short time segment with time-length t_m (memory time) of a trajectory (e.g., with $\langle\hat{\sigma}_z\rangle$ values taken with time-step dt_{train}), and the next value (e.g., $\langle\hat{\sigma}_z\rangle$) at t_{m+1} can be used as the target value to learn (label). Thus, we can generate much supervised data even from a single trajectory (see Figure 1). To propagate a new trajectory, ML should be given a short time trajectory of time length t_m and the trained ML model predicts dynamics a learned property for the next time step t_{m+1} (see Figure 1). This process is repeated iteratively as the predicted value for the previous time-step is included in the input vector for the ML model to predict the next time step. Figure 2 shows the time-dependent dynamics of $\langle\hat{\sigma}_z\rangle$ for a two-state system. It is worth emphasizing that the trained ML model is also capable of extrapolation beyond the maximum propagation time of the training trajectories (see Figure 1).

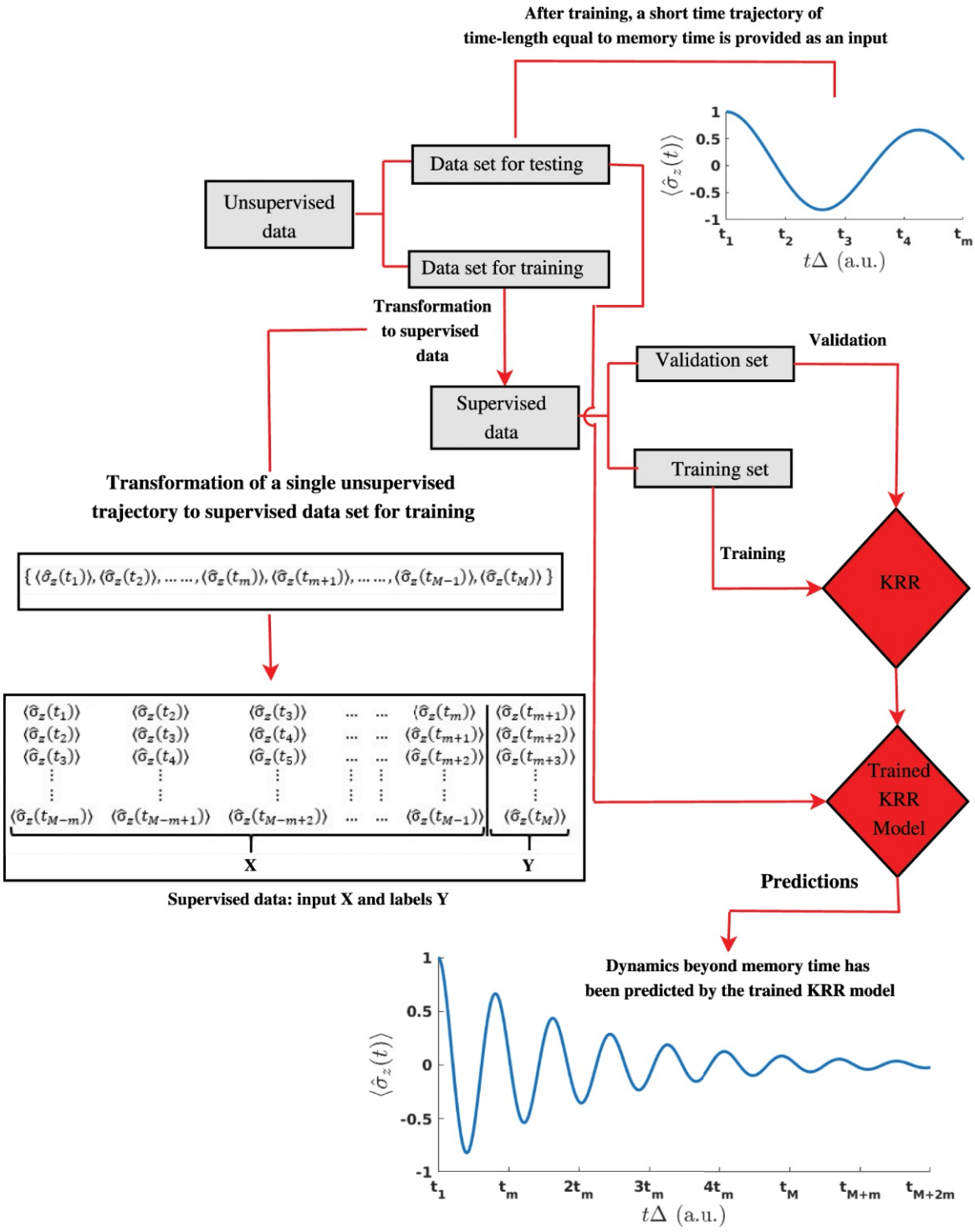

Figure 1. Flowchart of generating supervised training data from unsupervised training trajectories, training ML model, and making predictions with it to speed up NAMD simulations within HEOM approach. $\langle \hat{\sigma}_z(t) \rangle$ is given in Eq. (17). See Figure 2 for results.

Credit: Reproduced from Ref. [66] (New J. Phys. 2021, 23, 113019) under CC-BY 4.0.

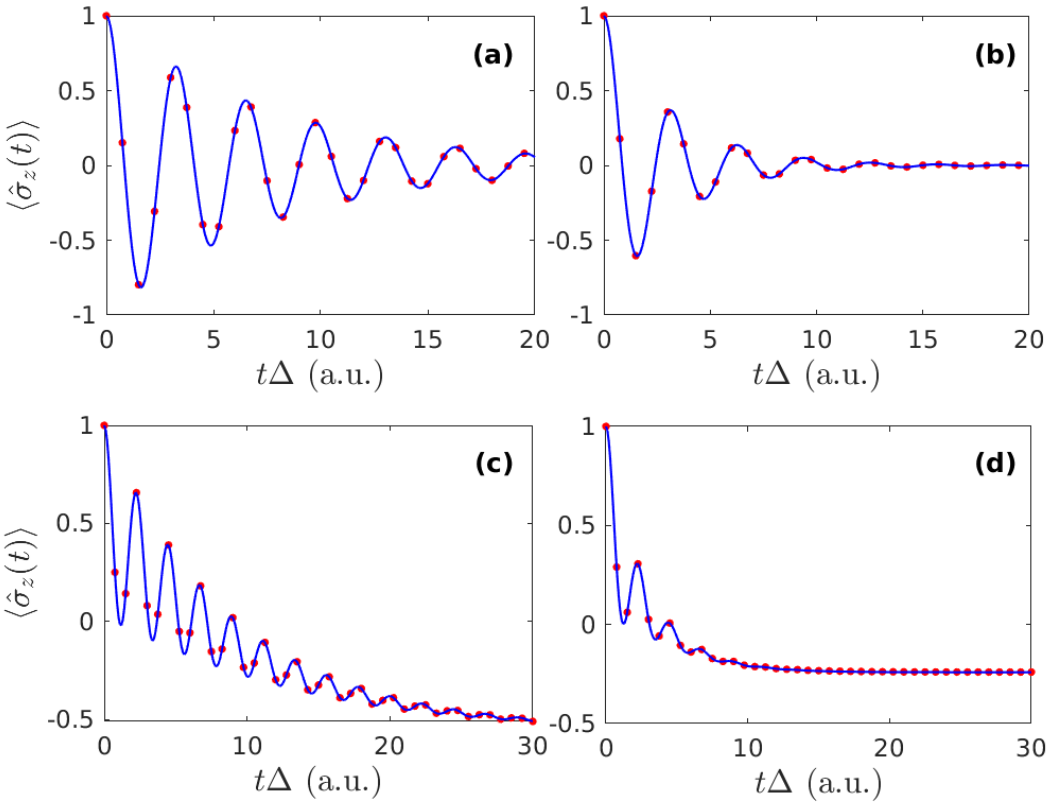

Figure 2. Expectation value of $\hat{\sigma}_z$ for two state system as a function of time. Results predicted by KRR model (blue line) are compared to the HEOM results (red dots). The adopted parameters are: (a) $\epsilon = 0.0,\ \lambda = 0.2,\ \omega_c = 8.0,\ \beta = 1.0$; (b) $\epsilon = 0.0,\ \lambda = 0.2,\ \omega_c = 10.0,\ \beta = 0.25$; (c) $\epsilon = 1.0,\ \lambda = 0.1,\ \omega_c = 6.0,\ \beta = 0.75$ and (d) $\epsilon = 1.0,\ \lambda = 0.2,\ \omega_c = 10.0,\ \beta = 0.25$. Here in all cases, $\Delta = 1.0$. All parameters are in units of Δ.

In the literature, both neural networks and kernel methods were used for this strategy of accelerating HEOM dynamics with ML. In Case study 2, we use a kernel method (kernel ridge regression, KRR, with the Gaussian kernel). We note that other strategies were also suggested [66-67,79]: one of them is to learn from a single short time trajectory and to propagate the remaining trajectory for a longer time, while in the other, ML was used to learn the entire trajectories as a function of continuous time, but we will not elaborate on these strategies here.

Trajectory surface hopping (TSH)

In mixed quantum-classical (MQC) dynamics, the degrees of freedom of the entire system are split into slow particle parts (usually, nuclei), which are treated classically, and fast particle parts (usually, electrons) treated quantum mechanically. In doing so, MQC methods circumvent extensive calculations usually performed when full quantum dynamics is applied.

Trajectory surface hopping (TSH) is one of the most well-known types of MQC methods. TSH aims to emulate nuclear wavepacket propagation moving through a nonadiabatic coupling region through an ensemble of independent classical trajectories. The wavepacket split due to nonadiabatic effects is simulated by allowing each trajectory to change the potential energy surface during the time propagation following a stochastic algorithm. Thus, TSH is composed of two core elements, the classical dynamics on Born–Oppenheimer surfaces and the hopping probabilities calculation.

TSH algorithms can use either instantaneous or global probability algorithms. Instantaneous algorithms tell the probability of making a transition between two electronic states at a specific time. The fewest-switches surface hopping (FSSH) introduced by Tully [42] is the most popular among the instantaneous algorithms. Global algorithms evaluate the transition probability over the entire duration of the nonadiabatic interaction and evaluate afterward whether the system should move to another surface or not. The Zhu–Nakamura surface hopping (ZNSH) is an example of such an approach [80].

Each type of algorithm has its pros and cons. FSSH is derived for arbitrary surface crossing topographies, is straightforward to implement, and is easy to generalize to other cases beyond internal conversion [81-82]. It, however, requires the computation of nonadiabatic couplings and needs to be corrected for decoherence effects [83]. ZNSH, in turn, is derived for Landau–Zener-like topographies (the adiabatic surfaces approach each other, creating a region of nonadiabatic interaction, and separate again) and requires a more involved implementation (when a hopping takes place, the trajectory must resume from an earlier time step). Nevertheless, ZNSH neither requires nonadiabatic couplings (estimated from the diabatic forces) nor decoherence corrections.

As surveyed in Table 1, both FSSH and ZNSH have been used with ML. In our Case Study 3, we will use FSSH. For this reason, let us take a moment to understand some technical details of this method. For a discussion of the ZNSH implementation, see Ref. [84].

Fewest switches surface hopping (FSSH)

If we want to compute a single trajectory for FSSH, we need to solve two equations of motion (EOM), a classical equation for the nuclei $\overline{\mathbf{R}}$ on a Born–Oppenheimer potential energy surface H_{LL}

$$\frac{\mathrm{d}^2\overline{\mathbf{R}}_{\boldsymbol{\alpha}}}{dt^2} = -\frac{1}{M_\alpha}\nabla_\alpha H_{LL} \tag{8}$$

where L denotes the Lth electronic state and α represents the nucleus α; and a quantum EOM [42]

$$\frac{\mathrm{d}c_J}{\mathrm{d}t} = \sum_K -c_K\left(\frac{i}{\hbar}H_{JK} + \sigma_{JK}\right) \tag{9}$$

This equation is a local approximation of the time-dependent Schrödinger equation for the electronic wave function. c are the complex-valued coefficients we are interested in. H_{JK} is the off-diagonal element of the electronic Hamiltonian matrix between electronic states J and state K (they are null in the adiabatic representation) and σ_{JK} is the time derivative coupling between these states, defined as

$$\sigma_{JK}(\overline{\mathbf{R}}) \equiv \left\langle \psi_J \middle| \frac{\partial \psi_K}{\partial t} \right\rangle = \mathbf{d}_{JK} \cdot \bar{\mathbf{v}} \tag{10}$$

The last equality shows that in FSSH, we can either use the time derivative coupling or the inner product between the nuclear velocity $\bar{\mathbf{v}}$ and nonadiabatic coupling vectors (NACs)

$$\mathbf{d}_{JK} \equiv \langle \psi_J | \nabla | \psi_K \rangle \tag{11}$$

After solving the two EOMs for one time step, we have updated electronic coefficients $\mathbf{c}$ and nuclear positions $\overline{\mathbf{R}}$ and velocities $\bar{\mathbf{v}}$. In adiabatic representation (which is commonly used in FSSH), the FSSH transition probability from state K to state J state is simply

$$P_{K\to J}^{FSSH} = \max\left[0, \frac{2\Delta t}{\rho_{KK}}\mathrm{Re}\left(\rho_{KJ}\sigma_{KJ}\right)\right] \tag{12}$$

where are the density matrix elements

$$\rho_{KJ}(t) \equiv {c_K}^*(t)c_J(t) \tag{13}$$

The decision on whether the trajectory will switch from K to J is made by checking two conditions. First, we sample a uniform random number ξ between 0 and 1. The switch may occur if

$$\sum_{I}^{J-1} P_{K\to I}^{FSSH} < \xi \leq \sum_{I}^{J} P_{K\to I}^{FSSH} \tag{14}$$

Second, the total energy (kinetic plus potential) must remain constant. The hopping is allowed only if adjusting the kinetic energy can balance the potential energy change. If both conditions are satisfied, a hopping from *K* to *J* happens. Before resuming the EOM propagation for the next time step, the nuclear velocities are rescaled to keep the total energy constant (see Ref. [85] for a detailed description of how this is done).

FSSH is over-coherent. In simple terms, it means that the non-diagonal terms of the density matrix in Eq. (13) do not go to zero as fast as they should [83], causing a wrong description of the populations. There are many ways to address this problem. A practical approach is to use the simplified decay of mixing [86], which corrects the coefficient *c* before resuming the integration of the EOMs to the next time step. Thus, most recent works do not employ the original FSSH but some flavor of decoherence-corrected (DC) FSSH.

FSSH can be generalized to work not only during internal conversion but also intersystem crossing [81,87]. In the latter case, instead of the NAC (Eq. (11)) we should use the spin-orbit coupling (SOC). Moreover, the dynamics should not be propagated in the adiabatic representation, but in the spin-adiabatic representation (more formally, the PESs are eigenvalues of the total electronic Hamiltonian including SOC [88]).

Nonadiabatic couplings

FSSH needs nonadiabatic couplings to evaluate the transition probabilities in Eq. (12). As we mentioned, we either work with time derivative couplings σ_{JK} or nonadiabatic coupling vectors $\mathbf{d}_{JK}$. If the electronic structure method used to compute the potential energy surfaces allows getting $\mathbf{d}_{JK}$, it should be used, as the proper treatment of the kinetic energy correction after hopping requires knowing it [85]. However, this quantity is not always available. In such a case, σ_{JK} becomes a good option because it can be readily computed [89-90] from the electronic wavefunction overlap at consecutive time steps using the Hammes-Schiffer/Tully approach [91]. In fact, σ_{JK} can be computed even for methods without explicit wavefunctions, like those based on linear-response theory [92].

σ_{JK} can also be computed with time-dependent Baeck–An (TD-BA) approach [93], a straightforward approximation based only on the energy gap between *J* and *K* and the second time derivative of this gap (which can be quickly evaluated numerically during the propagation):

$$\sigma_{JK} \approx \begin{cases} \frac{sgn\left(\Delta E_{JK}\right)}{2} \sqrt{\frac{1}{\Delta E_{JK}} \frac{d^2 \Delta E_{JK}}{dt^2}} & \text{if } \frac{1}{\Delta E_{JK}} \frac{d^2 \Delta E_{JK}}{dt^2} > 0 \\ 0 & \text{if } \frac{1}{\Delta E_{JK}} \frac{d^2 \Delta E_{JK}}{dt^2} \leq 0 \end{cases} \quad (15)$$

We will use this approximation in Case Study 3.

ML-assisted TSH

TSH requires two critical ingredients: potential energy surfaces and the hopping probabilities. ML can predict both ingredients, which we cover in the following subsections. In addition, we will elaborate on general strategies for the generation of satisfactory training data.

Learning potential energy surfaces

Propagating trajectories requires evaluating potential energies and energy gradients at each time step on either ground or excited state, for which machine learning potential (MLP) can be used. Here, we will mainly focus on the peculiarities of MLP application to NAMD, while a discussion of many different types of MLPs for learning ground-state potential energy can be found in the preceding chapters of this book. While ground-state dynamics is limited to moving on the lowest surface, NAMD faces the increased complexity of the multivalued landscape [94]. The excited-state potential surfaces stack on each other, and even interfaces between them may exist (see Figure 3, from the review paper of Mai and González [1], for a nice illustration of such an excited-state landscape featuring many types of topographic structures).

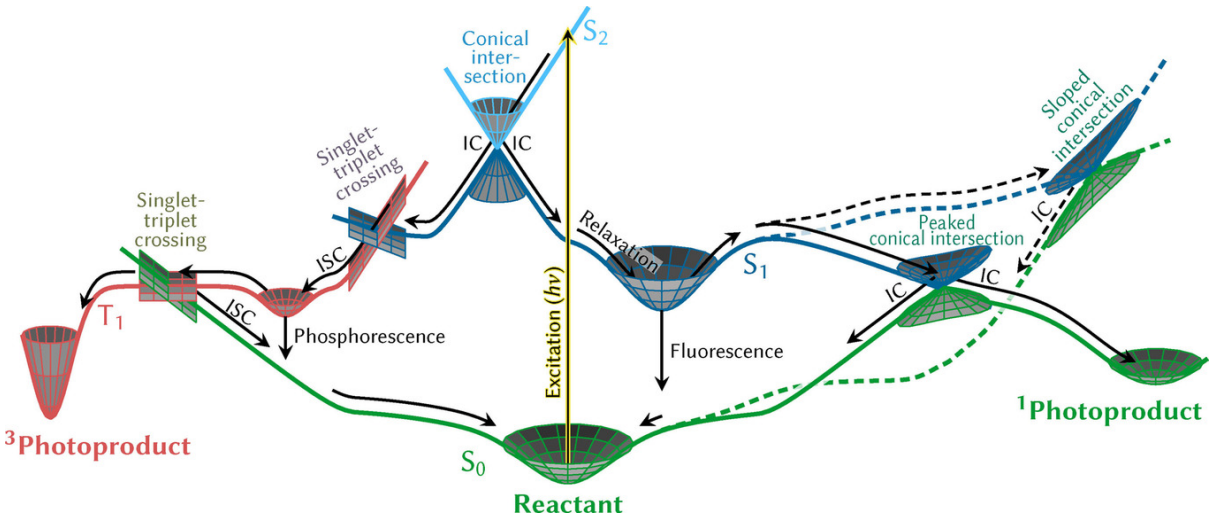

Figure 3. A typical potential energy surface in NAMD.

Credit: Reproduced from Ref. [1] (Angew. Chem. Int. Ed. 2020, 59, 16832–16846) under CC BY 4.0.

Among topographical structures, one particular structure called conical intersection [95-97] is of high interest to us. Conical intersections are crossing regions with a double-cone topography formed when two potential energy surfaces of two states with the same multiplicity touch [1]. Nonadiabatic processes often occur in the vicinity of these features because the Born–Oppenheimer approximation weakens there, and the nonadiabatic couplings between the two states are non-negligible.

Before discussing the ML training of PES for NAMD, we should briefly mention that any quantum-chemical reference data may have accuracy issues. First, the accuracy of the excited state prediction is usually lower than that of the ground state [98]. Second, the accuracy of excited states of different electronic characters may also differ. Therefore, the data we will learn is usually intrinsically inaccurate.

We can now summarize the peculiarities of learning potential energy surfaces for NAMD and possible strategies suggested in the literature to address them:

- First, we need to learn more than one surface in a balanced way. We can either learn each surface separately for each state or learn them together [3,31,98]. The latter option is done by using multi-output learning with neural networks [62,72-74] special extension of kernel methods [73] or simply adding the state number into the input vector [73]. All these approaches have been successfully applied.

- Second, learning excited-state energies tends to be more difficult than learning ground-state energies. Thus, usually, more training data for learning excited-state energies are required to achieve satisfactory accuracy [3,31,98].

- Third, at the seam of conical intersections, the potential energy surface presents removable discontinuities, which poses a challenge to the learning. One suggested solution is switching to reference calculations for PESs when NAMD moves into critical regions near conical intersections to ensure accurate treatment of these regions [3,31,98]. Another solution is to include more geometries near conical intersections in the training set [3,31,98].

Many different MLPs were used for learning both ground- and excited-state potential energy surfaces required for propagating TSH, and a dated list is given in Table 1. Here we briefly describe the MLP – the KREG model – used in Case study 3.

KREG [99] is based on kernel ridge regression (KRR) with RE (internuclear distances r relative to equilibrium distances r^{eq}) descriptor (a vector with elements r/r^{eq} calculated for all pairs of atoms in a molecule) and the Gaussian kernel function. Descriptions about KRR and Gaussian kernel function can be found in Chapter *Kernel methods*.

The RE descriptor is rotationally and translationally invariant, but it lacks permutational invariance concerning the interchange of atoms with the same element. Thus, a permutationally invariant KREG model called pKREG can be used, introducing permutational invariance on a kernel level.

Hopping in ML-assisted TSH: Internal conversion

Excited-state processes of molecules may involve states of the same or different spin multiplicity (Figure 3), which leads to two types of nonadiabatic processes: internal conversion (IC) and intersystem crossing (ISC) [82]. The IC process happens between states with the same multiplicity and is governed by the NACs, while the ISC process occurs between states with different multiplicities and is mediated by spin–orbit couplings (SOCs) [82]. In this subsection, we focus on ML-assisted description of IC, while in the following subsection, we discuss ISC. To describe IC within TSH, we need to choose a hopping algorithm to determine which properties the machine needs to learn.

Two different scenarios were explored in literature: one is the use of an algorithm such as ZNSH or TD-BA FSSH, which do not need the calculation of NACs and can derive hopping

probabilities from the potential energy surfaces, and another one is the use of other FSSH algorithms requiring the calculation of NACs. In the first case, learning potential energy surfaces, as described in the preceding section, is enough to propagate NAMD. This is what we will do in Case study 3. In the second case, we also need to learn NACs, which is a challenging problem. The difficulties and possible solutions to them are as follows:

- NAC data will show singularity around conical intersections, clearly seen after rewriting Eq. (11) according to the Hellmann–Feynman theorem [100]:

$$\mathbf{d}_{JK} = \frac{\langle \psi_J | \nabla \hat{H}_e | \psi_K \rangle}{E_J - E_K} \text{ for } J \neq K \tag{16}$$

  Obviously, when the related electronic states are nearly degenerate, the gap $E_J - E_K$ becomes very small, which leads to extremely large NACs (Figure 4). NACs often are very narrow functions, meaning that even a slight change in geometry can drastically change NAC values. This poses a severe problem for learning them [3] as, on the one hand, the training set may not contain enough (if any at all) points with substantial NAC values and, on the other hand, such a narrow function is intrinsically difficult to learn with ML. Possible solutions are including points with substantial NAC values into the training set [60,69,75] and switching to reference calculations in the region of small gaps, where the probability of large NAC values is larger [69]. An additional simple solution is to multiply values of NACs by the gap $E_J - E_K$ to generate easier to learn values [73-74] (Figure 4). Alternatively, one can perform diabatization (as is often done in full quantum dynamics simulations, and diabatization can also be assisted by ML [3,31,101]), and then ML can be trained on much smoother coupling values, which do not have a singularity problem [102-105]. However, there are few such studies due to the complexity of diabatization and the problems of using diabatic representation in TSH [82].

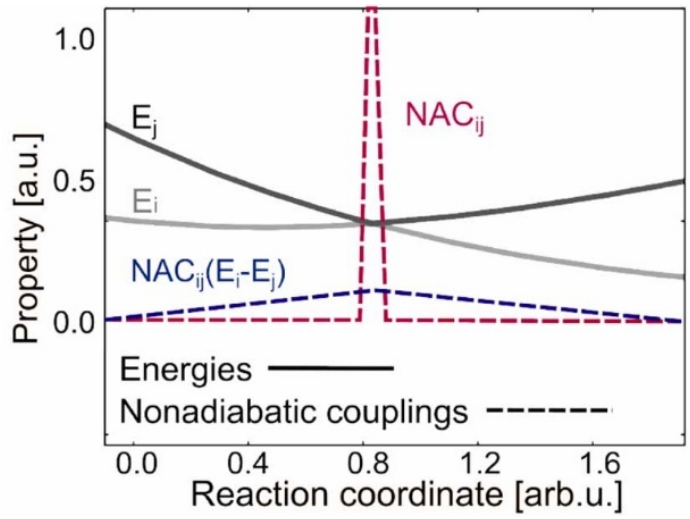

Figure 4. The norm of the NAC vectors along the reaction coordinate of $CH_2NH_2^+$.

Credit: Reproduced from Ref. [73] (Mach. Learn. Sci. Technol. 2020, 1, 025009) under CC-BY.

- NACs can have an arbitrary sign because they are calculated from two wavefunctions of different states, which have arbitrary signs (Figure 5). This feature makes it more complex to learn NACs, because even the same geometry may have NACs with opposite signs [3]. Thus, NACs signs may be corrected by standard approaches (such as by tracking wavefunction overlap [72]) before training the ML model on them. Alternatively, ML can choose the sign [62], leading to the NAC function easier to learn.

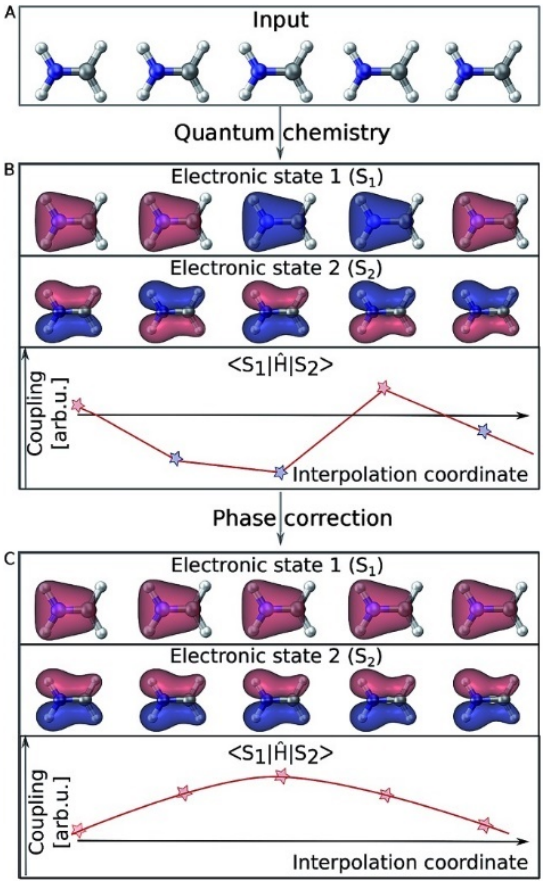

Figure 5. Arbitrary phases of wavefunctions and resulting properties.

Credit: Reproduced from Ref. [72] (Chem. Sci. 2019, 10, 8100–8107) under CC-BY.

Finally, we should mention that although couplings can be learned independently from potential energy surfaces, all of them can also be learned together by a single ML model, as is done in the SchNarc approach [74].

Learning spin-orbit couplings

Description of the ISC processes requires evaluations of spin–orbit couplings (SOCs) mediating transition between states with different multiplicity. ML can be applied to learn SOCs, although ML-assisted TSH dynamics for ISC is underexplored. In one study [74], ML was used within Surface Hopping including ARbitrary Couplings (SHARC) approach [81-82] (the generalization of the TSH method to include additional couplings like SOCs).

Learning SOCs should be much easier than learning NACs because the SOC data do not have the problem of singularity. However, the arbitrary sign problem still needs to be considered as SOCs are also properties derived from two wavefunctions. As in the case of NACs, this sign

problem can either be solved by traditional methods (e.g., by tracking phases through wave function overlaps) or by choosing easier-to-learn phases by ML. Again, Westermayr et al. [74] also suggested learning SOCs together with potential energy surfaces by a single ML model called SchNarc. This model was applied to model the slow ISC process in the CSH_2 molecule (investigated states were S_0, S_1, T_1, and T_2).

Training set generation

Readers may be very clear about the importance of the generation of training set right now since we have mentioned it several times. The ideal training set should be characterized by a few points and optimal coverage. Optimal coverage here means that the most representative configurations during excited-state processes should be included, enabling a good description of points around conical intersections. To obtain a satisfactory training set, many strategies can be used: sampling from MD trajectories [68]; sampling from conformational space [69] (e.g., by farthest-point sampling [106]); including enough points near conical intersections through handpicking [60] or sampling around conical intersections [75]; excluding problematic data points in critical regions [74].

Active learning is gaining momentum in ML-assisted TSH [3,31,72-74]. The next chapter is about *Constructing machine learning potentials with active learning* and will give details about this approach. Here we will briefly show how active learning is typically used in ML-assisted TSH and highlight points specific to TSH. Figure 6 demonstrates a typical scheme of active learning using the query-by-committee approach [107] with two NN models. Firstly, the initial training set is generated through scanning the normal modes, MD simulations, or other strategies. Then, two NN models can be trained based on the initial training set and applied to exploratory MD trajectories to generate new geometries with associated energies, gradients, and couplings. The points with the largest deviations in the predicted values of any of these properties by two NN models are flagged as potentially unreliable. Quantum chemical calculations are then performed for these points, and new data is then added to the training set. Models are retrained, NAMD trajectories propagated again until the difference between predictions of two models drops below a certain threshold.

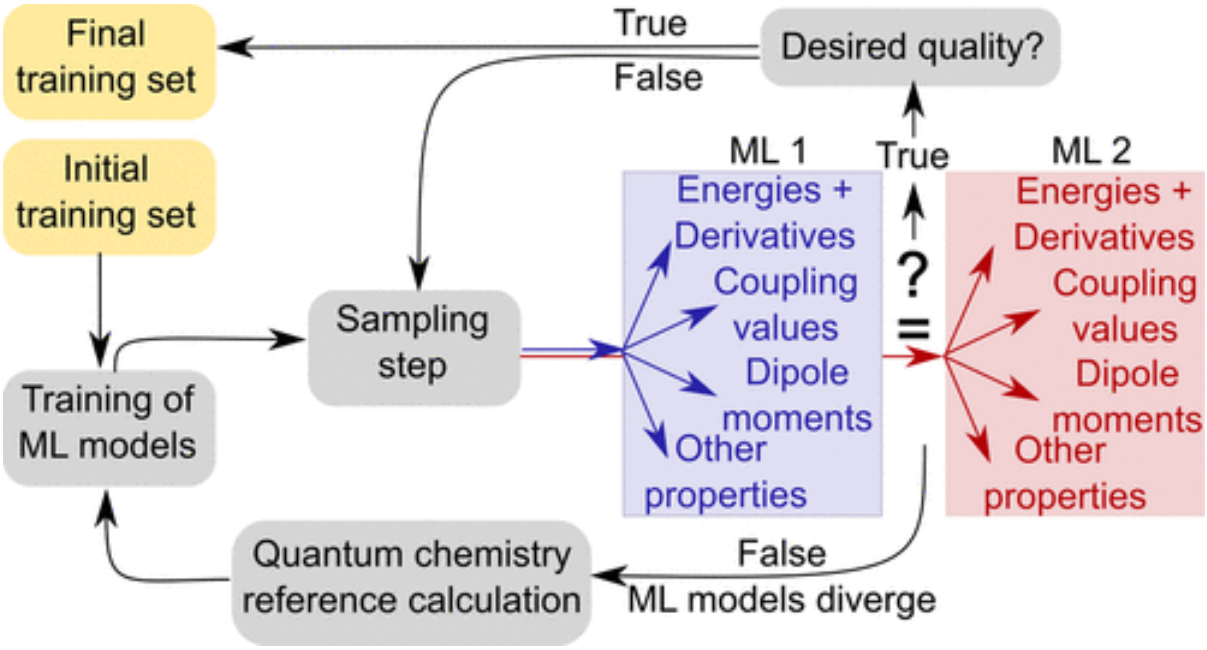

Figure 6. A typical active learning (adaptive sampling) scheme.

Credit: Reproduced from Ref. [3] (Chem. Rev. 2021, 121, 9873–9926) under CC-BY.

Case studies

This section demonstrates how ML accelerates two different types of excited-state dynamics: full quantum dynamics and trajectory surface hopping. Before that, we also show how to perform full quantum dynamics with the traditional HEOM approach for a better comparison. The full quantum dynamics is demonstrated on a standard example of the two-state spin-boson Hamiltonian, which is a general representation for many chemical systems. TSH is demonstrated on a specific chemical system – fulvene. This molecule is an excellent test system for surface hopping thanks to its nontrivial PES topography, with recurrences between S_1 and S_0 at an extended crossing seam with peaked and sloped conical intersections [108-109]. Instructions and materials needed to perform these case studies are provided at https://github.com/maxjr82/MLinQCbook16-NAMD.

Case study 1: Hierarchical equation of motion (HEOM)

In the case of a two-state system, the quantity of interest is the expectation value of $\hat{\sigma}_z$ (the population difference between the two states) which is expressed as

$$\langle \hat{\sigma}_z(t) \rangle = \mathrm{Tr}_S[\hat{\sigma}_z \hat{\rho}_S(t)] \tag{17}$$

where $\hat{\rho}_S(t)$ is the reduced density matrix of the system. Eq. (17) can be propagated with the HEOM method [Eq. (7)], which is computationally expensive (case study 1). However, ML can be used as a surrogate model to speed up the HEOM method, as discussed in the next section.

In this case study, we demonstrate the propagation of Eq. (17) for a two-state system. Using the HEOM method implemented in the publicly available QuTiP package [110], we generate trajectories of $\langle\hat{\sigma}_z(t)\rangle$ for all the possible combinations of the following parameters (all parameters are in atomic units (a.u.)): $\Delta = 1$, $\varepsilon \in \{0, 1\}$, $\lambda \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, $\omega_c \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, and $\beta \in \{0.1, 0.25, 0.5, 0.75, 1\}$, here $\beta = 1/(k_B T)$ with k_B as the Boltzmann constant and T as temperature. The total number of generated trajectories is 1000, propagated up to the maximum run-time of a trajectory $t_M = 20$ (a.u.) or ≈ 0.48 fs using timestep dt = 0.05 (a.u.). The data and script for running dynamics with HEOM are available at https://github.com/maxjr82/MLinQCbook16-NAMD. Here we have considered both symmetric ($\epsilon = 0$) and asymmetric ($\epsilon \neq 0$) cases of our two-state system. The cutoff frequency ranges from a mildly nonadiabatic environment ($\omega_c = 1$) to strongly nonadiabatic environment ($\omega_c = 10$). As has already been discussed, the computational cost of HEOM method depends on two factors; the number of exponential functions approximating the correlation function K (see Eq. (4)) and the depth of the hierarchy level L where K strongly depends on temperature and L on the strength of the system–bath coupling. An increase in K and L inevitably makes HEOM approach rather computationally expensive. For $K = 2$ and $L = 3$, the HEOM propagation with time-step dt = 0.05 (a.u) for time-length 20 (a.u.) takes few seconds on a single Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz while for $K = 5$ and $L = 30$, it takes ca. 13 hrs.

Case study 2: HEOM with machine learning

Instead of propagating dynamics with the expensive full quantum methods such as HEOM, we can train an ML model on the data and then propagate dynamics with less computational cost. To do that, we divide our data set of 1000 trajectories into two sets; a training set with 900 trajectories and a test set with 100 randomly chosen trajectories. Before training our ML model, we transfer our unsupervised training trajectories (unlabeled data) into supervised data (labeled data), as discussed in the Methods section (see Figure 1). We choose a short time trajectory with time-length t_m (memory time, in our case we choose $t_m = 4$) as a seed trajectory and we define the next $\langle\hat{\sigma}_z\rangle$ value at t_{m+1} as the target value making it the first training point. Though

our propagation time-step is dt = 0.05 in reference HEOM trajectories, we can use a different time-step of $\mathrm{dt}_{\mathrm{train}} = 0.1$ for training which makes our input vector 41 time-steps long. For the second training point, we delete the first value at t_0 and include the $\langle\hat{\sigma}_z\rangle$ value at t_{m+1} and as a target we choose $\langle\hat{\sigma}_z\rangle$ value at t_{m+2}. In a similar way, we transform all the training trajectories into supervised data (see Figure 1). Using the Gaussian kernel, we train a kernel ridge regression (KRR) model implemented in the publicly available MLatom package [111-113]. The data set of supervised trajectories is partitioned into two subsets: the training set, which contains 80% of the data, and the validation set with 20% randomly selected data points, which is used to optimize hyperparameters σ and λ on the logarithmic grid [111]. To minimize our training cost, we train separate models for both symmetric and asymmetric cases. The training for each case takes $\approx$ 8 hrs with parallel computation on 36 Intel(R) Xeon(R) Gold 6240 CPUs @ 2.60GHz (most of this time is taken by the optimization of hyperparameters σ and λ, and without optimization, the training takes less than 30 min). We prepare our input file input.inp for MLatom as shown in Figure 7 and we provide it to MLatom as `mlatom input.inp > output`.

```
createMLmodel                 # Requests creating a trained ML model
MLmodelOut=spin_boson.unf     # save KRR model with name 'spin_boson.unf'
XfileIn=X.dat                 # X.dat is the input file
Yfile=Y.dat                   # Y.dat is the file with target values
kernel=Gaussian               # use Gaussian kernel
sigma=opt                     # Requests optimization of sigma
lgSigmaL=-25                  # Lowest value on the logarithmic grid ……
                              # for the optimization of sigma
lgSigmaH=25                   # Highest value on the logarithmic grid ……
                              # for the optimization of sigma
lambda=opt                    # Requests optimization of lambda
lgLambdaL=-30                 # Lowest value on the logarithmic grid ……
                              # for the optimization of lambda.
sampling=random               # randomly split data set into ……
                              # training/validation sets using 80/20% split
```

Figure 7. Input options for training a KRR model using the publicly available MLatom package [111-113]. Any statement starting after # is a comment.

The input file is available at https://github.com/maxjr82/MLinQCbook16-NAMD. Upon completion, a trained KRR model `spin_boson.unf` should be saved. Now we can provide a short time trajectory of time length t_m from the hold-out test set trajectories, and the trained KRR model should be able to predict dynamics beyond $t_M = 20\ (a.u.)$ (used in the training trajectories propagated with HEOM) as shown in Figure 8. The process of prediction is iterative, where in order to predict $\langle\hat{\sigma}_z\rangle$ at the next time-step, the predicted values at previous

time-steps are included in the input. The algorithm for running dynamics with MLatom is presented in Figure 9. Figure 2 shows the time-dependent dynamics of $\langle\hat{\sigma}_z\rangle$ for our considered two-state system. It is worth emphasizing that the trained KRR model not only reproduce dynamics for unseen trajectories in the interpolation region $4 < t \leq 20$, but it is also capable of extrapolation beyond the training region $t > 20$ as shown in Figure 2. On a single Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz, prediction of the dynamics beyond the initial memory time t_m up to $t = 30(a.u.)$ takes ≈ 3 min.

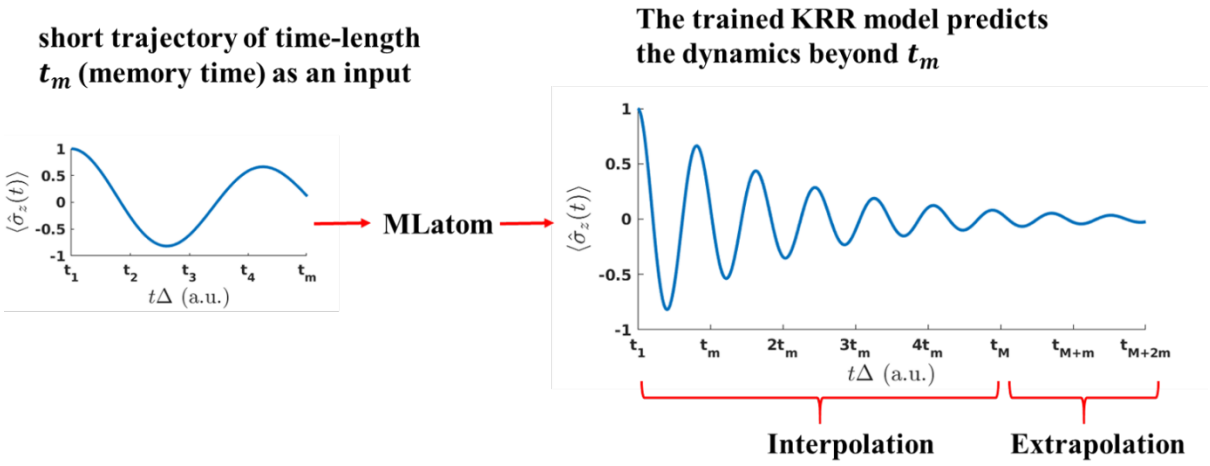

Figure 8. Propagation of quantum dynamics with MLatom where a short time trajectory of time-length t_m is provided as an input, and as a result, MLatom (using the trained KRR model) successfully predicts the dynamics beyond t_m [66].

```
filename=*.dat              # mention a filename from test data set
input = filename[0:t_m]     # take a trajectory of time-length t_m as an input

for i = 1: t_M;             # t_M is the number of propagation time-steps

     MLatom useMlmodel MlmodelIn=spin_boson.unf XfileIn=input ......
            YestFile=y_est.dat debug > output         # predict dynamics
     input <- y_est.dat     # include the predicted value in the input
Done

mv input pred_dynamics      # The predicted trajectory
```

Figure 9. A simple algorithm for doing full quantum dynamics with MLatom. The predictions beyond t_m is recursive, where predicted values at previous time-steps are included in the input for next time-step predictions.

As seen from the above case study, doing dynamics with ML is substantially faster. In our case, the amount of saved time, which depends on t_m and trajectory time (t_M), is ca. 90%. In contrast to HEOM, where the computational cost depends on temperature, the computational cost for

propagating dynamics with ML remains the same for all cases. Such a substantial reduction in computational cost may allow longer simulations with the HEOM method.

Case study 3: Trajectory-surface hopping with machine learning

In this case study, we demonstrate in a tutorial way step-by-step how to perform a TSH NAMD simulation using trained machine learning models. The fulvene molecule will be used as a prototype of a photoactive system. This molecule undergoes significant geometry deformation (C–CH_2 bond stretching and CH_2 rotation) during the internal conversion process from the S_1 to S_0 state [93,114]. Two independent program packages will be necessary to follow the tutorial and run the simulations: (i) the MLatom program [111-113] will be used to train the machine learning model, and (ii) the Newton-X software [115-116] interfaced with MLatom will be responsible for propagating the dynamics. Instructions and materials needed to perform this case study are provided at https://github.com/maxjr82/MLinQCbook16-NAMD.

Step 1: Create the training set

This is a crucial step to guarantee good accuracy for the machine learning model. It is, however, rather involved, as we discussed in the Methods section. We provide the training set by selecting points from a set of NAMD trajectories already computed at the quantum mechanical level. It can be downloaded from https://figshare.com/articles/dataset/Fulvene_DC-FSSH/14446998/1, but it is not necessary for this study. This data set contains 200 TSH CASSCF trajectories, run at the SA-2-CAS(6,6)/6-31G level with a time step of 0.1 fs up to a maximum time of 60 fs. More than 120000 data points are available in this fulvene dataset with information of molecular geometries, gradients, the potential energy of the S_0 and S_1 states, and the nonadiabatic coupling vectors between the two energy surfaces.

For this case study, we have randomly picked 40000 points from the first 199 NAMD trajectories while the trajectory 200 is left out as a reference to test the performance of the trained model. This training set is provided in three different files:

- `xyz_train_40k.dat` containing XYZ geometries;
- `en_s0_train_40k.dat` and `en_s1_train_40k.dat` with the potential energies in the ground and the first excited states

All quantities provided in these data files are given in atomic units; the order of energies corresponds to the order of XYZ geometries.

Step 2: Model training (MLatom/KREG)

To run a full nonadiabatic dynamics for the fulvene example, we will train two independent models to predict the potential energy of each electronic state. Next, the gradients are calculated by the MLatom program as the derivative of the trained model. NACS will be computed by the molecular dynamics program using the TD-BA approximation [93].

The readers can train the model themselves as described in the tutorial instructions or use the KREG models provided with this tutorial to save time because, depending on your machine, the training of a model on energies in one state may take up to a dozen hours. Either way, in the end, trained models are saved in the binary files `mlmod_engrad1.unf` and `mlmod_engrad2.unf` containing the KREG models that will be used to make predictions of energy and gradients for the electronic state 1 (S_0) and the electronic state 2 (S_1) at each time step during the dynamics.

Step 3: Run the ML-NAMD simulation (NX/MLatom)

Suppose the accuracy of the model obtained in step 2 is satisfactory (check the MLatom outputs for the error metrics, but note that the ultimate check of the quality of the ML model is in the analysis of many trajectories). In that case, we can use this model as an estimator for each state's potential energy and corresponding gradients to propagate the dynamics. To run one NAMD trajectory for the fulvene molecule using the ML models as a predictor, the reader can either use the input examples already provided with this tutorial or create a new TRAJ folder from scratch using the Newton-X interactive program, nxinp. In the latter case, one must ensure that the prog flag is in the `control.dyn` input file of Newton-X is set to 11, meaning that MLatom will be called by NX in every step of the dynamics to compute the required quantities. Also, the flag vdoth in the `sh.inp` file must be set to 2 to use the TD-BA to compute the nonadiabatic couplings. The JOB_NAD directory should contain the MLatom models `mlmod_engrad1.unf` and `mlmod_engrad2.unf` obtained in the previous step. In addition, one should also copy the `eq.xyz` file into the JOB_NAD folder.

Either way, the dynamics is ready to start with Newton-X as described in the tutorial manual. After finishing the simulation, the reader can visualize the results by using the plot utility script of Newton-X and obtain a plot similar to the one shown in Figure 10.

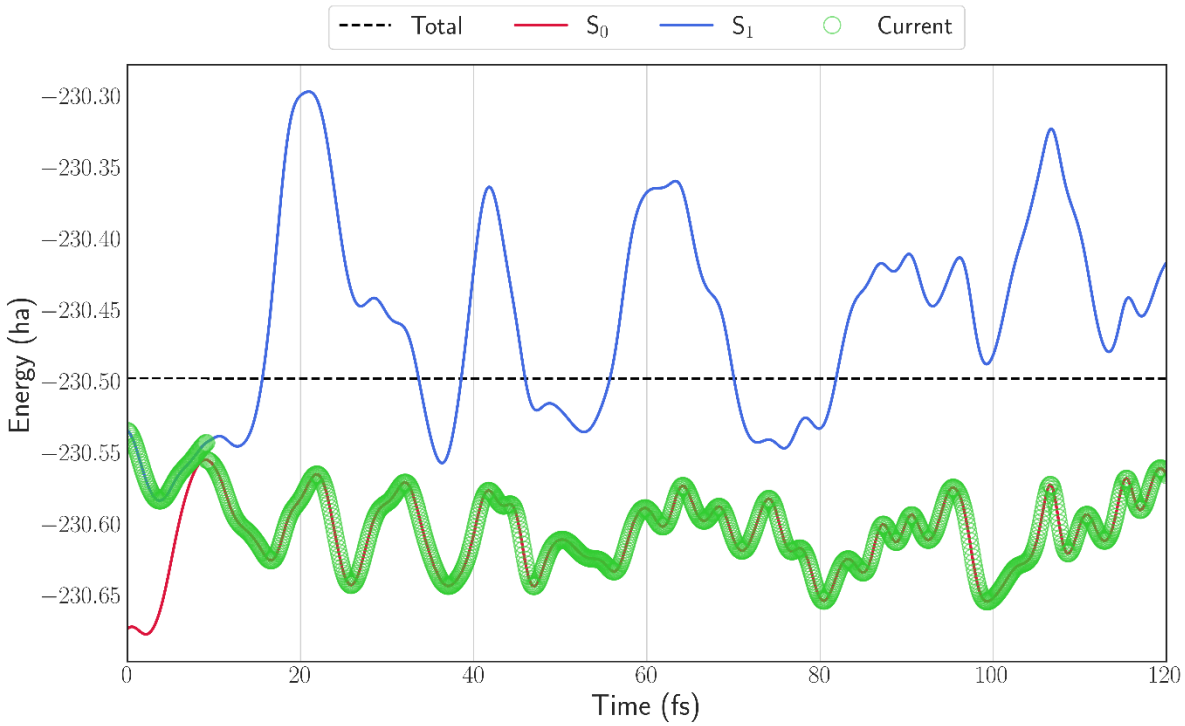

Figure 10. The plot shows the time evolution of the potential energies during TSH ML-NAMD propagation for a fulvene molecule in a single trajectory. The simulation starts in the first excited state S_1 (blue line) and then at ca. 9.2 fs hops from the S_1 state to S_0 state (red line) and stays there. The active surface, i.e., the current electronic state at each step, is shown with green markers. The total energy is shown with the dashed line, and it is evident that it does not change during dynamics.

The plot may differ because hopping is a stochastic process and can happen at different times when the simulation is restarted. ML NAMD is very fast, as the reader will see after running this simulation. It takes ca. 1 hour with a relatively inefficient implementation, while the reference dynamics take 15 hours. Importantly, we can observe that ML NAMD simulation also features hopping events, which is the first indication that the quality of ML PESs may be sufficient to describe challenging regions near the conical intersection because the reference quantum chemical NAMD trajectory also features hopping. To perform a more rigorous analysis of the quality of ML PES, one would need to run a large number (at least hundreds) of such trajectory simulations starting with different geometries and velocities and perform statistical analysis on them to calculate the populations and lifetimes in each state and other properties of interest such as structural parameters.

Conclusions and outlook

Excited-state nonadiabatic molecular dynamics (NAMD) simulations reveal the temporal evolution and provide insights into underlying principles of photophysics and photochemistry processes. However, the application of NAMD to large systems and long-time excited-state processes has been limited by the expensive underlying quantum chemical calculations. This chapter introduced the combination of ML with two classes of excited-state dynamics methods—full quantum dynamics (HEOM as an example) and mixed quantum-classical dynamics (TSH as an example) and showed the challenges and strategies of using ML to accelerate the NAMD simulations. In the ML-assisted HEOM section, the strategy of learning from multiple trajectories is introduced. In the ML-assisted TSH section, how to treat the complexity of potential energy surfaces, the singularity of NACs, and the arbitrary sign problem of NACs and SOCs are discussed, and strategies for generating training sets are also mentioned. As we have discussed, these two methods (TSH and HEOM) represent different aspects of NAMD investigations. On the one hand, TSH allows treating realistic molecules at the cost of neglecting many quantum effects. On the other, HEOM describes global quantum effects in the dynamics of dissipative systems using a system-bath model Hamiltonian. In the case studies, we have demonstrated that ML-assisted NAMD saves 90% of computational time compared to the traditional NAMD.

Since this is an emerging and rapidly changing field, it is difficult to predict what will happen in the future. We can be, however, sure that still much research is needed to explore when and why ML fails and how to mitigate such failures in ML-assisted NAMD. For example, the ML-TSH approach failed to learn nonadiabatic couplings for cubane systems [117], although the same approach worked for other systems [50]. As usual in ML applications, one of the largest problems is how to reduce the cost of generating training data. Finally, one can envision that someday we will be able to use universal ML models for running dynamics, and this model could be updated with more data as needed.

Acknowledgements

POD acknowledges funding by the National Natural Science Foundation of China (No. 22003051), 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. MB and MPJ were financially supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 832237 (project SubNano).

References

[1] S. Mai, L. González, *Angewandte Chemie International Edition* **2020,** *59*, 16832–16846.
[2] L. González, D. Escudero, L. Serrano-Andrés, *ChemPhysChem* **2012,** *13*, 28–51.
[3] J. Westermayr, P. Marquetand, *Chemical Reviews* **2021,** *121*, 9873–9926
[4] M. Han, S. Zhu, S. Lu, Y. Song, T. Feng, S. Tao, J. Liu, B. Yang, *Nano Today* **2018,** *19*, 201–218
[5] F. Strieth-Kalthoff, M. J. James, M. Teders, L. Pitzer, F. Glorius, *Chemical Society Reviews* **2018,** *47*, 7190–7202.
[6] L. Lindh, P. Chábera, N. W. Rosemann, J. Uhlig, K. Wärnmark, A. Yartsev, V. Sundström, P. Persson, *Catalysts* **2020,** *10*, 315.
[7] C. C. Jumper, S. Rafiq, S. Wang, G. D. Scholes, *Current Opinion in Chemical Biology* **2018,** *47*, 39–46.
[8] D. R. Whang, D. H. Apaydin, *ChemPhotoChem* **2018,** *2*, 148–160.
[9] A. Stirbet, D. Lazár, Y. Guo, G. Govindjee, *Annals of Botany* **2020,** *126*, 511–537
[10] T. Katagi, *Journal of Pesticide Science* **2018,** *43*, 57–72
[11] C. S. Blaszczak-Boxe, A. Saiz-Lopez, *Atmospheric Environment* **2018,** *193*, 224–241.
[12] B. Demmig-Adams, J. J. Stewart, M. López-Pozo, S. K. Polutchko, W. W. Adams, *Molecules* **2020,** *25*, 5825.
[13] K. P. Lawrence, P. F. Long, A. R. Young, *Current Medicinal Chemistry* **2018,** *25*, 5512–5527.
[14] E. L. Holt, K. M. Krokidi, M. A. Turner, P. Mishra, T. S. Zwier, N. d. N. Rodrigues, V. G. Stavros, *Physical Chemistry Chemical Physics* **2020,** *22*, 15509–15519.
[15] M. A. van der Horst, K. J. Hellingwerf, *Accounts of Chemical Research* **2004,** *37*, 13–20.
[16] T. Kottke, A. Xie, D. S. Larsen, W. D. Hoff, *Annual Review of Biophysics* **2018,** *47*, 291–313.
[17] A. Losi, W. Gärtner, *Photochemistry and Photobiology* **2017,** *93*, 141–158.
[18] Y. Lin, Y. Li, X. Zhan, *Chemical Society Reviews* **2012,** *41*, 4245–4272.
[19] M. Azzouzi, J. Yan, T. Kirchartz, K. Liu, J. Wang, H. Wu, J. Nelson, *Physical Review X* **2018,** *8*, 031055.
[20] P. A. Lane, Z. H. Kafafi, Solid-State Organic Photovoltaics: A Review of Molecular and Polymeric Devices. In *Organic Photovoltaics*, **2017**; pp. 49–104.
[21] B. J. Alder, T. E. Wainwright, *The Journal of Chemical Physics* **1959,** *31*, 459–466.
[22] M. E. Tuckerman, G. J. Martyna, *The Journal of Physical Chemistry B* **2000,** *104*, 159–178.
[23] M. Karplus, J. A. McCammon, *Nature Structural Biology* **2002,** *9*, 646–652.
[24] D. C. Rapaport, *The Art of Molecular Dynamics Simulation*. Cambridge University Press: **2004**.
[25] B. Leimkuhler, C. Matthews, *Molecular Dynamics*. Springer International Publishing: **2016**.
[26] M. H. Beck, A. Jäckle, G. A. Worth, H. D. Meyer, *Physics Reports* **2000,** *324*, 1–105.
[27] N. Makri, D. E. Makarov, *The Journal of Chemical Physics* **1995,** *102*, 4600–4610.
[28] Y. Yao, K.-W. Sun, Z. Luo, H. Ma, *The Journal of Physical Chemistry Letters* **2018,** *9*, 413–419.
[29] A. G. Redfield, *IBM Journal of Research and Development* **1957,** *1*, 19–31.
[30] Y. Tanimura, R. Kubo, *Journal of the Physical Society of Japan* **1989,** *58*, 101–114.
[31] J. Westermayr, P. Marquetand, *Machine Learning: Science and Technology* **2020,** *1*, 043001.
[32] M. A. Cazalilla, J. B. Marston, *Physical Review Letters* **2002,** *88*, 256403.

[33] J. Shao, *The Journal of Chemical Physics* **2004,** *120*, 5053–5056.
[34] W. Koch, F. Großmann, J. T. Stockburger, J. Ankerhold, *Physical Review Letters* **2008,** *100*, 230402.
[35] A. Ullah, L. Han, Y.-A. Yan, X. Zheng, Y. Yan, V. Chernyak, *The Journal of Chemical Physics* **2020,** *152*, 204106.
[36] R. Rosenbach, J. Cerrillo, S. F. Huelga, J. Cao, M. B. Plenio, *New Journal of Physics* **2016,** *18*, 023035.
[37] J. Prior, A. W. Chin, S. F. Huelga, M. B. Plenio, *Physical Review Letters* **2010,** *105*, 050404.
[38] S. M. Greene, V. S. Batista, *Journal of Chemical Theory and Computation* **2017,** *13*, 4034–4042.
[39] L. Diósi, N. Gisin, W. T. Strunz, *Physical Review A* **1998,** *58*, 1699–1712.
[40] Q. Shi, E. Geva, *The Journal of Chemical Physics* **2003,** *119*, 12063–12076.
[41] X. Li, J. C. Tully, H. B. Schlegel, M. J. Frisch, *The Journal of Chemical Physics* **2005,** *123*, 084106.
[42] J. C. Tully, *The Journal of Chemical Physics* **1990,** *93*, 1061–1071.
[43] G. Stock, M. Thoss, *Physical Review Letters* **1997,** *78*, 578–581.
[44] B. F. E. Curchod, T. J. Martínez, *Chemical Reviews* **2018,** *118*, 3305–3336.
[45] W. H. Miller, S. J. Cotton, *Faraday Discussions* **2016,** *195*, 9–30.
[46] X. Sun, H. Wang, W. H. Miller, *The Journal of Chemical Physics* **1998,** *109*, 7064–7074.
[47] I. R. Craig, D. E. Manolopoulos, *The Journal of Chemical Physics* **2004,** *121*, 3368–3373.
[48] J. Cao, G. A. Voth, *The Journal of Chemical Physics* **1993,** *99*, 10070–10073.
[49] R. Crespo-Otero, M. Barbatti, *Chemical Reviews* **2018,** *118*, 7026–7068.
[50] T. Terashima, M. Shiga, S. Okazaki, *The Journal of Chemical Physics* **2001,** *114*, 5663–5673.
[51] T. Cusati, G. Granucci, M. Persico, *Journal of the American Chemical Society* **2011,** *133*, 5109–5123.
[52] A. J. Atkins, L. González, *The Journal of Physical Chemistry Letters* **2017,** *8*, 3840–3845.
[53] L. M. Ibele, Y. Lassmann, T. J. Martínez, B. F. Curchod, *The Journal of Chemical Physics* **2021,** *154*, 104110.
[54] W. Xie, D. Holub, T. Kubař, M. Elstner, *Journal of Chemical Theory and Computation* **2020,** *16*, 2071–2084.
[55] L. Wang, J. Qiu, X. Bai, J. Xu, *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2020,** *10*, e1435.
[56] M. Mališ, S. Luber, *Journal of Chemical Theory and Computation* **2020,** *16*, 4071–4086.
[57] M. Saitow, Y. Kurashige, T. Yanai, *Journal of Chemical Theory and Computation* **2015,** *11*, 5120–5131.
[58] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, A. Walsh, *Nature* **2018,** *559*, 547–555.
[59] P. O. Dral, *The Journal of Physical Chemistry Letters* **2020,** *11*, 2336–2347.
[60] P. O. Dral, Quantum Chemistry Assisted by Machine Learning. In *Advances in Quantum Chemistry*, Ruud, K.; Brändas, E. J., Eds. Academic Press: **2020**; Vol. 81, pp. 291–324.
[61] O. A. von Lilienfeld, K.-R. Müller, A. Tkatchenko, *Nature Reviews Chemistry* **2020,** *4*, 347–358.

[62] J. Li, P. Reiser, B. R. Boswell, A. Eberhard, N. Z. Burns, P. Friederich, S. A. Lopez, *Chemical Science* **2021,** *12*, 5302–5314.
[63] A. V. Akimov, *The Journal of Physical Chemistry Letters* **2021,** *12*, 12119–12128.
[64] H. Rodríguez, L. E., A. A. Kananenka, *The Journal of Physical Chemistry Letters* **2021,** *12*, 24762–483.
[65] K. Lin, J. Peng, F. L. Gu, Z. Lan, *The Journal of Physical Chemistry Letters* **2021,** *12*, 10225–10234.
[66] A. Ullah, P. O. Dral, *New Journal of Physics* **2021,** *23*, 113019.
[67] A. Ullah, P. O. Dral, *Nature Communications* **2022**, accepted. See preprint on ChemRxiv: https://doi.org/10.26434/chemrxiv-2021-d2ksx.
[68] D. Hu, Y. Xie, X. Li, L. Li, Z. Lan, *The Journal of Physical Chemistry Letters* **2018,** *9*, 2725–2732.
[69] P. O. Dral, M. Barbatti, W. Thiel, *The Journal of Physical Chemistry Letters* **2018,** *9*, 5660–5663.
[70] W.-K. Chen, X.-Y. Liu, W. Fang, P. O. Dral, G. Cui, *The Journal of Physical Chemistry Letters* **2018,** *9*, 6702–6708.
[71] W.-K. Chen, W.-H. Fang, G. Cui, *The Journal of Physical Chemistry Letters* **2019,** *10*, 7836–7841.
[72] J. Westermayr, M. Gastegger, M. F. S. J. Menger, S. Mai, L. González, P. Marquetand, *Chemical Science* **2019,** *10*, 8100–8107.
[73] J. Westermayr, F. A. Faber, A. S. Christensen, O. A. von Lilienfeld, P. Marquetand, *Machine Learning: Science and Technology* **2020,** *1*, 025009.
[74] J. Westermayr, M. Gastegger, P. Marquetand, *The Journal of Physical Chemistry Letters* **2020,** *11*, 3828–3834.
[75] M. Ardiansyah, K. R. Brorsen, *The Journal of Physical Chemistry A* **2020,** *124*, 9326–9331.
[76] Y. Zhou, Y. Yan, J. Shao, *Europhysics Letters* **2005,** *72*, 334–340.
[77] Y. Tanimura, *The Journal of Chemical Physics* **2014,** *141*, 044114.
[78] L. Song, Q. Shi, *The Journal of Chemical Physics* **2015,** *143*, 194106.
[79] D. Wu, Z. Hu, J. Li, X. Sun, *The Journal of Chemical Physics* **2021,** *155*, 224104.
[80] C. Zhu, K. Nobusada, H. Nakamura, *The Journal of Chemical Physics* **2001,** *115*, 3031–3044.
[81] M. Richter, P. Marquetand, J. González-Vázquez, I. Sola, L. González, *Journal of Chemical Theory and Computation* **2011,** *7*, 1253–1258.
[82] S. Mai, P. Marquetand, L. González, *WIREs Computational Molecular Science* **2018,** *8*, e1370.
[83] J. E. Subotnik, A. Jain, B. Landry, A. Petit, W. Ouyang, N. Bellonzi, *Annual Review of Physical Chemistry* **2016,** *67*, 387–417.
[84] L. Yu, C. Xu, Y. Lei, C. Zhu, Z. Wen, *Physical Chemistry Chemical Physics* **2014,** *16*, 25883–25895.
[85] M. Barbatti, *Journal of Chemical Theory and Computation* **2021,** *17*, 3010–3018.
[86] G. Granucci, M. Persico, *The Journal of Chemical Physics* **2007,** *126*, 134114.
[87] M. Pederzoli, J. Pittner, *The Journal of Chemical Physics* **2017,** *146*, 114101.
[88] G. Granucci, M. Persico, G. Spighi, *The Journal of Chemical Physics* **2012,** *137*, 22A501.
[89] G. A. Meek, B. G. Levine, *The Journal of Physical Chemistry Letters* **2014,** *5*, 2351–2356.
[90] I. G. Ryabinkin, J. Nagesh, A. F. Izmaylov, *The Journal of Physical Chemistry Letters* **2015,** *6*, 4200–4203.

[91] S. Hammes‐Schiffer, J. C. Tully, *The Journal of Chemical Physics* **1994,** *101*, 4657‐4667.
[92] F. Plasser, R. Crespo-Otero, M. Pederzoli, J. Pittner, H. Lischka, M. Barbatti, *Journal of Chemical Theory and Computation* **2014,** *10*, 1395–1405.
[93] M. T. do Casal, J. Toldo, M. Pinheiro Jr, M. Barbatti, *Open Research Europe* **2021,** *1*.
[94] S. Mukherjee, M. Pinheiro Jr, B. Demoulin, M. Barbatti, *Philosophical Transactions of The Royal Society A* **2022,** *380*, 20200382.
[95] D. R. Yarkony, *Reviews of Modern Physics* **1996,** *68*, 985.
[96] W. Domcke, D. Yarkony, H. Köppel, *Conical Intersections: Electronic Structure, Dynamics & Spectroscopy*. World Scientific: **2004**; Vol. 15.
[97] W. Domcke, D. R. Yarkony, H. Köppel, *Conical Intersections: Theory, Computation and Experiment*. World Scientific: **2011**; Vol. 17.
[98] P. O. Dral, M. Barbatti, *Nature Reviews Chemistry* **2021,** *5*, 388–405.
[99] P. O. Dral, A. Owens, S. N. Yurchenko, W. Thiel, *The Journal of Chemical Physics* **2017,** *146*, 244108.
[100] P. Habitz, C. Votava, *The Journal of Chemical Physics* **1980,** *72*, 5532-5539.
[101] Y. Shu, D. G. Truhlar, *Journal of Chemical Theory and Computation* **2020,** *16*, 6456–6464.
[102] G. W. Richings, S. Habershon, *Chemical Physics Letters* **2017,** *683*, 228–233.
[103] D. M. G. Williams, A. Viel, W. Eisfeld, *The Journal of Chemical Physics* **2019,** *151*, 164118.
[104] G. W. Richings, S. Habershon, *The Journal of Chemical Physics* **2020,** *152*, 154108.
[105] G. W. Richings, S. Habershon, *The Journal of Physical Chemistry A* **2020,** *124*, 9299–9313.
[106] A. P. Bartók, S. De, C. Poelking, N. Bernstein, J. R. Kermode, G. Csányi, M. Ceriotti, *Science Advances* **2017,** *3*, e1701816.
[107] H. S. Seung, M. Opper, H. Sompolinsky. *Query by committee*. Proceedings of the fifth annual workshop on Computational learning theory, **1992**; pp. 287-294.
[108] L. Blancafort, F. Gatti, H.-D. Meyer, *The Journal of Chemical Physics* **2011,** *135*, 134303.
[109] L. M. Ibele, B. F. E. Curchod, *Physical Chemistry Chemical Physics* **2020,** *22*, 15183–15196.
[110] J. R. Johansson, P. D. Nation, F. Nori, *Computer Physics Communications* **2012,** *183*, 1760–1772.
[111] P. O. Dral, *Journal of Computational Chemistry* **2019,** *40*, 2339–2347.
[112] P. O. Dral, F. Ge, B.-X. Xue, Y.-F. Hou, M. Pinheiro, J. Huang, M. Barbatti, *Topics in Current Chemistry* **2021,** *379*, 27.
[113] P. O. Dral, P. Zheng, B.-X. Xue, F. Ge, Y.-F. Hou, M. P. Jr, *MLatom: A Package for Atomistic Simulations with Machine Learning*, version 2; Xiamen University: Xiamen, China, **2013–2022. http://MLatom.com**.
[114] D. Mendive-Tapia, B. Lasorne, G. A. Worth, M. J. Bearpark, M. A. Robb, *Physical Chemistry Chemical Physics* **2010,** *12*, 15725–15733.
[115] M. Barbatti, M. Ruckenbauer, F. Plasser, J. Pittner, G. Granucci, M. Persico, H. Lischka, *WIREs Computational Molecular Science* **2014,** *4*, 26–33.
[116] M. Barbatti, G. Granucci, M. Ruckenbauer, F. Plasser, R. Crespo-Otero, J. Pittner, M. Persico, H. Lischka, *NEWTON-X: A package for Newtonian dynamics close to the crossing seam.*, version 2; **2016. www.newtonx.org**.
[117] J. Li, R. Stein, D. M. Adrion, S. A. Lopez, *Journal of the American Chemical Society* **2021,** *143*, 20166–20175.