Article

MLatom 3: A Platform for Machine Learning-Enhanced Computational Chemistry Simulations and Workflows

Pavlo O. Dral,* Fuchun Ge, Yi-Fan Hou, Peikun Zheng, Yuxinxin Chen, Mario Barbatti, Olexandr Isayev, Cheng Wang, Bao-Xin Xue, Max Pinheiro Jr, Yuming Su, Yiheng Dai, Yangtao Chen, Lina Zhang, Shuang Zhang, Arif Ullah, Quanhao Zhang, and Yanchi Ou

Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: Machine learning (ML) is increasingly becoming a common tool in computational chemistry. At the same time, the rapid development of ML methods requires a flexible software framework for designing custom workflows. MLatom 3 is a program package designed to leverage the power of ML to enhance typical computational chemistry simulations and to create complex workflows. This open-source package provides plenty of choice to the users who can run simulations with the command-line options, input files, or with scripts using MLatom as a Python package, both on their computers and on the online XACS cloud computing service at XACScloud.com. Computational chemists can calculate energies and thermochemical properties, optimize geometries, run molecular and quantum dynamics, and simulate (ro)vibrational, one-photon UV/vis absorption, and two-photon absorption spectra with ML, quantum mechanical, and combined models. The users can choose from an extensive library of methods containing pretrained ML models and quantum mechanical approximations such as AIQM1 approaching coupled-cluster accuracy. The developers can build their own models using various ML algorithms. The great flexibility of MLatom is largely due to the extensive use of the interfaces to many state-of-the-art software packages and libraries.
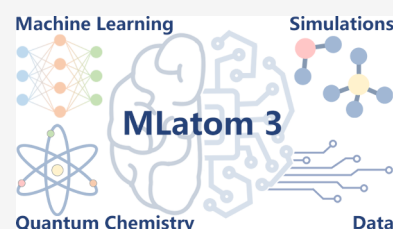
1. INTRODUCTION

Computational chemistry simulations are common in chemistry research thanks to abundant general-purpose software, most of which have started as purely quantum mechanical (QM) and molecular mechanical (MM) packages. More recently, the rise of artificial intelligence (AI)/machine learning (ML) applications for chemical simulations has caused the proliferation of programs mostly focusing on specific ML tasks such as learning potential energy surfaces (PESs).[1−17] The rift between the development of the traditional QM and MM packages on the one hand and ML programs on the other hand is bridged to some extent by the higher-level library ASE,[18] which enables usual computational tasks via interfacing heterogeneous software. The further integration of QM, MM, and ML has been prompted by the maturing of ML techniques and is evidenced by the growing trend of incorporating ML methods in the QM and MM computational chemistry software.[13,19−21]

Against this backdrop, the MLatom package started in 2013 as a pure standalone ML package to provide a general-purpose experience for computational chemists akin to the black-box QM packages.[22] The early MLatom could be used for training, testing, and using ML models and their combinations with QM methods (e.g., Δ-learning[23] and learning of Hamiltonian parameters[24]), accurate representation of PES,[25,26] sampling of points from data sets,[26] ML-accelerated nonadiabatic dynamics,[27] and materials design.[28] The fast pace of method

and software development in QM, MM, ML, and other computational science domains led to MLatom 2, which started to include interfaces to third-party packages.[29] Such an approach provided a unique opportunity for the package users to choose one of the many established ML models, similar to the users of the traditional QM software who can choose one of the many QM methods. MLatom 2 could perform training of the ML models, evaluate their accuracy, and then use the models for geometry optimization and frequency calculations. Special workflows were also implemented, such as acceleration of the absorption UV/vis spectra calculations with ML[30] and prediction of two-photon absorption spectra.[31] In addition, MLatom 2 could be used to perform simulations with the general-purpose AI-enhanced QM method[32] AIQM1 and universal machine learning potentials of the ANI family[2,33−35] with the accurate scheme developed for calculating heats of formation[36] with uncertainty quantification with these methods.

With time, the need to develop increasingly complex workflows that incorporate ML and QM for a broad range of

Models

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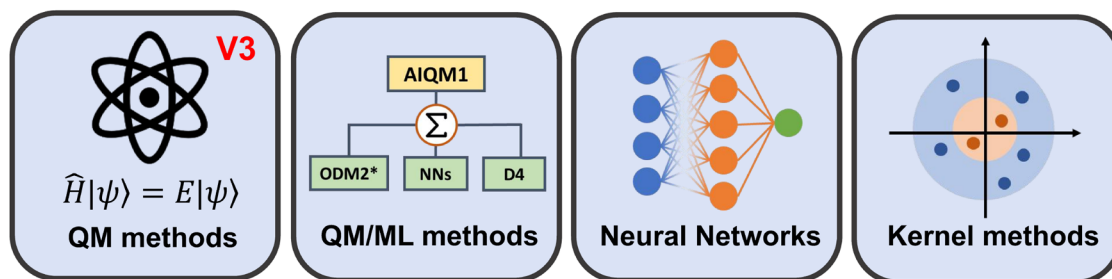
$\hat{H}|\psi\rangle = E|\psi\rangle$

QM/ML methods

Neural Networks

Kernel methods

Simulations

Single point calculations

$$E,\ \frac{\partial E}{\partial \mathbf{x}},\ \frac{\partial^2 E}{\partial \mathbf{x}^2}$$

Geometry optimizations

Frequencies & thermochemistry

$$\Delta H_{\mathrm{at},T} = \left[\sum_A H_T(A)\right] - H_T$$

Molecular dynamics **V3**

Quantum dissipative dynamics with ML

(Ro)vibrational spectra

UV/vis spectra (ML-NEA)

Two-photon absorption spectra (ML-TPA)

Implementation and use

➢ Data in different formats and types **V3**

molecular_database   molecule   atom

➢ Python API **V3**

```
import mlatom as ml
```

➢ Input file

```
ANI-1ccx
geomopt
xyzfile=init.xyz
optxyz=opt.xyz
```

➢ Command line

```
> $mlatom ANI-1ccx geomopt xyzfile=init.xyz optxyz=opt.xyz
```
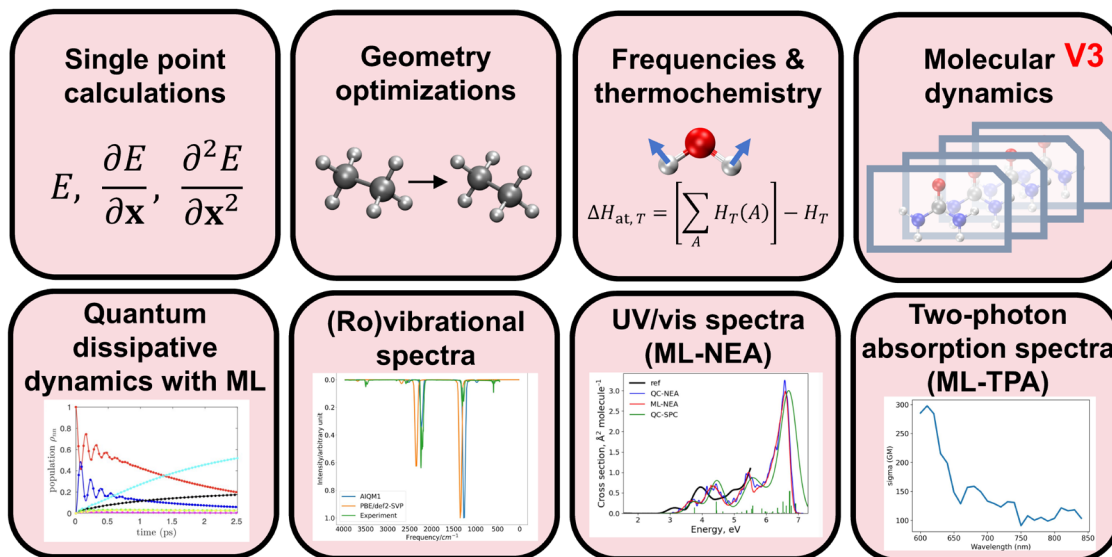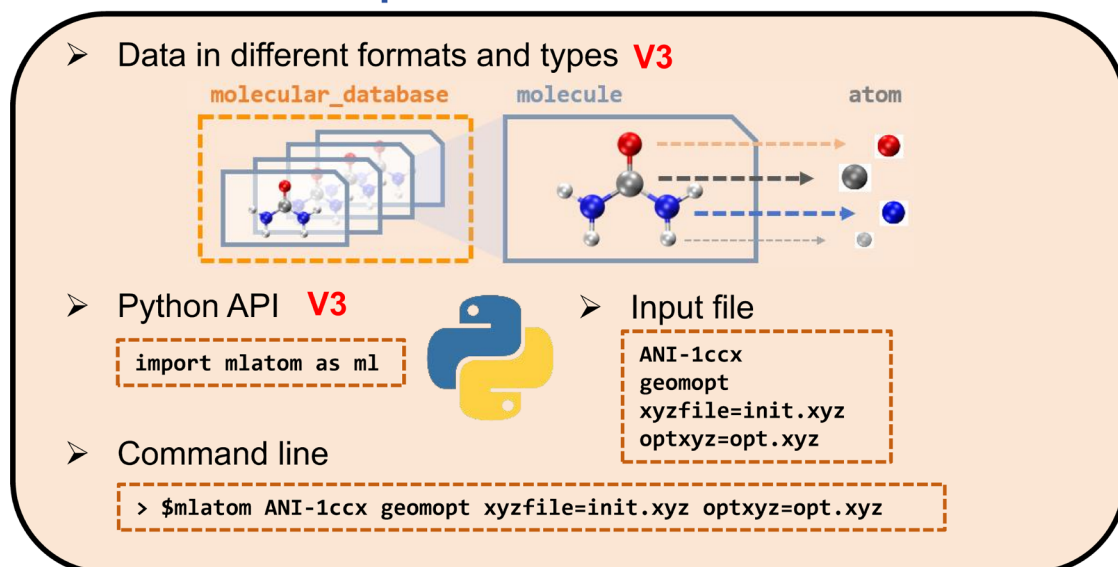
* 'V3' marks implementations released in MLatom 3

Figure 1. Overview of MLatom 3 capabilities. The plot in panel "Quantum dissipative dynamics with ML" is adapted with permission from ref 38. Copyright 2022, the Authors. The plot in panel "UV/vis spectra (ML-NEA)" is adapted from ref 29. Copyright 2021, the Authors.

applications has necessitated the rethink and redesign of MLatom to enable the rapid development of highly customized routines. These additional design requirements for MLatom to serve not only as a black-box general-purpose package but also as a flexible platform for developers resulted in a significant extension, redesign, and rewrite of the program. The subsequent
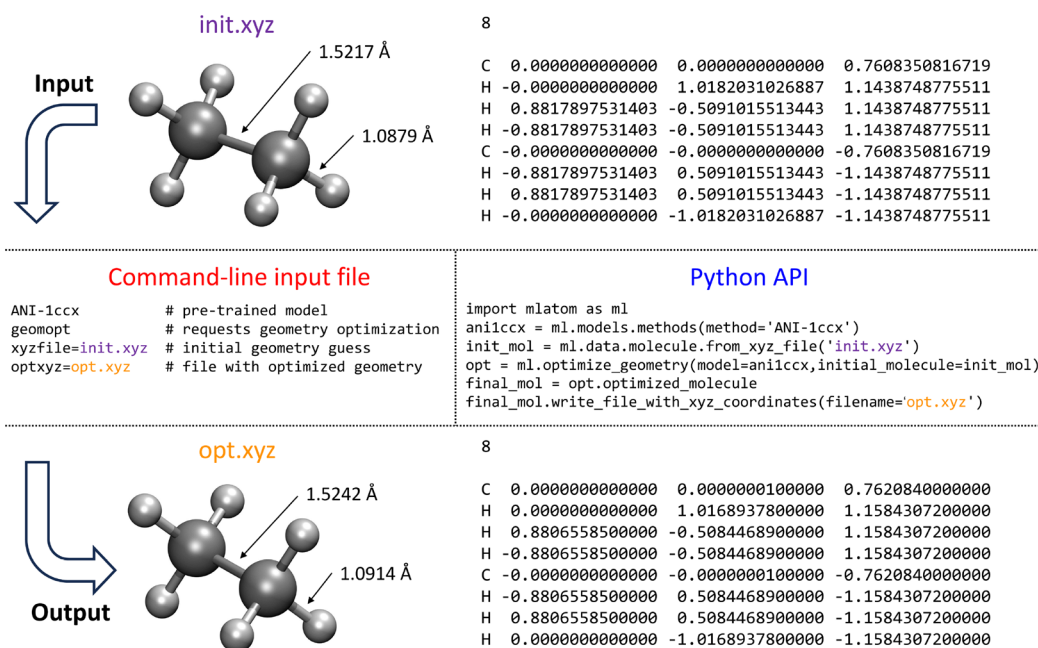
Figure 2. Side-by-side comparison of the usage of MLatom in both the command-line mode and via Python API for a common task of geometry optimization with one of the pretrained ML models ANI-1ccx.

upgrade has allowed the use of MLatom through the versatile Python API (MLatom PyAPI) and also included the implementation of more simulation tasks, such as molecular and quantum dynamics, and the support of QM methods and composite schemes based on the combinations of QM and ML models. This upgrade was released[37] as MLatom 3 in 2023, 10 years after the start of the project. During this decade, MLatom went through a drastic transformation from a pure Fortran package to a predominantly Python package, with one-third of the code written in Fortran for efficient implementations of critical parts. MLatom 3 comes under the open-source permissive MIT license (modified to request proper citations), and the source code is available on open repositories so that, e.g., external developers are encouraged to contribute to the main project and may create their independent, derived, projects. Here, we give an overview of the capabilities of MLatom 3 and provide examples of its applications.

2. OVERVIEW

MLatom merges the functionality from typical quantum chemical and other atomistic simulation packages with the capabilities of desperate ML packages, with a strong focus on molecular systems. The user can choose from a selection of ready-to-use QM and ML models and design and train ML models to perform the required simulations. The bird's view of the MLatom capabilities is best given in Figure 1.

One of the current main goals of MLatom is to enable simulation tasks of interest for computational chemists with generic types of models that can be based on ML, QM, and their combinations (see Section 4). These tasks include single-point calculations, optimization of geometries of minima and transition states (which can be followed by intrinsic reaction coordinate (IRC) analysis[39]), frequency and thermochemical property calculations, molecular and quantum dynamics, rovibrational (infrared (IR) and power) spectra, ML-accelerated UV/vis absorption, and two-photon absorption spectra simulations. This part of MLatom is more similar to traditional

QM and MM packages but with much more flexibility in model choice and unique tasks. A dedicated Section 5 will give a more detailed account of the simulations.

Enabling the users to create their own ML models was MLatom's original main focus, and it continues to play a major role. The MLatom supports a range of carefully selected representative ML algorithms that can learn the desired properties as a function of the 3D atomistic structure. Typically, these algorithms are used, but not limited to, for learning PESs and hence often can be called, for simplicity, ML (interatomic) potentials (MLPs).[40−44] One particular specialization of MLatom is the original implementation of kernel ridge regression (KRR) algorithms for learning any property as a function of any user-provided input vectors or *XYZ* molecular coordinates.[22] In addition, the user can create custom multicomponent models based on concepts of Δ-learning,[23] hierarchical ML,[25] and self-correction.[26] These models may consist of the ML and QM methods. MLatom provides standardized means for training, hyperparameter optimization, and evaluation of the models so that switching from one model type to another may need just one keyword change.[29] This allows one to easily experiment with different models and choose the most appropriate one for the task.

The data are as important as choosing and training the ML algorithms. MLatom 3 provides several data structures specialized for computational chemistry needs, mainly based on versatile Python classes for atoms, molecules, molecular databases, and dynamics trajectories. These classes allow not just storing the data in a clearly structured format but also handling it by, e.g., converting to different molecular representations and data formats and splitting and sampling the data sets into the training, validation, and test subsets. Because data structure is a central concept in the age of data-driven models and MLatom as a package, we describe data structures in Section 3 before describing models, simulations, and machine learning.

How the user interacts with the program is also important, and ideally, the features should be easily accessible and their use
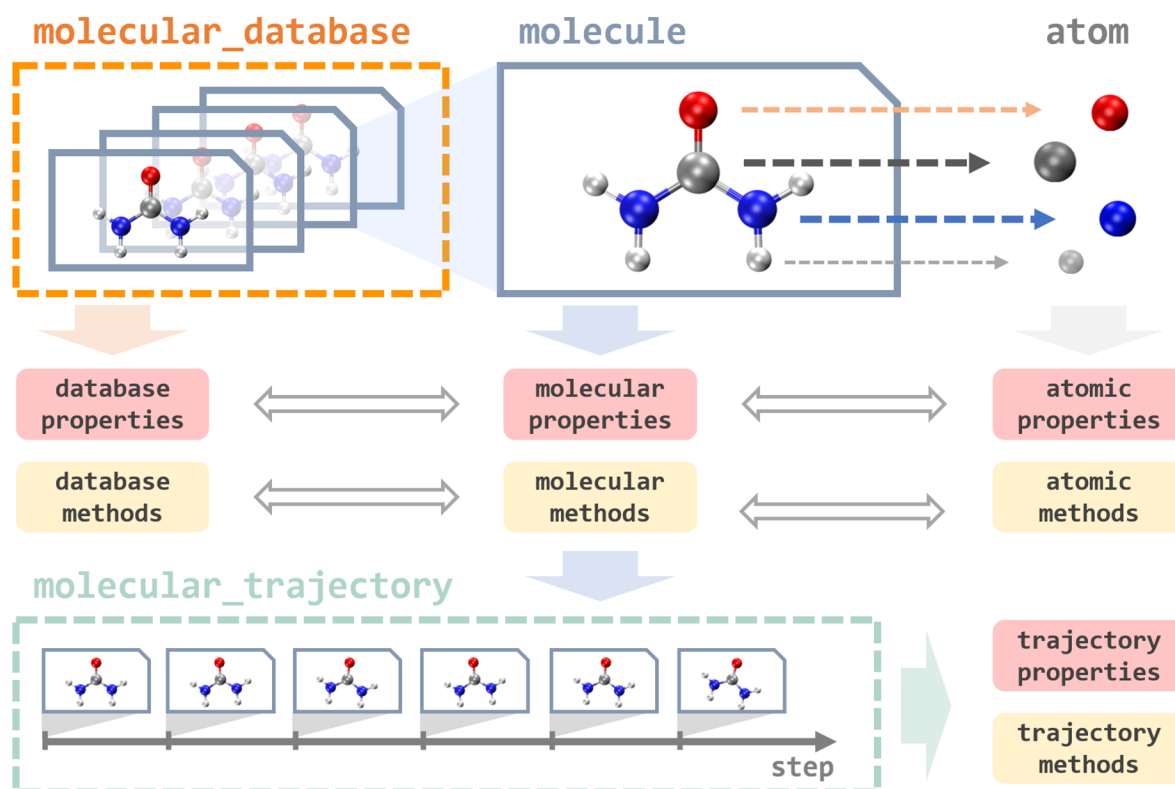
Figure 3. Overview of different data types in MLatom.

intuitive. MLatom calculations can be requested by providing command-line options either directly or through the input file. Alternatively, MLatom can be used as a Python module, which can be imported and used for creating calculation workflows of varying complexity. A side-by-side comparison of these two approaches is given in Figure 2. More examples highlighting different use cases of MLatom are interspersed throughout this article.

MLatom as an open-source package can be conveniently installed via PyPI, i.e., simply using the command `pip install mlatom` or from the source code available on GitHub at https://github.com/dralgroup/mlatom. To additionally facilitate access to AI-enhanced computational chemistry, MLatom can be conveniently used in the XACS cloud computing service at https://XACScloud.com whose basic functionality is free for noncommercial uses such as education and research. Cloud computing eliminates the need for program installation and might be particularly useful for users with limited computational resources.

3. DATA

In MLatom, everything revolves around operations on data: databases and data points of different types, such as an atom, molecule, molecular database, and molecular trajectory (Figure 3). They are implemented as Python classes that contain many useful properties and provide different tools to load and dump these data-type objects using different formats. For example, the key type is a molecule that can be loaded from an *XYZ* file or SMILES and then automatically parsed into the constituent atom objects. Atom objects contain information about the nuclear charge and mass as well as nuclear coordinates. A molecule object is assigned with charge and multiplicity. Information about molecular and atomic properties can be passed to perform simulations, e.g., MD, with models that update and create new molecule objects with calculated quantum mechanical properties such as energies and energy gradients.

See Figure 2 for an example of loading a molecule object `init_mol` from the file `init.xyz`, used as the initial guess for the geometry optimization, returning an optimized geometry as a new molecule object `final_mol`, which is saved into the `opt.xyz` file. Data objects can be directly accessed and manipulated via the MLatom Python API. When using the MLatom in the command-line mode, many similar operations are done under the hood so that the user often just needs to prepare input files in standard formats such as files with *XYZ* coordinates.

Molecule objects can be combined into or created by parsing the molecular database that has functions to split it into the different subsets needed for training and validation of ML models. The databases can be loaded and dumped in plain text (i.e., several files including *XYZ* coordinates, labels, and *XYZ* derivatives), JSON, and npz formats. Another data type is molecular trajectory, which consists of steps containing molecules and other information. Molecular trajectory objects are created during geometry optimization and MD simulations, and in the latter case, the step is a snapshot of MD trajectory, containing information about the time, nuclear coordinates and velocities, atomic numbers and masses, energy gradients, kinetic, potential, and total energies, and, if available, dipole moments and other properties. The trajectories can be loaded and dumped in JSON, H5MD,[45] and plain text formats.

Molecules for which *XYZ* coordinates are provided can be transformed in several supported descriptors: inverse internuclear distances and their version normalized relative to the
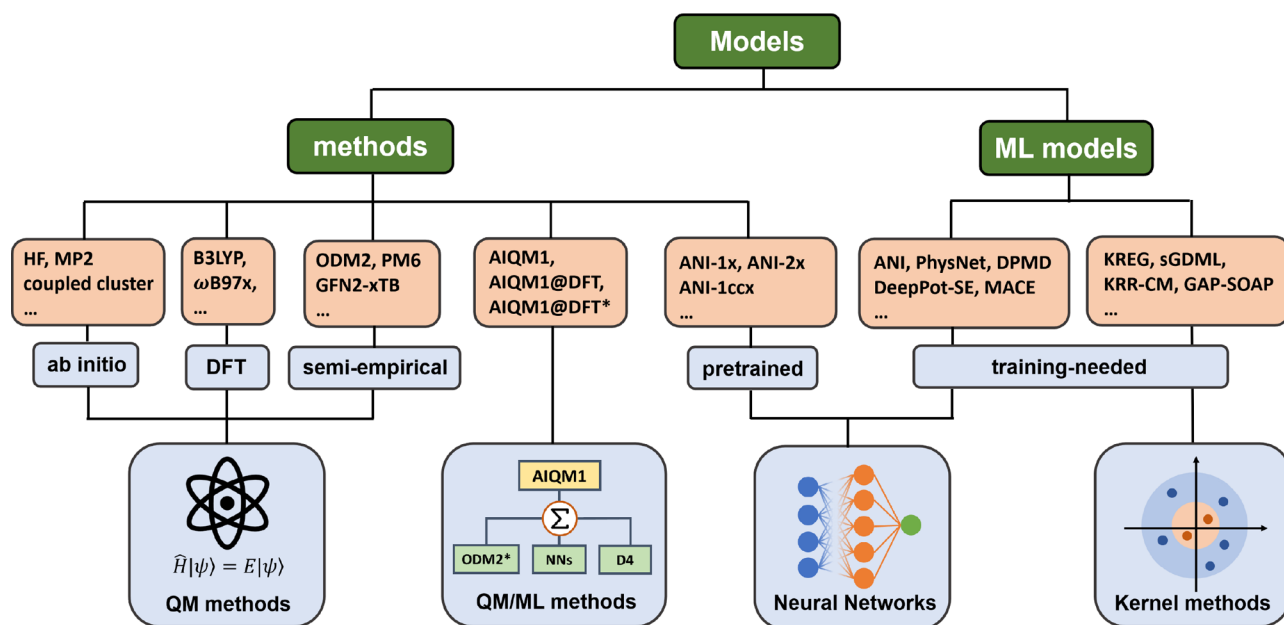
Figure 4. Overview of different model types in MLatom.

Table 1. Overview of Models in MLatom 3 and Their Implementations

model type	model name	implementation
Methods (models that can be used without training)		
QM methods	*ab initio* methods, DFT	interfaces to PySCF[48], Gaussian[48]
	semiempirical OMx[49], DFTB, NDDO-type methods	interfaces to MNDO[50], Sparrow[51]
	semiempirical GFNx-TB[52] methods	interface to xtb[53]
	CCSD(T)*/CBS[34]	interface to Orca[54,55]
QM/ML methods	AIQM1, AIQM1@DFT, AIQM1@DFT*[32]	interfaces to MNDO[50] and Sparrow[51] for the ODM2*[32,49] part, TorchANI[2] for the NN part, dftd4[56] for D4 corrections[57]
pretrained ML models	ANI-1x[33], ANI-2x[35], ANI-1ccx[34]	interface to TorchANI[2]
Models needing training		
neural networks	MACE[58,59]	interface to MACE[60]
	ANI-type[2]	interface to TorchANI[2]
	DPMD[61], DeepPot-SE[62]	interface to DeePMD-kit[63]
	PhysNet[64]	interface to PhysNet[64]
kernel methods	(p)KREG[26,65]	native implementation
	sGDML[66]	interface to sGDML[3]
	KRR-CM[46,47]	native implementation
	GAP[67]-SOAP[68]	interfaces to GAP suite[67] and QUIP[69]

equilibrium structure (RE),[26] Coulomb matrix,[46,47] and their variants.[29]

MLatom also has separate statistics routines to calculate different error measures and perform other data analyses.[29] Routines for preparing common types of plots, such as scatter plots and spectra, are available too.

4. MODELS AND METHODS

Any of the simulations need a model that provides the required output for a given input. The architecture and algorithms behind the models can be designed by an expert or chosen from the available selection. ML models typically require training to find their parameters before they can be used for simulations. Some of these models, such as universal MLPs of the ANI family,[2,33−35] are already pretrained for the user who does not have to train them. This is similar to QM methods, which are commonly used out-of-the-box without tuning their parameters.

In MLatom, we call a method any model that can be used out-of-the-box for simulations. Both pretrained ML models and QM methods belong to the methods in MLatom's terminology, which is reflected in the keyword names. This model type also includes hybrid pretrained ML and QM methods. Below, we overview models available in MLatom when writing this article, the selection of available methods and models with provided architectures that need to be trained, and the ways to design custom models (Figure 4 and Table 1).

4.1. Methods. MLatom provides access to a broad range of methods through interfaces to many third-party, state-of-the-art software packages:

- Pretrained ML models:
  - Universal potentials ANI-1ccx,[34] ANI-1x,[33] ANI-2x,[35] ANI-1x-D4, and ANI-2x-D4. ANI-1ccx is the most accurate and approaches gold-standard CCSD(T) accuracy. We have seen an example of

Table 2. Summary of the Available Kernel Functions for Solving the Kernel Ridge Regression Problem (Eq. 1) as Implemented in MLatom.

Kernel function	Formula	Hyperparameters in kernel function
Linear	$k(\mathbf{x}, \mathbf{x}_j) = \mathbf{x}^T\mathbf{x}_j$	
Gaussian	$k(\mathbf{x}, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2}\sum_s^{N_x}(x_s - x_{j,s})^2\right)$	$\sigma > 0$, length scale
exponential	$k(\mathbf{x}, \mathbf{x}_j) = \exp\left(-\frac{1}{\sigma}[\sum_s^{N_x}(x_s - x_{j,s})^2]^{1/2}\right)$	$\sigma > 0$, length scale
Laplacian	$k(\mathbf{x}, \mathbf{x}_j) = \exp\left(-\frac{1}{\sigma}\sum_s^{N_x}\left|x_s - x_{j,s}\right|\right)$	$\sigma > 0$, length scale
Matérn	$k(\mathbf{x}, \mathbf{x}_j) = \exp\left(-\frac{1}{\sigma}[\sum_s^{N_x}(x_s - x_{j,s})^2]^{\frac{1}{2}}\right) \times \sum_{k=0}^{n}\frac{(n+k)!}{(2n)!}\binom{n}{k}$ $\times \left(\frac{2}{\sigma}[\sum_s^{N_x}(x_s - x_{j,s})^2]^{1/2}\right)^{n-k}$	$\sigma > 0$, length scale; n is a non-negative integer
periodic	$k(\mathbf{x}, \mathbf{x}_j) = \exp\left(-\frac{2}{\sigma^2}\sin^2\left\{\frac{\pi}{p}[\sum_s^{N_x}(x_s - x_{j,s})^2]^{1/2}\right\}\right)$	$\sigma > 0$, length scale; $p > 0$, period
decaying periodic	$k(\mathbf{x}, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2}\sum_s^{N_x}(x_s - x_{j,s})^2 - \frac{2}{\sigma_p^2}\sin^2\left\{\frac{\pi}{p}[\sum_s^{N_x}(x_s - x_{j,s})^2]^{1/2}\right\}\right)$	$\sigma > 0$, length scale; $p > 0$, period; $\sigma_p > 0$, length scale for the periodic term

its use in geometry optimization in Figure 2. Other methods approach the density functional theory (DFT) level. ANI-1ccx and ANI-1x are limited to CHNO elements, while ANI-2x can be used for CHNOFClS elements. We allow the user to use D4 dispersion-corrected universal ANI potentials that might be useful for noncovalent complexes. D4 correction[57] is taken for the ωB97X functional[70] used to generate data for pretraining ANI-1x and ANI-2x. ANI models are provided via an interface to TorchANI[2] and D4 corrections via the interface to dftd4.[56] These methods are limited to predicting energies and forces for neutral closed-shell compounds in their ground state. MLatom reports uncertainties for calculations with these methods based on the standard deviation between neural network (NN) predictions.[36]

- The special ML-TPA model for predicting the two-photon absorption (TPA) cross sections.[31]

- Hybrid QM/ML methods AIQM1, AIQM1@DFT, and AIQM1@DFT*[32] are more transferable and accurate than pretrained ML models but slower (the speed of semiempirical QM methods, which are still much faster than DFT). AIQM1 is approaching gold-standard CCSD(T) accuracy, while AIQM1@DFT and AIQM1@DFT* target the DFT accuracy for neutral, closed-shell molecules in their ground state. All these methods are limited to the CHNO elements. AIQM1 and AIQM1@DFT include explicit D4 dispersion corrections for the ωB97X functional, while AIQM1@DFT* does not. They also include modified ANI-type networks and the modified semiempirical QM method ODM2[49] (ODM2*, provided by either the MNDO[50] or Sparrow[51] program). These methods can also be used to calculate charged species, radicals, excited states, and other QM properties such as dipole moments, charges, oscillator strengths, and nonadiabatic couplings. MLatom reports uncertainties for calculations with these methods based on the standard deviation between NN predictions.[36]

- A range of established QM methods from *ab initio* (e.g., HF, MP2, coupled cluster, *etc.*) to DFT (e.g., B3LYP,[71,72] ωB97X,[70] *etc.*) via interfaces to PySCF[48] and Gaussian.[48]

- A range of semiempirical QM methods (GFN2-xTB,[52] OM2,[73] ODM2,[49] AM1,[74] PM6,[75] *etc.*) via interfaces to the xtb,[53] MNDO,[50] and Sparrow[51] programs.
- A special composite method CCSD(T)*/CBS[34] extrapolating CCSD(T) to the complete basis set via an interface to Orca.[54,55] This method is relatively fast and accurate. It allows the user to check the quality of calculations with other methods and generate robust reference data for ML. This method was used to generate the reference data for AIQM1 and ANI-1ccx.

4.2. Available Standard Models Needing Training. The field of MLPs is very rich in models. Hence, the user can often choose one of the popular MLP architectures reported in the literature rather than developing a new one. MLatom provides a toolset of MLPs from different types (see ref 40 for an overview and ref 29 for implementation details). These supported types can be categorized in a simplified scheme as follows:

- Models based on neural networks (NNs) with fixed local descriptors to which ANI-type MLPs[2] and DPMD[61] belong and with learned local descriptors represented by PhysNet[64] and DeepPot-SE.[62] MLatom also supports a representative equivariant NN MACE, which shows superior performance for many tasks.[58,59]
- Models based on kernel methods (KMs)[76] with global descriptors to which (p)KREG,[26,65] sGDML,[66] and KRR-CM[46,47] belong as well as with local descriptors represented by only GAP[67]-SOAP.[68]

Any of these models can be trained and used for simulations, e.g., geometry optimizations or dynamics. MLatom also supports hyperparameter optimization with many algorithms including grid search,[22] Bayesian optimization via the hyperopt package,[77,78] and standard optimization algorithms available in SciPy.[79] Generalization errors of the resulting models can also be evaluated in standard ways (hold-out and cross-validation). More on this is available in a dedicated Section 6.

4.3. Custom Models Based on Kernel Methods. MLatom also provides the flexibility of training custom models based on kernel ridge regression (KRR) for a given set of input vectors **x** or *XYZ* coordinates and any labels **y**.[80,81] If *XYZ* coordinates are provided, they can be transformed in one of the several supported descriptors (e.g., inverse internuclear distances and their version normalized relative to the equilibrium structure (RE) and the Coulomb matrix). The user can choose from one of the implemented kernel functions,
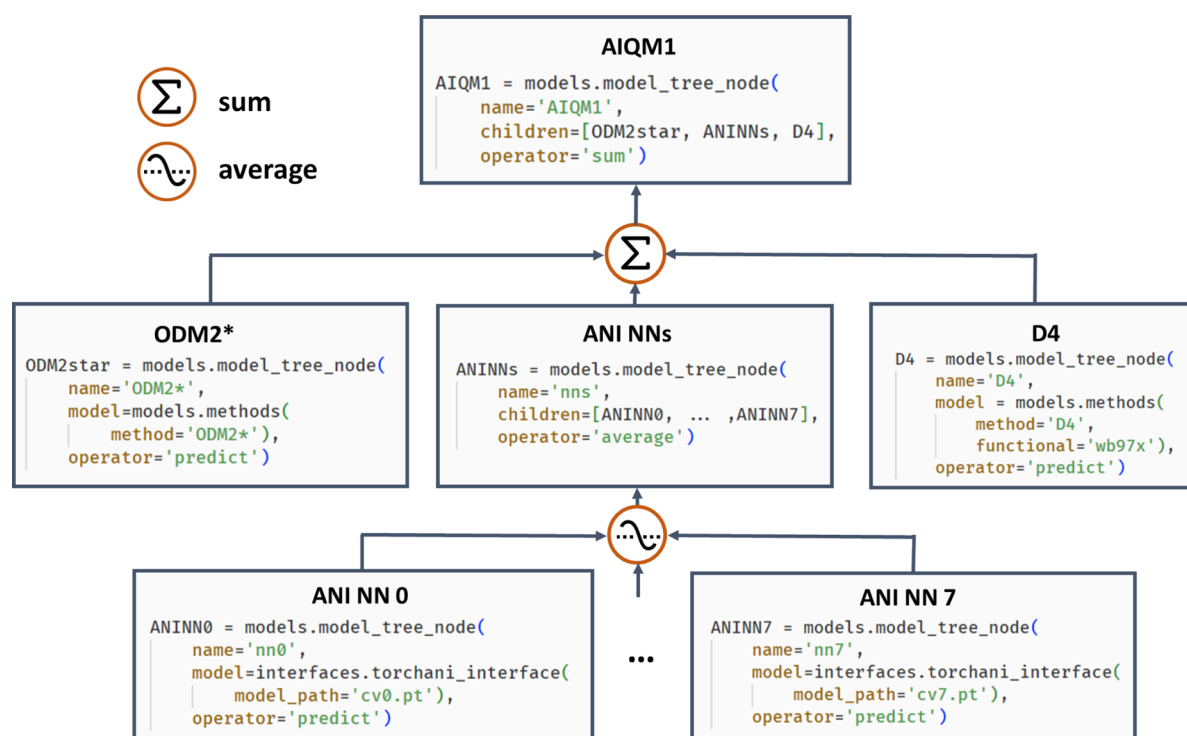
Figure 5. Composite models can be constructed as a model tree in MLatom. Here, an example is shown for the AIQM1 method where the root parent node comprises 3 children, the semiempirical QM method ODM2*, the NN ensemble, and additional D4 dispersion correction. The NN ensemble in turn is a parent of 8 ANI-type NN children. Predictions of parents are obtained by applying an operation "average" or "sum" to children's predictions. The code snippets are shown, too.

including the linear,[22,81,82] Gaussian,[22,81,82] exponential,[22,81,82] Laplacian,[22,81,82] and Matérn[22,81−83] as well as periodic[82,84,85] and decaying periodic[82,84,86] functions, which are summarized in Table 2. These kernel functions $k(\mathbf{x}, \mathbf{x}_j; \mathbf{h})$ are key components required to solve the KRR problem of finding the regression coefficients α of the approximating function $\hat{f}(\mathbf{x}; \mathbf{h})$ of the input vector $\mathbf{x}$:[80,81]

$$\hat{f}(\mathbf{x}; \mathbf{h}) = \sum_{j=1}^{N_{tr}} \alpha_j k(\mathbf{x}, \mathbf{x}_j; \mathbf{h}) \tag{1}$$

The kernel function, in most cases, has hyperparameters $\mathbf{h}$ to tune, and they can be viewed as measuring similarity between the input vector $\mathbf{x}$ and all of the N_{tr} training points $\mathbf{x}_j$ (both vectors should be of the same length N_x). In addition to the hyperparameters in the kernel function, all KRR models have at least one more regularization parameter, λ, used during training to improve the generalizability.

4.4. Composite Models. Often, it is beneficial to combine several models. One example of such composite models is based on Δ-learning[23] where the low-level QM method is used as a baseline, which is corrected by an ML model to approach the accuracy of the target higher-level QM method. Another example is ensemble learning[87] where multiple ML models are created, and their predictions are averaged during the simulations to obtain more robust results and use in the query-by-committee strategy of active learning.[88] Both of these concepts can also be combined in more complex workflows as exemplified by the AIQM1 method,[32] which uses the NN ensemble as a correcting Δ-learning model and the semi-empirical QM method as the baseline. To easily implement these workflows, MLatom allows the construction of the

composite models as model trees; see an example of AIQM1 in Figure 5.

Other examples of possible composite models are hierarchical ML,[25] which combines several (correcting) ML models trained on (differences between) QM levels, and self-correction,[26] when each next ML model corrects the prediction by the previous model.

5. SIMULATIONS

MLatom supports a range of simulation tasks such as single-point simulations, geometry optimizations, frequency and thermochemistry calculations, molecular and quantum dynamics, one- and two-photon absorption, and (ro)vibrational spectra simulations (Figure 1). Most of them need any model that can provide energies and energy derivatives (gradients and Hessians).

5.1. Single-Point Calculations. Single-point calculations are calculations of quantum mechanical properties—mainly energies and energy gradients, but also Hessians, charges, dipole moments, *etc.*—for a single geometry. These calculations are very common in ML research in computational chemistry as they are used both to generate the reference data with QM methods for training and validating ML and to make inferences with ML to validate the trained model and generate required data for new geometries. MLatom is a convenient tool to perform single-point calculations not just for a single geometry, as in many QM packages, but for data sets with many geometries.

5.2. Geometry Optimizations. Locating stationary points on the PES, such as energy minima and transition states, is crucial for understanding the molecular structure and reactivity. Hence, geometry optimizations are among the most important
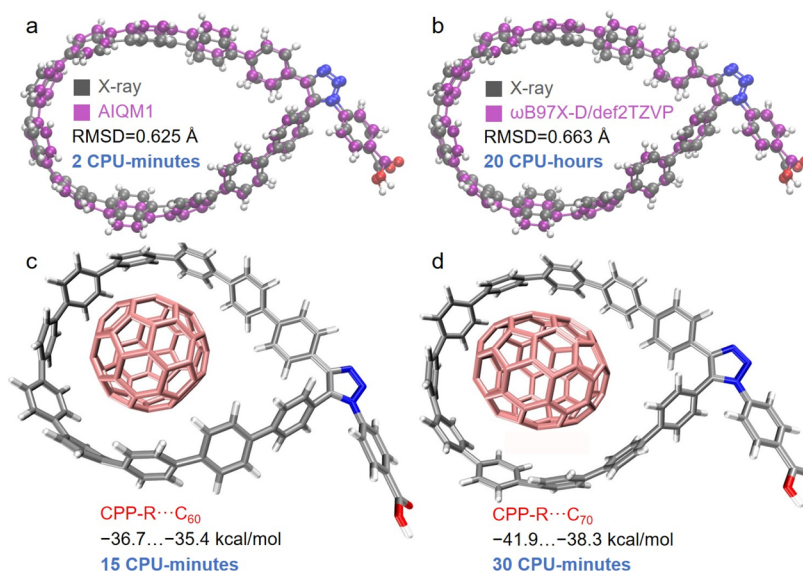
Figure 6. X-ray structure of the functionalized cycloparaphenylene (CPP) nanolasso superimposed with the structure optimized in vacuum at (a) AIQM1 and (b) ωB97X-D/def2-TZVP. Complexes of functionalized CPP and (c) C_{60} and (d) C_{70} with binding energies in kcal/mol calculated at AIQM1 in vacuum. The CPU time for these calculations is also reported.

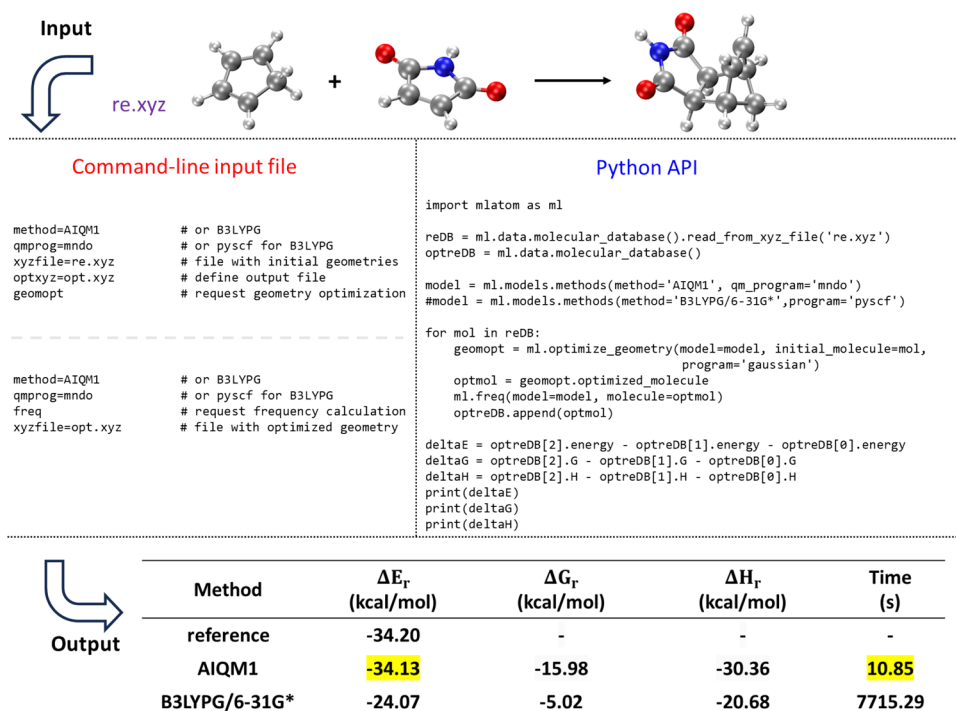| Method	ΔE_r (kcal/mol)	ΔG_r (kcal/mol)	ΔH_r (kcal/mol)	Time (s)
reference	-34.20	-	-	-
AIQM1	-34.13	-15.98	-30.36	10.85
B3LYPG/6-31G*	-24.07	-5.02	-20.68	7715.29

Figure 7. Calculations of ZPVE-exclusive energy, Gibbs free energy, and enthalpy changes in the Diels−Alder reaction of cyclopentadiene and maleimide forming the corresponding endo product with AIQM1 and B3LYPG/6-31G* (from the interface to PySCF; "G" in B3LYPG means that we use the B3LYP variant according to the Gaussian program convention). The reference reaction energy is from the GMTKN55 set.[93]

and frequent tasks in computational chemistry. MLatom can locate energy minima and transition states (TS) with any model providing energies and gradients. An example of geometry optimization is given in Figure 2. A practical application of MLatom for efficient and accurate geometry optimization was performed previously for rather large cycloparaphenylene (CPP) nanolassos and their complexes with fullerene molecules (systems with up to 200 atoms, Figure 6).[89] The AIQM1 method can provide an optimized functionalized CPP structure, which has better agreement with the X-ray structure than that

obtained from the DFT method at a speed 600 times faster than the DFT method. In our laboratories, we also use the AIQM1 method to optimize systems with more than a thousand of atoms on a single CPU, while for more computationally intensive tasks such as dynamics of large systems, one can use the pretrained ANI methods. Hessians are also required for the Berny TS optimization algorithm. Once the TS is located, the user can follow the intrinsic reaction coordinate (IRC)[39] to check its nature. Geometry optimizations can be performed with many algorithms provided by the interfaces to SciPy,[79] ASE,[18] or

Gaussian.[48] TS search can be performed with the dimer method[90] in ASE and the Berny algorithm[91] in Gaussian. IRC calculations can only be performed with the interface to Gaussian.

The seamless integration of the variety of QM and ML methods for performing geometry optimizations is advantageous because it allows the use of methods from interfaced programs that do not implement some of these simulation tasks by themselves. For example, MLatom can be used to perform TS search with the GFN2-xTB method via an interface to the xtb program, while there is no option for TS search with the latter program. Similarly, Sparrow, which provides access to many semiempirical methods, can only be used for single-point calculations. Since analytical gradients and Hessians are not available for many models and implementations, MLatom also implements a finite-difference numerical differentiation, further expanding the applicability of the models for geometry optimizations.

5.3. Frequency Calculations. Simulation of vibrational frequencies is another common and important task in computational chemistry as it is useful to additionally verify the nature of stationary points, visualize molecular vibrations, calculate zero-point vibrational energy (ZPE) and thermochemical properties, and obtain spectroscopic information, which can be compared to experimental vibrational spectra. These calculations can be performed within the ridge-rotor harmonic approximation via an adapted TorchANI implementation[2] and Gaussian[48] interface. The latter also allows the calculation of anharmonic frequencies using the second-order perturbative approach.[92]

Similarly to geometry optimizations, MLatom can perform these simulations with any model—ML and QM or their combination—that provides energies. Calculations also need Hessian, and wherever available, analytical Hessian is used. If it is unavailable, semianalytical (with analytical gradients) or fully numerical Hessian can be calculated.

5.4. Relative Energy Calculations. Relative energy is crucial for understanding and predicting various aspects of chemical behavior, from kinetics to thermodynamics, e.g., via calculating reaction energies, barrier heights, isomerization energies, and molecular stabilities. MLatom can produce various types of energies for molecules such as ZPE-exclusive and inclusive total energies, enthalpies, entropies, Gibbs free energies, and internal energies. Hence, the package can readily be used to evaluate different types of relative energies, e.g., the reaction enthalpies and Gibbs free energies as shown for investigating which fullerene molecules bind stronger to the cycloparaphenylene nanolassos (Figure 6) and for the Diels–Alder reaction of cyclopentadiene and maleimide (Figure 7).

5.4.1. Calculation of Heats of Formation. The special type of relative energy calculation is evaluation of heats (enthalpies) of formation. MLatom uses the scheme analogous to those employed in the *ab initio*[94] and semiempirical QM calculations[49] to derive heats of formation:

$$\Delta H_{\mathrm{f},T} = \left[\sum_A \Delta H_{\mathrm{f},T}(A) \right] - \Delta H_{\mathrm{at},T} \tag{2}$$

where $\Delta H_{\mathrm{f},T}(A)$ is the experimental enthalpies of formation of the free atom A and $\Delta H_{\mathrm{at},T}$ is the atomization enthalpy. In AIQM1 and ANI-1ccx, we use the same $\Delta H_{\mathrm{f},T}(A)$ values as other semiempirical QM methods, i.e., 52.102, 170.89, 113.00,

and 59.559 kcal/mol for elements H, C, N, and O, respectively.[50]

The atomization enthalpy $\Delta H_{\mathrm{at},T}$ can be obtained from the difference between molecular H_T and atomic absolute enthalpies $H_T(A)$:

$$\Delta H_{\mathrm{at},T} = \left[\sum_A H_T(A) \right] - H_T \tag{3}$$

Analogous to *ab initio* methods, harmonic-oscillator and rigid-rotor approximations are explicitly considered in the calculation of absolute enthalpies:

$$H_T = E_{\mathrm{tot}} + \mathrm{ZPVE} + E_{\mathrm{trans},T} + E_{\mathrm{rot},T} + E_{\mathrm{vib},T} + RT \tag{4}$$

$$H_T(A) = E(A) + E_{\mathrm{trans},T}(A) + RT \tag{5}$$

where E_{tot} and $E(A)$ are the total energy of the molecule and free atom, respectively, and ZPVE is the zero-point vibrational energy. $E_{\mathrm{trans},T}$, $E_{\mathrm{rot},T}$, and $E_{\mathrm{vib},T}$ are the translational, rotational, and vibrational thermal contributions, respectively, and R is the gas constant.

The scheme requires knowledge of the free atom energies $E(A)$. Any model able to calculate them can be used for predicting heats of formation. This is straightforward for QM methods and also possible for ML models if the energies of isolated atoms were included in the training data. However, if the ML-based models are trained only on molecular species, as is commonly done, they cannot be expected to produce reasonable heats of formation. In the case of the pretrained models supported by MLatom, we have previously fitted free atom energies (see Table 3) for AIQM1 and ANI-1ccx methods to

Table 3. Atomic Energies (in hartree) of AIQM1 and ANI-1ccx Used in Heat of Formation Calculations[32,36]

element	AIQM1	ANI-1ccx
H	−0.50088038	−0.50088088
C	−37.79221710	−37.79199048
N	−54.53360298	−54.53379230
O	−75.00986203	−75.00968205

reproduce experimental heats of formation for a set of common molecules because the NNs in these methods were not trained on an isolated atom.[32,36] As a result, both methods can provide heats of formation close to chemical accuracy with speed orders of magnitude higher than those of alternative, high-accuracy QM methods. In addition, we provide an uncertainty quantification scheme based on the deviation of NN predictions in these methods to tell the users when the predictions are confident. This was useful to find errors in the experimental data set of heats of formation.[36]

An example of using MLatom to calculate the heats of formation with the AIQM1 and B3LYP/6-31G* methods is shown in Figure 8. AIQM1 is both faster and more accurate than B3LYP, as can be seen by comparing the values with the experiment. This is also consistent with our previous benchmark.[36]

5.5. Molecular Dynamics. Molecular dynamics propagates nuclear motion based on the equation of motion according to the classical mechanics.[96] This requires knowledge of forces acting on nuclei, which are typically derived as the negative of the potential energy gradients (i.e., negative of the derivatives of
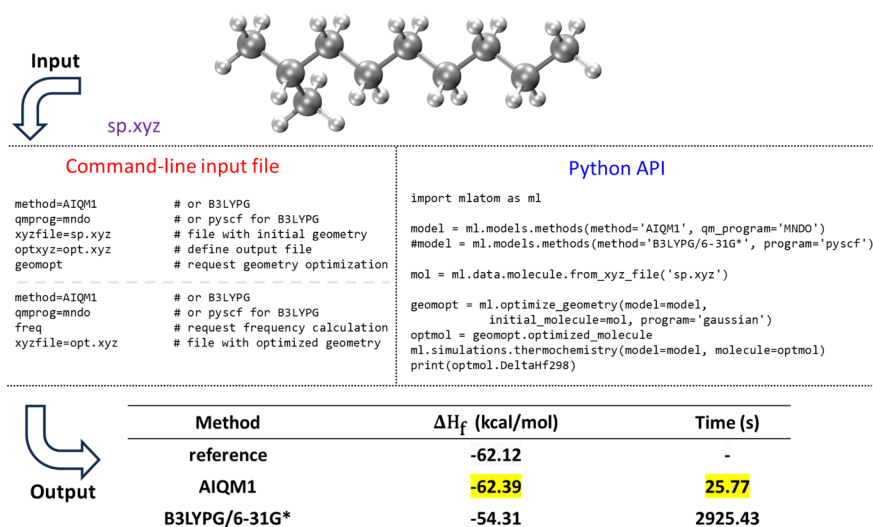
Figure 8. Calculation of heats of formation of 2-methylnonane with AIQM1 and B3LYPG/6-31G* (from the interface to PySCF; "G" in B3LYPG means that we use the B3LYP variant according to the Gaussian program convention) compared to the experiment.[95]

the model for potential energies) for conservative forces. Due to the high cost of the approach, it is most commonly used with molecular mechanics force fields,[97] but often, calculations based on QM methods are possible in variants called *ab initio* or Born–Oppenheimer MD (BOMD).[96] The proliferation of ML potentials makes it possible to perform BOMD-quality dynamics at a cost comparable to molecular mechanics force fields or much faster than commonly used DFT-based BOMD,[40–44] which allows routine simulations of large systems such as a quadruple assembly of octatetrayne-bridged *ortho*-perylene diimide dyads with ca. 400 atoms[98] at ANI-1ccx (Figure 9). The accuracy of such simulations can be also high; for example, the IR spectra obtained from the MD with AIQM1 method are more accurate than those from a much slower DFT MD (Figure 10).[99]

MLatom has a native implementation of MD supporting any kind of model that provides forces, not necessarily conservative.[99] Currently, simulations in NVE and NVT ensembles,[101] based on the velocity Verlet algorithm,[102] are possible. NVT simulations can be carried out with the Andersen[101,103] and Nosé–Hoover[104,105] thermostats, and the implementation of other thermostats is expected to be available in the future. Trajectories can be saved in different formats, including plain text, JSON, and more compact H5MD[29] database formats. The Nosé–Hoover thermostat is a deterministic thermostat that couples the system to a thermal bath through extra terms in the Hamiltonian. Its theory and implementation details are described elsewhere.[99] Here, we briefly mention the relevant methodology[101,103] used in the Andersen thermostat. In this thermostat, the system is coupled to a heat bath by stochastically changing the velocity of each atom. The changing frequency (or collision frequency) is controlled by the tunable parameter v. The collisions follow the Poisson distribution, so that the probability of changing the velocity of each atom during a time step Δt is $v\Delta t$. If the atoms collide, new velocities will be assigned to them, sampled from a Maxwell–Boltzmann distribution at target temperature T.

Multiple independent MD trajectories can be propagated in parallel, dramatically speeding up the calculations. In addition, we made an effort to better integrate the KREG model

implemented in Fortran into the main Python-based MLatom code, which makes MD with KREG very efficient.

Note that MD can also be propagated without forces using the concept of the 4D-spacetime AI atomistic models, which directly predict nuclear configurations as a function of time.[85] Our realization of this concept, called the GICnet model, is currently available in a publicly available development version of MLatom version.[85]

The above implementations can propagate MD on an adiabatic potential energy surface, i.e., typically for ground-state dynamics. Nonadiabatic MD based on the trajectory surface hopping algorithms can also be performed with the help of MLatom, currently, via Newton-X's[106] interface to MLatom.[27,107,108] MLatom also supports quantum dissipative dynamics, as described in the next section.

5.6. Quantum Dissipative Dynamics. It is often necessary and beneficial to treat the entire system quantum mechanically and also include the environmental effects.[109] This is possible via many quantum dissipative dynamics (QD) algorithms, and an increasing number of ML techniques were suggested to accelerate such simulations.[107] MLatom allows performing several unique ML-accelerated QD simulations using either a recursive scheme based on KRR[110] or a conceptually different AI-QD approach[38] predicting the trajectories as a function of time or the OSTL technique[111] outputting the entire trajectories in one shot. These approaches are enabled via an interface to a specialized program MLQD.[112]

In the recursive KRR scheme, a KRR model is trained, establishing a map between the future and past dynamics. This KRR model, when provided with a brief snapshot of the current dynamics, can be leveraged to forecast future dynamics. In the AI-QD approach, a convolution neural network (CNN) model is trained mapping simulation parameters and time to the corresponding system's state. Using the trained CNN model, the state of the system can be predicted at any time without the need to explicitly simulate the dynamics. Similarly, the ultrafast OSTL method utilizes a CNN-based architecture and, based on simulation parameters, predicts future dynamics of the system's state up to a predefined time in a single shot. In addition, as optimization is a key component in training, users can optimize both KRR and CNN models using MLatom's grid search
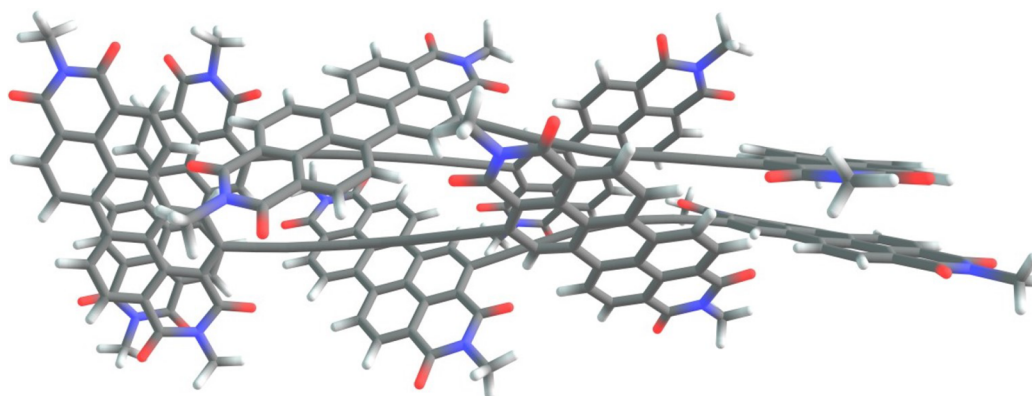
Command-line input file

```
MD
method=ANI-1ccx
initConditions=random
initTemperature=333
initXYZ=init.xyz
thermostat=nose-hoover
temperature=333
dt=0.5
trun=20000
trajh5mdout=traj.h5
```

Python API

```python
import mlatom as ml
mol = ml.data.molecule()
mol.read_from_xyz_file(filename='init.xyz')
method = ml.models.methods(method='ANI-1ccx')
init_cond_db = ml.generate_initial_conditions(
    molecule=mol,
    generation_method='random',
    initial_temperature=333)
init_mol = init_cond_db.molecules[0]
nose_hoover = ml.md.Nose_Hoover_thermostat(
    temperature=333,
    molecule=init_mol)
dyn = ml.md(
    model=method,
    molecule_with_initial_conditions=init_mol,
    ensemble='NVT',
    thermostat=nose_hoover,
    time_step=0.5,
    maximum_propagation_time=20000.0)
traj = dyn.molecular_trajectory
traj.dump(filename='traj.h5', format='h5md')
```
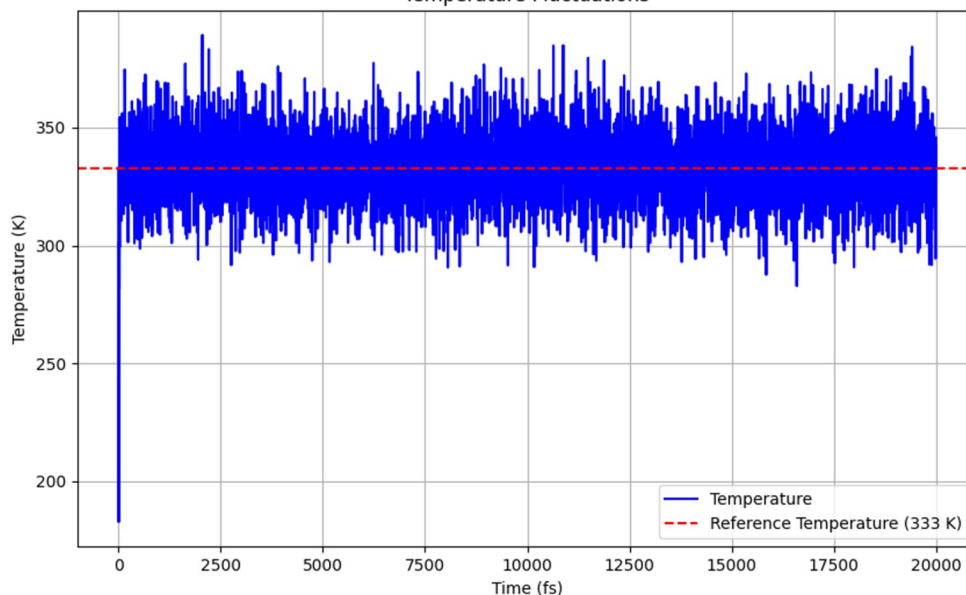

Figure 9. Structure of the $(POP)_4$ complex,[98] a quadruple assembly of octatetrayne-bridged *ortho*-perylene diimide dyads. The command-line input file and the Python script used for NVT MD propagation with the ANI-1ccx method for this molecule are provided. The evolution of the temperature over time during NVT MD is also shown.

functionality for KRR and Bayesian optimization via the hyperopt[77] library for CNN. Moreover, we also incorporate

the autoplotting functionality, where the predicted dynamics is plotted against the provided reference trajectory.
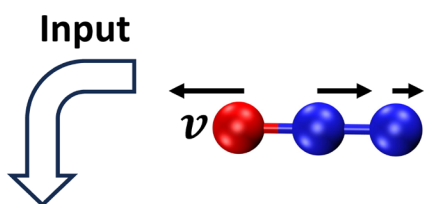
Input

init.xyz

```
3
N  -0.46805134   0.0   0.0
O  -2.76912975   0.0   0.0
N  -1.59908605   0.0   0.0
```

init.vxyz

```
3
 0.00258547   0.0   0.0
-0.0062165    0.0   0.0
 0.00451528   0.0   0.0
```

Command-line input file

```
MD
method=AIQM1
method=PBE/def2-SVP
prog=Gaussian
initXYZ=init.xyz
initVXYZ=init.vxyz
dt=0.5
trun=23000
trajh5mdout=traj.h5
```

Python API

```
import mlatom as ml
method = ml.models.methods(method='AIQM1')
method = ml.models.methods(
    method='PBE/def2-SVP',program='Gaussian')
mol = ml.data.molecule.from_xyz_file('init.xyz')
init_cond_db = ml.generate_initial_conditions(
    molecule=mol,
    generation_method='user-defined',
    file_with_initial_xyz_coordinates='init.xyz',
    file_with_initial_xyz_velocities='init.vxyz')
init_mol = init_cond_db.molecules[0]
dyn = ml.md(
    model=method,
    molecule_with_initial_conditions=init_mol,
    ensemble='NVE',
    time_step=0.5,maximum_propagation=23000)
traj=dyn.molecular_trajectory
traj.dump(filename='traj.h5',format='h5md')
```

Molecular trajectory in H5MD format: traj.h5

Command-line input file

```
MD2vibr
trajH5MDin=traj.h5
dt=0.5
start_time=3000
end_time=23000
output=ir
```

Python API

```
import mlatom as ml
traj = ml.data.molecular_trajectory()
traj.load(filename='traj.h5',format='h5md')
moldb = ml.data.molecular_database()
moldb.molecules = [step.molecule for step in traj.steps]
md2vibr = ml.vibrational_spectrum(
    molecular_database=moldb,dt=0.5)
md2vibr.plot_infrared_spectrum(
    filename='ir.png',lb=3000,ub=23000)
```
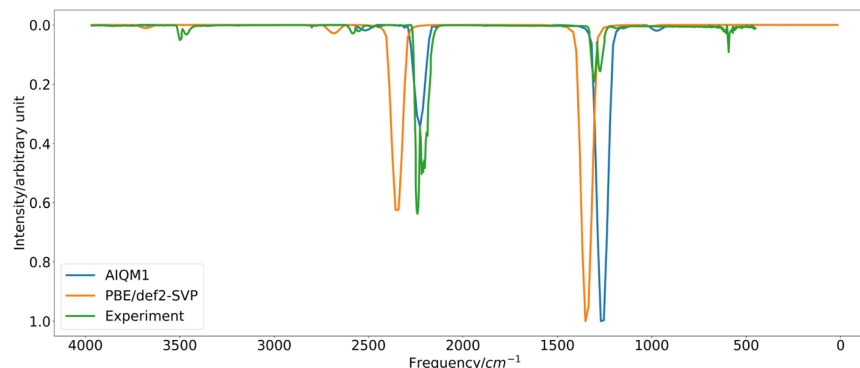
Output

Figure 10. Propagation of MD with AIQM1 and PBE/def2-SVP (from the interface to Gaussian) and the IR spectra of the N_2O molecule derived from trajectories. MLatom generates spectra for each method; here, the results are collated and shown together with the experimental spectrum[100] for comparison.
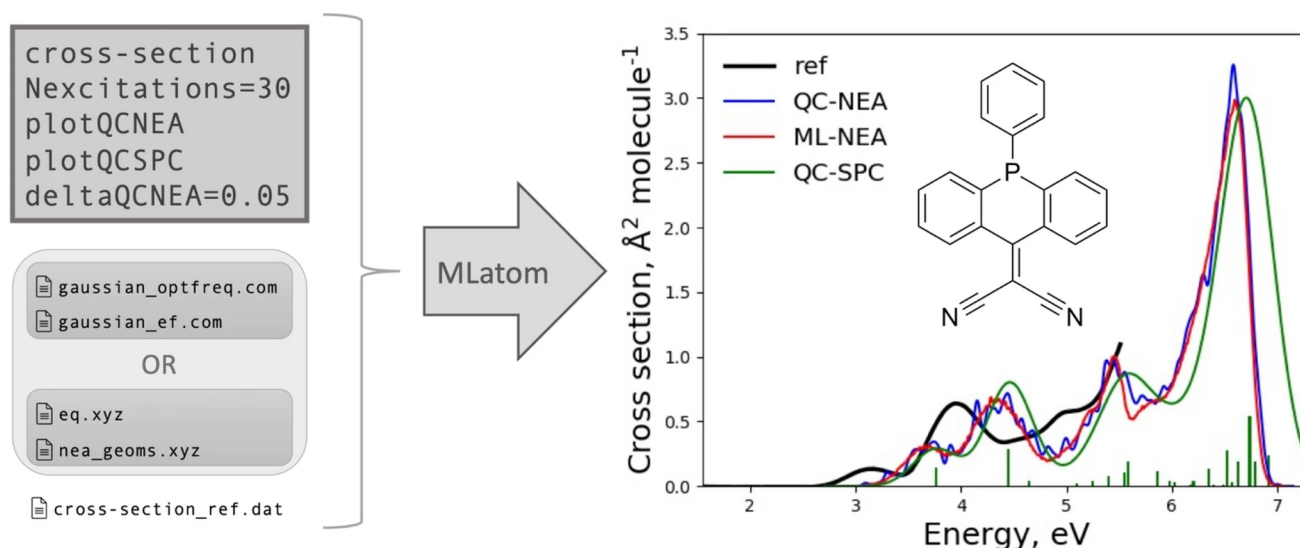
Figure 11. Using MLatom to predict the UV/vis absorption spectra of the acridophosphine derivative molecule with the ML-NEA[30] method. The MLatom input file and the list of additional required files are shown on the left. The cross section predicted by ML-NEA shown on the right is compared to traditional QC-NEA and the single-point convolution approach (QC-SPC). This figure is adapted from ref 29. Copyright 2021, the Authors.
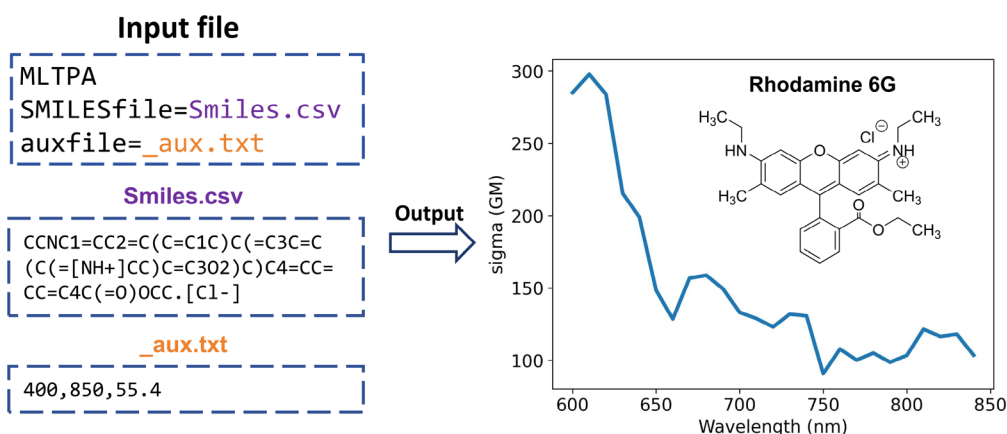

Figure 12. Using MLatom to predict the two-photon absorption cross section of the Rhodamine 6G molecule with the ML-TPA approach. The MLatom command-line input file and additional files are shown on the left. The cross section predicted by ML-TPA is shown on the right.

5.7. Rovibrational (Infrared and Power) Spectra.
Rovibrational spectra can be calculated in several ways with MLatom. The simplest method is by performing frequency calculations on an optimized molecular geometry. This requires any model providing Hessians and, preferably, dipole moments. Another one is performing molecular dynamics simulations with any model providing energy gradients and, then, postprocessing the trajectories.

Both frequency calculations and the MD-based approach require the model to also provide dipole moments to calculate the absorption intensities. If no dipole moments are provided, only frequencies are available, or, in the case of MD, only power spectra rather than IR can be obtained. The IR spectra are obtained via the fast Fourier transform using the autocorrelation function of dipole moment[113,114] with our own implementation.[99] The power spectra only need the fast Fourier transform,[113] which is also implemented[85] in MLatom.

We have previously shown[99] that the high quality of the AIQM1 method results in rather accurate IR spectra obtained from MD simulations compared to spectra obtained with a representative DFT (which is also substantially slower; see example in Figure 10) or a semiempirical QM method.

5.8. One-Photon UV/Vis Absorption Spectra. UV/vis absorption spectra simulations are computationally intensive because they require calculation of excited-state properties. In addition, better-quality spectra can be obtained via the nuclear ensemble approach (NEA),[115] which necessitates the calculation of excited-state properties for thousands of geometries for high precision. MLatom implements an interpolation ML-NEA scheme[30] that improves the precision of the spectra with a fraction of the computational cost of traditional NEA simulations (Figure 11). Currently, the ML-NEA calculations are based on interfaces to Newton-X[106] and Gaussian[48] and utilize the sampling of geometries from a harmonic Wigner distribution.[116] This scheme also automatically determines the optimal number of required reference calculations, providing a user-friendly, black-box implementation of the algorithm.[29]

5.9. Two-Photon Absorption. Beyond one-photon absorption, MLatom has an implementation of a unique ML approach for calculating two-photon absorption (TPA) cross sections of molecules just based on their SMILES strings,[45]
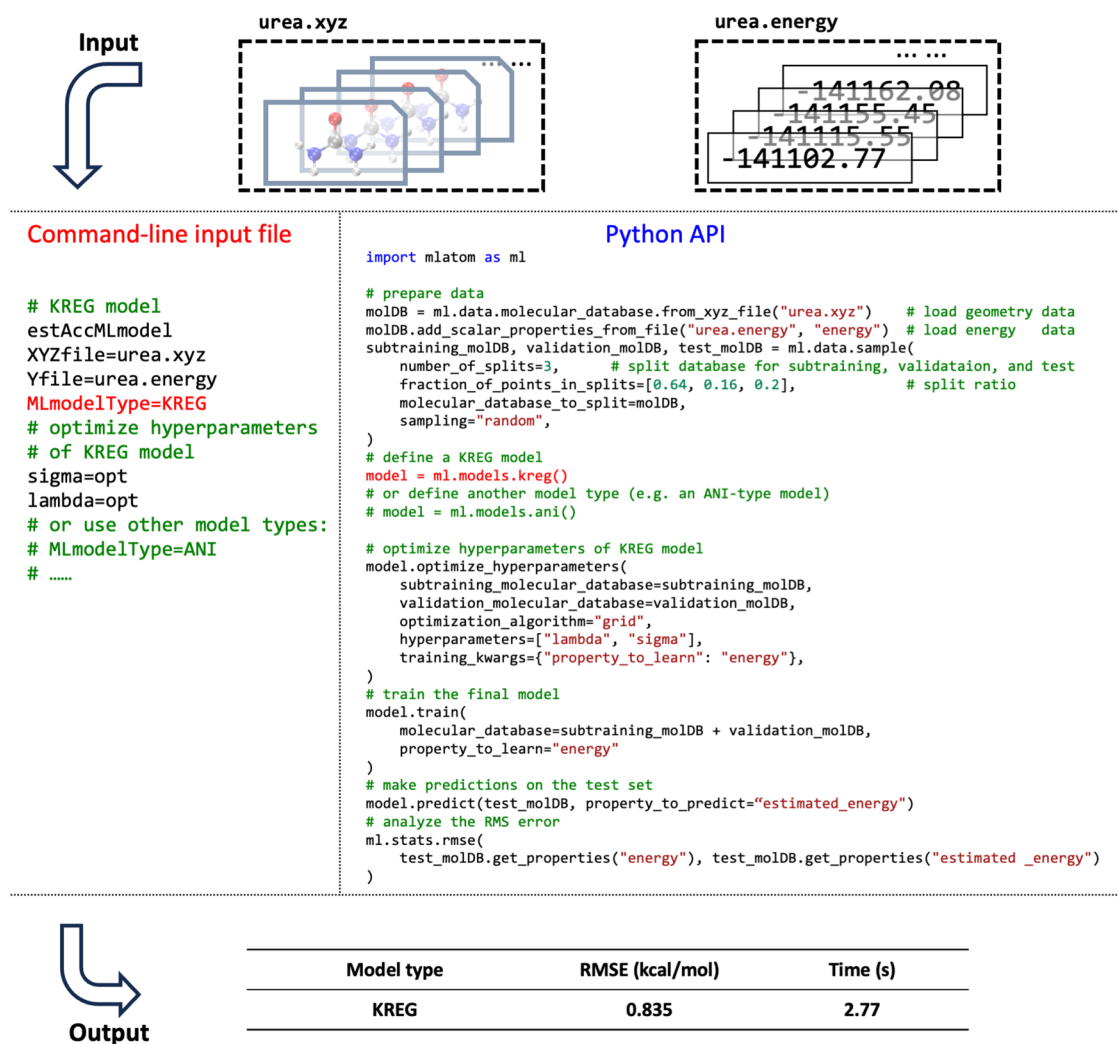
Figure 13. Side-by-side comparison of the usage of MLatom in both the command-line mode and via Python API for training and testing the KREG model on a 1000-point data set on the urea molecular PES data set randomly sampled from the WS22 database.[121] Hyperparameter optimization of the KREG model required is also shown. Calculations were run on a 36 Intel(R) Xeon(R) Gold 6240 CPU @ 2.60 GHz.

which are converted into the required descriptors using the interface to RDKit,[117] and solvent information.[31] This ML-TPA approach is very fast with accuracy comparable to that of much more computationally intensive QM methods. We provide an ML model pretrained on experimental data. ML-TPA was tested in real laboratory settings and was shown to provide a good estimate for new molecules not present in the training experimental database. An example of using ML-TPA to predict two-photon absorption is shown in Figure 12.

6. MACHINE LEARNING

In Sections 4 and 5, we discussed the supported types of models and how they can be applied to simulations. Here, we briefly overview the general considerations for training and validating the ML models with MLatom. The models share the standard MLatom's conventions for input, output, training, hyperparameter optimization, and testing, which allows one to conveniently switch from one model to another and benchmark them.

6.1. Training. To create an ML model, the user has to choose and train the ML model and prepare data. MLatom provides many tools for the different stages of this process. The model can be either chosen from a selection of provided types of ML

models with a predefined architecture or customized based on available algorithms and preset models. Once a model is chosen, it must be trained, and, in many cases, it is advisable or even required (particularly in the case of the kernel methods) to optimize its hyperparameters, which can be done as explained in Section 6.2.

For training, the data set should be appropriately prepared. MLatom has strict naming conventions for data set splits to avoid any confusion when changing and comparing different model types. All of the data that are used directly or indirectly for creating an ML model are called the training set. This means that the validation set, which can be used for hyperparameter optimization or early stopping during NN training, is a subset of the training set. Thus, the part of the training set remaining after excluding the validation set is called the subtraining set and is actually used for training the model, i.e., optimizing model parameters (weights in NN terminology and regression coefficients in kernel method terminology).

MLatom can split the training data set into the subtraining and validation data subsets or create a collection of these subsets via cross-validation.[24,29] The sampling into the subsets can be performed randomly or using furthest-point or structure-based sampling.

In the case of kernel methods, the final model in MLatom is typically trained on the entire training set after hyperparameter optimization. This is possible because the kernel methods have a closed analytical solution to finding their regression coefficients, and after hyperparameters are appropriately chosen, overfitting can be mitigated to a great extent. In the case of NNs, the final model is the one trained on the subtraining set because it would be too dangerous to train on the entire training set without any validation subset to check for the signs of overfitting.

6.1.1. Training Predefined Types of ML Models. Most predefined types of ML models, such as ANI-type or KREG models, expect *XYZ* molecular coordinates as input. This should be either provided by the user or can be obtained using MLatom's conversion routines, e.g., from the SMILES strings,[118] which rely on OpenBabel's[119] Pybel API. These models have a default set of hyperparameters, but, especially in the case of kernel methods such as KREG, it is still strongly advised to optimize them. The models can be, in principle, trained on any molecular property. Most often, they are used to learn PESs and hence require energy labels in the training set. The PES model accuracy can be greatly improved if the energy gradients are also provided for training. Thus, the increased training time is usually justified.[40,120] An example of training and testing the KREG model on a data set with energies and energy gradients for the urea molecule in the WS22 database[121] is shown in Figure 13. The KREG model is both fast to train and accurate (achieved an RMSE below 1 kcal/mol within a few seconds), which is a typical situation for small-size molecular databases, while for larger databases, NN-based models might be preferable.[40] Command-line and Python script inputs for using a different type of ML model (e.g., ANI-type[2]) are also shown in the figure as comments.

6.1.2. Designing and Training Custom ML Models. MLatom's user can also create models on any set of input vectors and labels using a variety of KRR kernel functions. In this case, hyperparameter optimization is strongly advised too. In all other aspects, training of such KRR models is similar to training the predefined models, i.e., the preparation of the data set is also performed by splitting it into the required subsets for training and validation.

Importantly, the user can construct models of varying complexity using a model tree implementation. Special cases of such composite models are Δ-learning and self-correcting models, and they can be trained similarly to other ML models by supplying input vectors or *XYZ* coordinates and labels. In the case of Δ-learning, the user must supply the baseline values. For other more complicated models, the user must train and combine each component separately.

6.2. Hyperparameter Optimization. The performance of ML models strongly depends on the chosen hyperparameters, such as the regularization parameters for training kernel methods and the number of layers in NNs. Hence, it is often necessary to optimize the hyperparameters to achieve reasonable results and to improve the accuracy. The hyperparameter optimization commonly requires multiple trainings, making it an expensive endeavor, and caution must be paid in balancing performance/cost issues.

MLatom can optimize hyperparameters by minimizing the validation loss using one of the many available algorithms. The validation loss is usually based on the error in the validation set, which can be a single hold-out validation set or a combined cross-validation error.

For a few hyperparameters, the robust grid search on the log or linear scale can be used to find optimal values. It is a common choice for kernel methods (see Figure 13 for an example of optimizing hyperparameters of the KREG model, which is the kernel method). For a larger number of hyperparameters, other algorithms are recommended instead. Popular choices are Bayesian optimization with the tree-structured Parzen estimator (TPE)[78] and many SciPy optimizers.

The choice of the validation loss also matters. In most cases, MLatom minimizes the root-mean-square error (RMSE) for the labeled data. However, when multiple labels are provided, i.e., energies and energy gradients for learning PES, the choice should be made on how to combine them in the validation loss. By default, MLatom calculates the geometric mean of the RMSEs for energies and gradients.[29] The users can also choose a weighted sum of RMSEs, but in this case, they must choose the weight. In addition, the user can supply MLatom with any custom validation loss function, which can be arbitrarily complicated.

6.3. Evaluating Models. Once the model has been trained, it is common to evaluate its generalization ability before deployment in production simulations. MLatom provides dedicated options for such evaluations. The simplest and one of the most widespread approaches is calculating the error for the independent hold-out test set not used in the training. To emphasize, in MLatom terminology, the test set has no overlap with the training set, which might consist of the subtraining and validation subsets.[29] Alternatively, cross-validation and its variant leave-one-out cross-validation are recommended whenever computationally affordable, especially for small data sets. MLatom provides a broad range of error measures for the test set, including RMSE, mean absolute error (MAE), mean signed error, the Pearson correlation coefficient, the R^2 value, outliers, *etc*.[29] The testing can be performed with training and hyperparameter optimization for most models, including Δ-learning and self-correcting models.

Since the errors depend on the size of the training set, the learning curves showing this dependence are very useful for comparing different models.[29] MLatom can generate the learning curves, which have been instrumental in preparing guidelines for choosing the ML interatomic potential.[40]

7. SUMMARY

MLatom 3 is a unique software package combining machine learning and quantum mechanical models for accelerating and improving the accuracy of computational chemistry simulations. It can be used as a black-box package accepting input files with a simple structure or as a transparent Python module enabling custom workflows. MLatom provides access to pretrained models such as AIQM1 and ANI-1ccx aiming at high accuracy of the coupled-cluster level, making them more accurate and much faster than common DFT approaches for ground-state properties of closed-shell organic molecules. Another special pretrained model can be used to simulate two-photon absorption spectra.

The user of MLatom has an option to create their own models. Predefined ML architectures of the MACE, ANI-type, KREG, PhysNet, GAP-SOAP, DPMD, or sGDML make it easier. Alternatively, the custom models of varying complexity and based on combinations of both ML and QM models, such as Δ-learning, can be easily built with the package. MLatom provides a toolset for training, hyperparameter optimization, and performance analysis of the models.

This wide variety of models can be used for single-point calculations on large data sets, geometry optimizations, calculation of rovibrational (frequencies and IR spectra) and thermochemical (enthalpies, entropies, and heats of formation) properties, molecular dynamics, and UV/vis absorption spectra. The ML models can also be trained and used for quantum dissipative dynamics simulations.

For developers, MLatom provides a flexible platform for implementation of the new interfaces as they just need to provide a new class supporting prediction (and optionally training) with the new model. For example, the implementation of MACE was done in one working day, and another working day was needed for testing. Once implemented, these models can be readily used for simulations.

The richness of the MLatom functionality is available open-source and can be exploited on the XACS cloud computing service. We also welcome new contributions to the package. The package is accompanied by extensive and detailed manuals and tutorials that are developed and improved in close connection with teaching computational chemistry and machine learning in regular workshops and university courses.

■ ASSOCIATED CONTENT

Data Availability Statement

The MLatom code is open-source and available both on GitHub (https://github.com/dralgroup/mlatom, main GitHub repository) and PyPI (i.e., it can be installed via the command `pip install mlatom`). The contributions to the main GitHub repository of MLatom are highly welcome and can be done via pull requests from branches (on request) and forks that the contributors may also create for their private developments of methods and features. The pull requests may be incorporated into official releases after the review and eventual adjustments by the main developers' team managing the main GitHub repository. The simulations can also be run on the MLatom@ XACS cloud computing service on https://XACScloud.com.

■ AUTHOR INFORMATION

Corresponding Author

Pavlo O. Dral — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China;* orcid.org/0000-0002-2975-9876; Email: dral@xmu.edu.cn

Authors

Fuchun Ge — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China;* orcid.org/0000-0002-0112-5193

Yi-Fan Hou — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry,*

Xiamen, Fujian 361005, China; orcid.org/0000-0001-9188-5323

Peikun Zheng — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China;* orcid.org/0000-0002-0248-936X

Yuxinxin Chen — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China*

Mario Barbatti — *Aix Marseille University, CNRS, ICR, Marseille 13013, France; Institut Universitaire de France, Paris 75231, France;* orcid.org/0000-0001-9336-6607

Olexandr Isayev — *Department of Chemistry, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, United States;* orcid.org/0000-0001-7581-8497

Cheng Wang — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM) and iChem, Xiamen University, Xiamen, Fujian 361005, China;* orcid.org/0000-0002-7906-8061

Bao-Xin Xue — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China; Present Address: Present address: Xiamen Double Ten Middle School, Xiamen, Fujian 361009, China (B.-X.X.)*

Max Pinheiro Jr — *Aix Marseille University, CNRS, ICR, Marseille 13013, France; Present Address: Present address: Alstom Transport S.A., Saint-ouen-sur-seine, France (M.P.J.).;* orcid.org/0000-0002-5120-4172

Yuming Su — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM) and iChem, Xiamen University, Xiamen, Fujian 361005, China*

Yiheng Dai — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM) and iChem, Xiamen University, Xiamen, Fujian 361005, China; Present Address: Present address: Beijing National Laboratory for Molecular Sciences, College of Chemistry and Molecular Engineering, Peking University, Beijing 100190, China (Y.D.).*

Yangtao Chen — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM) and iChem, Xiamen University, Xiamen, Fujian 361005, China*

Lina Zhang — *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering,*

and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China; ⊙ orcid.org/0000-0002-4202-1067

Shuang Zhang − *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China;* Present Address: Present address: Neotrident (Suzhou) Co., Ltd., Suzhou, Jiangsu 215028, China (S.Z.).

Arif Ullah − *School of Physics and Optoelectronic Engineering, Anhui University, Hefei 230601, China;* ⊙ orcid.org/0000-0003-1702-3463

Quanhao Zhang − *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China;* ⊙ orcid.org/0009-0007-6416-3561

Yanchi Ou − *State Key Laboratory of Physical Chemistry of Solid Surfaces, College of Chemistry and Chemical Engineering, and Innovation Laboratory for Sciences and Technologies of Energy Materials of Fujian Province (IKKEM), Xiamen University, Xiamen, Fujian 361005, China; Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen, Fujian 361005, China;* Present Address: Present address: Shanghai Mayoo Technology, Inc., Shanghai 201318, China (Y.O.).

Complete contact information is available at:
https://pubs.acs.org/10.1021/acs.jctc.3c01203

Author Contributions

P.O.D. is the lead designer, developer, and maintainer of MLatom. F.G. is comaintaining the MLatom package, implemented interfaces to third-party machine learning packages (MACE, PhysNet, DeePMD-kit, TorchANI, GAP-SOAP, and hyperopt), wrote the code for learning curves, and made numerous other improvements in MLatom. Y.-F.H. coimplemented the KREG model, implemented molecular dynamics and vibrational spectra simulations, and improved many other parts of the code such as interfaces. P.Z. implemented AIQM1 and the ANI family of models (ANI-1ccx, ANI-2x, ANI-1x, and their dispersion-corrected variants) through interfaces to third-party packages (MNDO, TorchANI, and Sparrow) as well as geometry optimizations and frequency and thermochemistry simulations via interfaces to Gaussian, ASE, and TorchANI. Y.C. implemented interfaces to PySCF and Orca and extended thermochemical calculations to many methods. M.B. contributed to planning the implementation of MLPs and the methodology behind the ML-NEA approach. O.I. contributed to the research involving AIQM1 methods and ANI universal potentials. C.W. led the development of the ML-TPA methodology. B.-X.X. implemented the ML-NEA approach and initial argument parsing routines. M.P.J. helped implement the interfaces to TorchANI, PhysNet, DeePMD-kit, and

Newton-X. Y.S., Y.D., and Y.T.C. implemented the ML-TPA approach. L.Z. implemented routines for nonadiabatic dynamics and extensions of the MNDO interface to excited-state properties and tests of the MD code. S.Z. contributed to atomic property collection and implemented some of the NN-based approaches. A.U. interfaced MLQD to MLatom. Q.Z. contributed to the program documentation and tests. Y.O. contributed to plotting routines. P.O.D. wrote the original manuscript, and all authors revised and commented on the manuscript. F.G., Y.-F.H., Y.C., L.Z., Q.Z., and P.O.D. prepared the figures.

Notes

The authors declare no competing financial interest.
No data were generated for this article.

REFERENCES

(1) Himanen, L.; Jäger, M. O. J.; Morooka, E. V.; Federici Canova, F.; Ranawat, Y. S.; Gao, D. Z.; Rinke, P.; Foster, A. S. DScribe: Library of descriptors for machine learning in materials science. *Comput. Phys. Commun.* **2020**, *247*, No. 106949.

(2) Gao, X.; Ramezanghorbani, F.; Isayev, O.; Smith, J. S.; Roitberg, A. E. TorchANI: A Free and Open Source PyTorch-Based Deep Learning Implementation of the ANI Neural Network Potentials. *J. Chem. Inf. Model.* **2020**, *60*, 3408−3415.

(3) Chmiela, S.; Sauceda, H. E.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A. sGDML: Constructing accurate and data efficient molecular force fields using machine learning. *Comput. Phys. Commun.* **2019**, *240*, 38−45.

(4) Burn, M. J.; Popelier, P. L. A. FEREBUS: a high-performance modern Gaussian process regression engine. *Digit. Discovery* **2023**, *2*, 152−164.

(5) Browning, N. J.; Faber, F. A.; Anatole von Lilienfeld, O. GPU-accelerated approximate kernel method for quantum machine learning. *J. Chem. Phys.* **2022**, *157*, 214801.

(6) Abbott, A. S.; Turney, J. M.; Zhang, B.; Smith, D. G. A.; Altarawy, D.; Schaefer, H. F., 3rd PES-Learn: An Open-Source Software Package for the Automated Generation of Machine Learning Models of

Molecular Potential Energy Surfaces. *J. Chem. Theory Comput.* **2019**, *15*, 4386−4398.

(7) Quintas-Sanchez, E.; Dawes, R. AUTOSURF: A Freely Available Program To Construct Potential Energy Surfaces. *J. Chem. Inf. Model.* **2019**, *59*, 262−271.

(8) Novikov, I. S.; Gubaev, K.; Podryabinkin, E. V.; Shapeev, A. V. The MLIP package: moment tensor potentials with MPI and active learning. *Mach. Learn.: Sci. Technol.* **2021**, *2*, No. 025002.

(9) Laghuvarapu, S.; Pathak, Y.; Priyakumar, U. D. BAND NN: A Deep Learning Framework for Energy Prediction and Geometry Optimization of Organic Small Molecules. *J. Comput. Chem.* **2020**, *41*, 790−799.

(10) Zeng, J.; Zhang, D.; Lu, D.; Mo, P.; Li, Z.; Chen, Y.; Rynik, M.; Huang, L.; Li, Z.; Shi, S.; Wang, Y.; Ye, H.; Tuo, P.; Yang, J.; Ding, Y.; Li, Y.; Tisi, D.; Zeng, Q.; Bao, H.; Xia, Y.; Huang, J.; Muraoka, K.; Wang, Y.; Chang, J.; Yuan, F.; Bore, S. L.; Cai, C.; Lin, Y.; Wang, B.; Xu, J.; Zhu, J. X.; Luo, C.; Zhang, Y.; Goodall, R. E. A.; Liang, W.; Singh, A. K.; Yao, S.; Zhang, J.; Wentzcovitch, R.; Han, J.; Liu, J.; Jia, W.; York, D. M.; E, W.; Car, R.; Zhang, L.; Wang, H. DeePMD-kit v2: A software package for deep potential models. *J. Chem. Phys.* **2023**, *159*, No. 054801.

(11) Schütt, K. T.; Hessmann, S. S. P.; Gebauer, N. W. A.; Lederer, J.; Gastegger, M. SchNetPack 2.0: A neural network toolbox for atomistic machine learning. *J. Chem. Phys.* **2023**, *158*, 144801.

(12) Li, X. G.; Blaiszik, B.; Schwarting, M. E.; Jacobs, R.; Scourtas, A.; Schmidt, K. J.; Voyles, P. M.; Morgan, D. Graph network based deep learning of bandgaps. *J. Chem. Phys.* **2021**, *155*, 154702.

(13) Song, K.; Käser, S.; Töpfer, K.; Vazquez-Salazar, L. I.; Meuwly, M. PhysNet meets CHARMM: A framework for routine machine learning/molecular mechanics simulations. *J. Chem. Phys.* **2023**, *159*, No. 024125.

(14) López-Zorrilla, J.; Aretxabaleta, X. M.; Yeu, I. W.; Etxebarria, I.; Manzano, H.; Artrith, N. aenet-PyTorch: A GPU-supported implementation for machine learning atomic potentials training. *J. Chem. Phys.* **2023**, *158*, 164105.

(15) Ingolfsson, H. I.; Bhatia, H.; Aydin, F.; Oppelstrup, T.; Lopez, C. A.; Stanton, L. G.; Carpenter, T. S.; Wong, S.; Di Natale, F.; Zhang, X.; Moon, J. Y.; Stanley, C. B.; Chavez, J. R.; Nguyen, K.; Dharuman, G.; Burns, V.; Shrestha, R.; Goswami, D.; Gulten, G.; Van, Q. N.; Ramanathan, A.; Van Essen, B.; Hengartner, N. W.; Stephen, A. G.; Turbyville, T.; Bremer, P. T.; Gnanakaran, S.; Glosli, J. N.; Lightstone, F. C.; Nissley, D. V.; Streitz, F. H. Machine Learning-Driven Multiscale Modeling: Bridging the Scales with a Next-Generation Simulation Infrastructure. *J. Chem. Theory Comput.* **2023**, *19*, 2658−2675.

(16) Houston, P. L.; Qu, C.; Yu, Q.; Conte, R.; Nandi, A.; Li, J. K.; Bowman, J. M. PESPIP: Software to fit complex molecular and many-body potential energy surfaces with permutationally invariant polynomials. *J. Chem. Phys.* **2023**, *158*, No. 044109.

(17) Gelžinytė, E.; Wengert, S.; Stenczel, T. K.; Heenen, H. H.; Reuter, K.; Csányi, G.; Bernstein, N. wfl Python toolkit for creating machine learning interatomic potentials and related atomistic simulation workflows. *J. Chem. Phys.* **2023**, *159*, 124801.

(18) Hjorth Larsen, A.; Jorgen Mortensen, J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dulak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Bjerre Jensen, P.; Kermode, J.; Kitchin, J. R.; Leonhard Kolsbjerg, E.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Bergmann Maronsson, J.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; Schiotz, J.; Schutt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W. The atomic simulation environment-a Python library for working with atoms. *J. Phys.: Condens. Matter* **2017**, *29*, 273002.

(19) te Velde, G.; Bickelhaupt, F. M.; Baerends, E. J.; Fonseca Guerra, C.; van Gisbergen, S. J. A.; Snijders, J. G.; Ziegler, T. Chemistry with ADF. *J. Comput. Chem.* **2001**, *22*, 931−967.

(20) McSloy, A.; Fan, G.; Sun, W.; Hölzer, C.; Friede, M.; Ehlert, S.; Schütte, N.-E.; Grimme, S.; Frauenheim, T.; Aradi, B. TBMaLT, a flexible toolkit for combining tight-binding and machine learning. *J. Chem. Phys.* **2023**, No. 034801. DOI: 10.1063/5.0132892.

(21) Ple, T.; Mauger, N.; Adjoua, O.; Inizan, T. J.; Lagardere, L.; Huppert, S.; Piquemal, J. P. Routine Molecular Dynamics Simulations Including Nuclear Quantum Effects: From Force Fields to Machine Learning Potentials. *J. Chem. Theory Comput.* **2023**, *19*, 1432−1445.

(22) Dral, P. O. *MLatom*: A Program Package for Quantum Chemical Research Assisted by Machine Learning. *J. Comput. Chem.* **2019**, *40*, 2339−2347.

(23) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. Big Data Meets Quantum Chemistry Approximations: The Δ-Machine Learning Approach. *J. Chem. Theory Comput.* **2015**, *11*, 2087−2096.

(24) Dral, P. O.; von Lilienfeld, O. A.; Thiel, W. Machine Learning of Parameters for Accurate Semiempirical Quantum Chemical Calculations. *J. Chem. Theory Comput.* **2015**, *11*, 2120−2125.

(25) Dral, P. O.; Owens, A.; Dral, A.; Csányi, G. Hierarchical Machine Learning of Potential Energy Surfaces. *J. Chem. Phys.* **2020**, *152*, 204110.

(26) Dral, P. O.; Owens, A.; Yurchenko, S. N.; Thiel, W. Structure-based sampling and self-correcting machine learning for accurate calculations of potential energy surfaces and vibrational levels. *J. Chem. Phys.* **2017**, *146*, 244108.

(27) Dral, P. O.; Barbatti, M.; Thiel, W. Nonadiabatic Excited-State Dynamics with Machine Learning. *J. Phys. Chem. Lett.* **2018**, *9*, 5660−5663.

(28) de Rezende, A.; Malmali, M.; Dral, P. O.; Lischka, H.; Tunega, D.; Aquino, A. J. A. Machine Learning for Designing Mixed Metal Halides for Efficient Ammonia Separation and Storage. *J. Phys. Chem. C* **2022**, *126*, 12184−12196.

(29) Dral, P. O.; Ge, F.; Xue, B. X.; Hou, Y. F.; Pinheiro, M.; Huang, J.; Barbatti, M. MLatom 2: An Integrative Platform for Atomistic Machine Learning. *Top. Curr. Chem.* **2021**, *379*, 27.

(30) Xue, B.-X.; Barbatti, M.; Dral, P. O. Machine Learning for Absorption Cross Sections. *J. Phys. Chem. A* **2020**, *124*, 7199−7210.

(31) Su, Y.; Dai, Y.; Zeng, Y.; Wei, C.; Chen, Y.; Ge, F.; Zheng, P.; Zhou, D.; Dral, P. O.; Wang, C. Interpretable Machine Learning of Two-Photon Absorption. *Adv. Sci.* **2023**, 2204902 DOI: 10.1002/advs.202204902.

(32) Zheng, P.; Zubatyuk, R.; Wu, W.; Isayev, O.; Dral, P. O. Artificial Intelligence-Enhanced Quantum Chemical Method with Broad Applicability. *Nat. Commun.* **2021**, *12*, 7022.

(33) Smith, J. S.; Nebgen, B.; Lubbers, N.; Isayev, O.; Roitberg, A. E. Less is more: Sampling chemical space with active learning. *J. Chem. Phys.* **2018**, *148*, 241733.

(34) Smith, J. S.; Nebgen, B. T.; Zubatyuk, R.; Lubbers, N.; Devereux, C.; Barros, K.; Tretiak, S.; Isayev, O.; Roitberg, A. E. Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning. *Nat. Commun.* **2019**, *10*, 2903.

(35) Devereux, C.; Smith, J. S.; Huddleston, K. K.; Barros, K.; Zubatyuk, R.; Isayev, O.; Roitberg, A. E. Extending the Applicability of the ANI Deep Learning Molecular Potential to Sulfur and Halogens. *J. Chem. Theory Comput.* **2020**, *16*, 4192−4202.

(36) Zheng, P.; Yang, W.; Wu, W.; Isayev, O.; Dral, P. O. Toward Chemical Accuracy in Predicting Enthalpies of Formation with General-Purpose Data-Driven Methods. *J. Phys. Chem. Lett.* **2022**, *13*, 3479−3491.

(37) Dral, P. O.; Ge, F.; Hou, Y.-F.; Zheng, P.; Chen, Y.; Xue, B.-X.; Pinheiro, Jr, M.; Su, Y.; Dai, Y.; Chen, Y.; Zhang, S.; Zhang, L.; Ullah, A.; Ou, Y. *MLatom: A Package for Atomistic Simulations with Machine Learning*; Xiamen University: Xiamen, China, http://MLatom.com (accessed August 22, 2023), 2013−2023.

(38) Ullah, A.; Dral, P. O. Predicting the future of excitation energy transfer in light-harvesting complex with artificial intelligence-based quantum dynamics. *Nat. Commun.* **2022**, *13*, 1930.

(39) Gonzalez, C.; Schlegel, H. B. An improved algorithm for reaction path following. *J. Chem. Phys.* **1989**, *90*, 2154−2161.

(40) Pinheiro, M., Jr; Ge, F.; Ferré, N.; Dral, P. O.; Barbatti, M. Choosing the right molecular machine learning potential. *Chem. Sci.* **2021**, *12*, 14396−14413.

(41) Zhang, Y.; Lin, Q.; Jiang, B. Atomistic neural network representations for chemical dynamics simulations of molecular,

condensed phase, and interfacial systems: Efficiency, representability, and generalization. *WIREs Comput. Mol. Sci.* **2022**, No. e1645.

(42) Unke, O. T.; Chmiela, S.; Sauceda, H. E.; Gastegger, M.; Poltavsky, I.; Schutt, K. T.; Tkatchenko, A.; Muller, K. R. Machine Learning Force Fields. *Chem. Rev.* **2021**, *121*, 10142−10186.

(43) Manzhos, S.; Carrington, T., Jr. Neural Network Potential Energy Surfaces for Small Molecules and Reactions. *Chem. Rev.* **2021**, *121*, 10187−10217.

(44) Behler, J. Four Generations of High-Dimensional Neural Network Potentials. *Chem. Rev.* **2021**, *121*, 10037−10072.

(45) de Buyl, P.; Colberg, P. H.; Höfling, F. H5MD: A structured, efficient, and portable file format for molecular data. *Comput. Phys. Commun.* **2014**, *185*, 1546−1553.

(46) Rupp, M.; Tkatchenko, A.; Müller, K.-R.; von Lilienfeld, O. A. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.* **2012**, *108*, No. 058301.

(47) Hansen, K.; Montavon, G.; Biegler, F.; Fazli, S.; Rupp, M.; Scheffler, M.; von Lilienfeld, O. A.; Tkatchenko, A.; Müller, K.-R. Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies. *J. Chem. Theory Comput.* **2013**, *9*, 3404−3419.

(48) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H.; Li, X.; Caricato, M.; Marenich, A. V.; Bloino, J.; Janesko, B. G.; Gomperts, R.; Mennucci, B.; Hratchian, H. P.; Ortiz, J. V.; Izmaylov, A. F.; Sonnenberg, J. L.; Williams; Ding, F.; Lipparini, F.; Egidi, F.; Goings, J.; Peng, B.; Petrone, A.; Henderson, T.; Ranasinghe, D.; Zakrzewski, V. G.; Gao, J.; Rega, N.; Zheng, G.; Liang, W.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Throssell, K.; Montgomery, Jr., J. A.; Peralta, J. E.; Ogliaro, F.; Bearpark, M. J.; Heyd, J. J.; Brothers, E. N.; Kudin, K. N.; Staroverov, V. N.; Keith, T. A.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A. P.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Millam, J. M.; Klene, M.; Adamo, C.; Cammi, R.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Farkas, O.; Foresman, J. B.; Fox, D. J.*Gaussian 16, Rev. A.01*; Gaussian Inc.: Wallingford, CT, 2016.

(49) Dral, P. O.; Wu, X.; Thiel, W. Semiempirical Quantum-Chemical Methods with Orthogonalization and Dispersion Corrections. *J. Chem. Theory Comput.* **2019**, *15*, 1743−1760.

(50) Thiel, W., with contributions from ; Beck, M.; Billeter, S; Kevorkiants, R.; Kolb, M; Koslowski, A.; Patchkovskii, S.; Turner, A.; Wallenborn, E.-U.; Weber, W.; Spörkel, L.; Dral, P. O.*MNDO, development version*; Max-Planck-Institut für Kohlenforschung: Mülheim an der Ruhr, 2019.

(51) Bosia, F.; Zheng, P.; Vaucher, A.; Weymuth, T.; Dral, P. O.; Reiher, M. Ultra-Fast Semi-Empirical Quantum Chemistry for High-Throughput Computational Campaigns with Sparrow. *J. Chem. Phys.* **2023**, *158*, No. 054118.

(52) Bannwarth, C.; Ehlert, S.; Grimme, S. GFN2-xTB-An Accurate and Broadly Parametrized Self-Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions. *J. Chem. Theory Comput.* **2019**, *15*, 1652−1671.

(53) *Semiempirical extended tight-binding program package xtb*.https://github.com/grimme-lab/xtb (accessed on Nov. 19, 2022).

(54) Neese, F. Software update: the ORCA program system, version 4.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2018**, *8*, No. e1327.

(55) Neese, F. The ORCA program system. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2012**, *2*, 73−78.

(56) Caldeweyher, E.; Ehlert, S.; Grimme, S.*DFT-D4, Version 2.5.0*; Mulliken Center for Theoretical Chemistry, University of Bonn, 2020.

(57) Caldeweyher, E.; Bannwarth, C.; Grimme, S. Extension of the D3 dispersion coefficient model. *J. Chem. Phys.* **2017**, *147*, No. 034112.

(58) Batatia, I.; Kovács, D. P.; Simm, G. N. C.; Ortner, C.; Csányi, G. In *MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields*; Advances in Neural Information Processing Systems, https://openreview.net/forum?id=YPpSngE-ZU, 2022.

(59) Batatia, I.; Batzner, S.; Kovács, D. P.; Musaelian, A.; Simm, G. N. C.; Ortner, C.; Kozinsky, B.; Csányi, G. The Design Space of E(3)-Equivariant Atom-Centered Interatomic Potentials. *arXiv:2205.06643* **2022**, DOI: 10.48550/arXiv.2205.06643.

(60) *mace on* https://github.com/ACEsuit/mace.

(61) Zhang, L.; Han, J.; Wang, H.; Car, R.; E, W. Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. *Phys. Rev. Lett.* **2018**, *120*, 143001.

(62) Zhang, L. F.; Han, J. Q.; Wang, H.; Saidi, W. A.; Car, R.; E, W. N. End-To-End Symmetry Preserving Inter-Atomic Potential Energy Model for Finite and Extended Systems. *Adv. Neural. Inf. Process. Syst.* **2018**, *31*, 4436−4446.

(63) Wang, H.; Zhang, L.; Han, J.; E, W. DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Comput. Phys. Commun.* **2018**, *228*, 178−184.

(64) Unke, O. T.; Meuwly, M. PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *J. Chem. Theory Comput.* **2019**, *15*, 3678−3693.

(65) Hou, Y.-F.; Ge, F.; Dral, P. O. Explicit Learning of Derivatives with the KREG and pKREG Models on the Example of Accurate Representation of Molecular Potential Energy Surfaces. *J. Chem. Theory Comput.* **2023**, *19*, 2369−2379.

(66) Chmiela, S.; Sauceda, H. E.; Müller, K. R.; Tkatchenko, A. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.* **2018**, *9*, 3887.

(67) Bartók, A. P.; Payne, M. C.; Kondor, R.; Csányi, G. Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons. *Phys. Rev. Lett.* **2010**, *104*, 136403.

(68) Bartók, A. P.; Kondor, R.; Csányi, G. On representing chemical environments. *Phys. Rev. B* **2013**, *87*, 187115.

(69) Csanyi, G.; Winfield, S.; Kermode, J.; Payne, M. C.; Comisso, A.; De Vita, A.; Bernstein, N.*Expressive Programming for Computational Physics in Fortran 95+,*; Newsletter of the Computational Physics Group, 1−24: 2007.

(70) Chai, J.-D.; Head-Gordon, M. Long-range corrected hybrid density functionals with damped atom-atom dispersion corrections. *Phys. Chem. Chem. Phys.* **2008**, *10*, 6615−6620.

(71) Becke, A. D. Density-functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys.* **1993**, *98*, 5648−5652.

(72) Stephens, P. J.; Devlin, F. J.; Chabalowski, C. F.; Frisch, M. J. Ab Initio Calculation of Vibrational Absorption and Circular Dichroism Spectra Using Density Functional Force Fields. *J. Phys. Chem.* **1994**, *98*, 11623−11627.

(73) Dral, P. O.; Wu, X.; Spörkel, L.; Koslowski, A.; Weber, W.; Steiger, R.; Scholten, M.; Thiel, W. Semiempirical Quantum-Chemical Orthogonalization-Corrected Methods: Theory, Implementation, and Parameters. *J. Chem. Theory Comput.* **2016**, *12*, 1082−1096.

(74) Dewar, M. J. S.; Zoebisch, E. G.; Healy, E. F.; Stewart, J. J. P. Development and use of quantum mechanical molecular models. 76. AM1: a new general purpose quantum mechanical molecular model. *J. Am. Chem. Soc.* **1985**, *107*, 3902−3909.

(75) Stewart, J. J. P. Optimization of parameters for semiempirical methods V: Modification of NDDO approximations and application to 70 elements. *J. Mol. Model.* **2007**, *13*, 1173−1213.

(76) Hou, Y.-F.; Dral, P. O., Kernel method potentials. In *Quantum Chemistry in the Age of Machine Learning*, Dral, P. O., Ed. Elsevier: Amsterdam, Netherlands, 2023.

(77) Bergstra, J.; Yamins, D.; Cox, D. D. In Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures, *Proceedings of the 30th International Conference on International Conference on Machine Learning, Atlanta, GA, USA*; JMLR.org: Atlanta, GA, USA, 2013; pp I−115−I−123.

(78) Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B., Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*; Shawe-Taylor, J.; Zemel, R.; Bartlett, P.; Pereira, F.; Weinberger, K. Q., Eds. Curran Associates, Inc.: 2011; Vol. *24*.

(79) Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.;

Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; SciPy Contributors. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* **2020**, *17*, 261−272.

(80) Hofmann, T.; Schölkopf, B.; Smola, A. J. Kernel methods in machine learning. *Ann. Statist.* **2008**, *36*, 1171−1220.

(81) Pinheiro, Jr., M.; Dral, P. O., Kernel methods. In *Quantum Chemistry in the Age of Machine Learning*; Dral, P. O., Ed. Elsevier: Amsterdam, Netherlands, 2023.

(82) Rasmussen, C. E.; Williams, C. K. I. *Gaussian Processes for Machine Learning*; The MIT Press: Boston, 2006.

(83) Gneiting, T.; Kleiber, W.; Schlather, M. Matérn Cross-Covariance Functions for Multivariate Random Fields. *J. Am. Stat. Assoc.* **2010**, *105*, 1167−1177.

(84) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825−2830.

(85) Ge, F.; Zhang, L.; Hou, Y.-F.; Chen, Y.; Ullah, A.; Dral, P. O. Four-Dimensional-Spacetime Atomistic Artificial Intelligence Models. *J. Phys. Chem. Lett.* **2023**, *14*, 7732−7743.

(86) Herrera Rodríguez, L. E.; Ullah, A.; Rueda Espinosa, K. J.; Dral, P. O.; Kananenka, A. A. A comparative study of different machine learning methods for dissipative quantum dynamics. *Mach. Learn. Sci. Technol.* **2022**, *3*, No. 045016.

(87) Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed.; Springer-Verlag: New York, 2009.

(88) Freund, Y.; Seung, H. S.; Shamir, E.; Tishby, N. Selective Sampling Using the Query by Committee Algorithm. *Mach. Learn.* **1997**, *28*, 133−168.

(89) Schaub, T. A.; Zieleniewska, A.; Kaur, R.; Minameyer, M.; Yang, W.; Schüßlbauer, C. M.; Zhang, L.; Freiberger, M.; Zakharov, L. N.; Drewello, T.; Dral, P. O.; Guldi, D. M.; Jasti, R. Tunable Macrocyclic Polyparaphenylene Nanolassos via Copper-Free Click Chemistry. *Chem. - Eur. J.* **2023**, *29*, No. e202300668.

(90) Henkelman, G.; Jónsson, H. A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives. *J. Chem. Phys.* **1999**, *111*, 7010−7022.

(91) Schlegel, H. B. Optimization of equilibrium geometries and transition structures. *J. Comput. Chem.* **1982**, *3*, 214−218.

(92) Barone, V. Anharmonic vibrational properties by a fully automated second-order perturbative approach. *J. Chem. Phys.* **2005**, *122*, 14108.

(93) Goerigk, L.; Hansen, A.; Bauer, C.; Ehrlich, S.; Najibi, A.; Grimme, S. A look at the density functional theory zoo with the advanced GMTKN55 database for general main group thermochemistry, kinetics and noncovalent interactions. *Phys. Chem. Chem. Phys.* **2017**, *19*, 32184−32215.

(94) Curtiss, L. A.; Raghavachari, K.; Redfern, P. C.; Pople, J. A. Assessment of Gaussian-2 and density functional theories for the computation of enthalpies of formation. *J. Chem. Phys.* **1997**, *106*, 1063−1079.

(95) Pedley, J. B.; Naylor, R. D.; Kirby, S. P. *Thermochemical Data of Organic Compounds*; Springer Dordrecht: New York, 1986.

(96) Zhong, X.; Zhao, Y., Basics of dynamics. In *Quantum Chemistry in the Age of Machine Learning*; Dral, P. O., Ed. Elsevier: Amsterdam, Netherlands, 2023; pp 117−133.

(97) Groenhof, G. Introduction to QM/MM simulations. *Methods Mol. Biol.* **2013**, *924*, 43−66.

(98) Chen, S.; Feng, S.; Markvoort, A. J.; Zhang, C.; Zhou, E.; Liang, W.; Zhang, H.; Jiang, Y.; Lin, J. Unequal Perylene Diimide Twins in a Quadruple Assembly. *Angew. Chem., Int. Ed. Engl.* **2023**, *62*, No. e202300786.

(99) Zhang, L.; Hou, Y.-F.; Ge, F.; Dral, P. O. Energy-conserving molecular dynamics is not energy conserving. *Phys. Chem. Chem. Phys.* **2023**, *25*, 23467−23476.

(100) Linstrom, E. P.; Mallard, W. *NIST Chemistry WebBook, NIST Standard Reference Database Number 69*. https://webbook.nist.gov/chemistry/.

(101) Frenkel, D.; Smit, B.; Tobochnik, J.; Mckay, S. R.; Christian, W. *Understanding Molecular Simulation*; Elsevier: Bodmin, Cornwall, 1997.

(102) Swope, W. C.; Andersen, H. C.; Berens, P. H.; Wilson, K. R. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.* **1982**, *76*, 637−649.

(103) Andersen, H. C. Molecular dynamics simulations at constant pressure and/or temperature. *J. Chem. Phys.* **1980**, *72*, 2384−2393.

(104) Martyna, G. J.; Klein, M. L.; Tuckerman, M. Nosé−Hoover chains: The canonical ensemble via continuous dynamics. *J. Chem. Phys.* **1992**, *97*, 2635−2643.

(105) Martyna, G. J.; Tuckerman, M. E.; Tobias, D. J.; Klein, M. L. Explicit reversible integrators for extended systems dynamics. *Mol. Phys.* **1996**, *87*, 1117−1157.

(106) Barbatti, M.; Bondanza, M.; Crespo-Otero, R.; Demoulin, B.; Dral, P. O.; Granucci, G.; Kossoski, F.; Lischka, H.; Mennucci, B.; Mukherjee, S.; Pederzoli, M.; Persico, M.; Pinheiro, M., Jr; Pittner, J.; Plasser, F.; Sangiogo Gil, E.; Stojanovic, L. Newton-X Platform: New Software Developments for Surface Hopping and Nuclear Ensembles. *J. Chem. Theory Comput.* **2022**, *18*, 6851−6865.

(107) Zhang, L.; Ullah, A.; Pinheiro, Jr, M.; Barbatti, M.; Dral, P. O., Excited-state dynamics with machine learning. In *Quantum Chemistry in the Age of Machine Learning*; Dral, P. O., Ed. Elsevier: Amsterdam, Netherlands, 2023.

(108) Mukherjee, S.; Pinheiro, M.; Demoulin, B.; Barbatti, M. Simulations of molecular photodynamics in long timescales. *Philos. Trans. Soc. A* **2022**, *380*, 20200382.

(109) Weiss, U. *Quantum Dissipative Systems*; World Scientific Publishing: Singapore, 2012.

(110) Ullah, A.; Dral, P. O. Speeding up quantum dissipative dynamics of open systems with kernel methods. *New J. Phys.* **2021**, *23*, 113019.

(111) Ullah, A.; Dral, P. O. One-Shot Trajectory Learning of Open Quantum Systems Dynamics. *J. Phys. Chem. Lett.* **2022**, *13*, 6037−6041.

(112) Ullah, A.; Dral, P. O. MLQD: A package for machine learning-based quantum dissipative dynamics. *Comput. Phys. Commun.* **2024**, *294*, 108940.

(113) Thomas, M.; Brehm, M.; Fligg, R.; Vohringer, P.; Kirchner, B. Computing vibrational spectra from ab initio molecular dynamics. *Phys. Chem. Chem. Phys.* **2013**, *15*, 6608−6622.

(114) Tikhonov, D. S.; Sharapa, D. I.; Schwabedissen, J.; Rybkin, V. V. Application of classical simulations for the computation of vibrational properties of free molecules. *Phys. Chem. Chem. Phys.* **2016**, *18*, 28325−28338.

(115) Crespo-Otero, R.; Barbatti, M. Spectrum simulation and decomposition with nuclear ensemble: formal derivation and application to benzene, furan and 2-phenylfuran. *Theor. Chem. Acc.* **2012**, *131*, 1237.

(116) Schinke, R. *Photodissociation dynamics: spectroscopy and fragmentation of small polyatomic molecules*; Cambridge University Press: Cambridge, 1995.

(117) *RDKit: Open-source cheminformatics*; http://www.rdkit.org.

(118) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31−36.

(119) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. Open Babel: An open chemical toolbox. *J. Cheminform.* **2011**, *3*, 33.

(120) Christensen, A. S.; von Lilienfeld, O. A. On the role of gradients for machine learning of molecular energies and forces. *Mach. Learn.: Sci. Technol.* **2020**, *1*, No. 045018.

(121) Pinheiro, M., Jr.; Zhang, S.; Dral, P. O.; Barbatti, M. WS22 database: combining Wigner Sampling and geometry interpolation

towards configurationally diverse molecular datasets. *Sci. Data* **2023**, *10*, 95.