MiMiC: A High-Performance Framework for Multiscale Molecular Dynamics Simulations

Andrej Antalík,[1] Andrea Levy,[1] Sonata Kvedaravičiūtė,[2] Sophia K. Johnson,[1] David Carrasco-Busturia,[2, a)] Bharath Raghavan,[3, 4] François Mouvet,[1] Angela Acocella,[5] Sambit Das,[6] Vikram Gavini,[6, 7] Davide Mandelli,[3] Emiliano Ippoliti,[3] Simone Meloni,[8] Paolo Carloni,[3, 4] Ursula Rothlisberger,[1] and Jógvan Magnus Haugaard Olsen[2]

[1)] *Laboratory of Computational Chemistry and Biochemistry, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland*

[2)] *DTU Chemistry, Technical University of Denmark (DTU), DK-2800 Kongens Lyngby, Denmark*

[3)] *Computational Biomedicine, Institute of Advanced Simulations IAS-5/Institute for Neuroscience and Medicine INM-9, Forschungszentrum Jülich GmbH, Jülich 52428, Germany*

[4)] *Department of Physics, RWTH Aachen University, Aachen 52074, Germany*

[5)] *CINECA, 40033 Casalecchio di Reno, BO, Italy*

[6)] *Department of Mechanical Engineering, University of Michigan, Ann Arbor, Michigan 48109, USA*

[7)] *Department of Materials Science and Engineering, University of Michigan, Ann Arbor, Michigan 48109, USA*

[8)] *Dipartimento di Scienze Chimiche, Farmaceutiche ed Agrarie (DOCPAS), Università degli Studi di Ferrara (Unife), I-44121 Ferrara, Italy*

(*Author to whom correspondence should be addressed: jmho@kemi.dtu.dk)

(Dated: 29 March 2024)

MiMiC is a framework for performing multiscale simulations, where individual subsystems are handled at different resolutions and/or levels of theory by loosely coupled external programs. To make it highly efficient and flexible, we adopt an interoperable approach based on a multiple-program multiple-data paradigm, serving as an intermediary responsible for fast data exchange and interactions between the subsystems. The main goal of MiMiC is to avoid interfering with the underlying parallelization of the external programs, including the operability on hybrid architectures (e.g., CPU/GPU), and keep their setup and execution as close as possible to the original. At the moment, MiMiC offers an efficient implementation of electrostatic embedding QM/MM that has demonstrated unprecedented parallel scaling in simulations of large biomolecules using CPMD and GROMACS as QM and MM engines, respectively. However, as it is designed for high flexibility with general multiscale models in mind, it can be straightforwardly extended beyond QM/MM. In this article, we illustrate the software design and the features of the framework, which make it a compelling choice for multiscale simulations in the upcoming era of exascale high-performance computing.

I. INTRODUCTION

Chemical and biological phenomena span large temporal and spatial scales. An accurate representation of the physics across these diverse scales requires the development of multiscale simulation methods able to employ different resolutions and levels of theory within the scope of a single simulation. An approach involving quantum mechanics (QM) is essential to describe the reorganization of electrons and nuclei, e.g., during reactive events involving bond breaking and formation, yet a molecular mechanics (MM) model characterizes most protein and nucleic-acid properties with sufficient accuracy. Furthermore, applying even lower resolutions such as coarse-grained (CG)[1–4] models enables simulations of larger systems over longer time scales. Probably the most well-known example of a multiscale model is QM/MM, where a small part of a system is treated at a QM level of theory, and the remainder is modeled using an MM force field.[5–11]

Multiscale methods are usually implemented by extending a program with the functionality of another, e.g., in the case of QM/MM, a QM program with MM functionality, or vice versa. These implementations range from fully monolithic, through linked libraries with ad hoc interfaces to stand-alone programs, up to general interfaces between independent programs. Although most QM/MM implementations adopt the first two of these strategies, some software packages implement more flexible interfaces that allow coupling to, in principle, any other program without substantial effort.[12–14] Apart from these already more general interfaces, an even more flexible approach is to implement multiscale methods through integrative frameworks. These do not themselves implement any QM, MM, or other such functionality for calculating the properties of individual subsystems, but instead rely completely on external programs.[15–30]

An important aspect of multiscale implementations involving several distinct software components is their coupling, which can be classified as tight or loose based on the degree of interdependency between them. Loose coupling helps to avoid code duplication and greatly reduces maintenance costs, as modifications made to one component do not impact the rest and conversely, new features implemented in a given component are instantly available with practically no additional effort. In addition, replacing individual components becomes almost trivial. Loose coupling to the external programs is fundamental to essentially all general multiscale frameworks. However, obtaining high computational efficiency can pose a challenge due to potentially slow data exchange compared to

[a)] Current address: Department of Theoretical Chemistry and Biology, School of Engineering Sciences in Chemistry, Biotechnology and Health, KTH Royal Institute of Technology, SE-106 91 Stockholm, Sweden

the tight-coupling approach, where the relevant data is available in memory. In particular, most frameworks rely on file-based communication, i.e., using input and output files, which can have a substantial impact on performance.[31,32] To overcome these issues, some adopt a library-linking approach that requires converting the external programs into libraries, thus losing some of the flexibility, while others are realized through network-based communication.

Recently, we introduced MiMiC[33], a high-performance multiscale modeling framework that combines the best of both worlds, the flexibility of loose coupling, while achieving high computational efficiency[34]. It attains these advantages by employing a client-server approach together with a multiple-program multiple-data (MPMD) model through which it loosely couples external programs that concurrently calculate different subsystem contributions while MiMiC itself calculates interactions between them, see illustration in Figure 1. MiMiC has a dual-sided application programming interface (API), one side of which is used to communicate with a molecular dynamics (MD) driver that integrates the equations of motion, while the other side interfaces to external programs. To ensure efficient and straightforward interfacing with the latter, we developed the MiMiC communication library (MCL) that features a simple API and, moreover, provides a means to achieve efficient network-based communication while minimizing the impact on the underlying parallelization of the external programs. This allows us to exploit the efficiency and features of specialized software packages, thus granting access to multiscale simulations that can take full advantage of state-of-the-art high-performance computing (HPC) architectures.

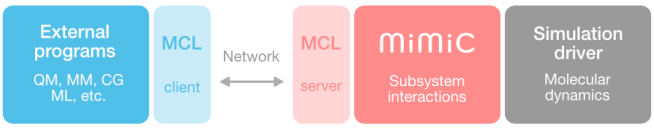

FIG. 1. Illustration of the strategy used by the MiMiC framework

In this article, we give a detailed presentation of the software design of the MiMiC framework and showcase its features, performance, and recent applications of large biomolecular systems. Section II describes the internal organization of the code and the key aspects of communication, which might also prove useful in potential future interfaces. Section III illustrates workflows of multiscale simulations with MiMiC, focusing on electrostatic and polarizable embedding QM/MM schemes and a multiple time step (MTS) approach for QM/MM MD. Section IV provides an overview of current and upcoming features either implemented directly in MiMiC or accessible through external programs. Section V summarizes some notable studies in which MiMiC enabled the efficient and accurate modeling of complex biophysical and biochemical processes. We conclude this paper by outlining the road ahead.

II. SOFTWARE DESIGN

The design of the MiMiC framework is guided by our primary objectives: achieving optimal flexibility and performance while minimizing implementation and maintenance efforts. Flexibility is attained through a modular design where individual subsystems are assigned to external client programs that compute the relevant properties needed for a multiscale simulation. In this MPMD setup, MiMiC serves as the intermediary that collects and distributes data among the concurrently running client programs, and at the same time, efficiently calculates contributions from the interactions between the subsystems. To accommodate different types of subsystems, particles, and other quantities, we draw extensively on the object-oriented capabilities of modern Fortran and organize data structures into a hierarchy of derived types that can be easily extended. Once a given type of subsystem interaction is implemented, it is then trivially transferable to any compatible program with no additional coding effort besides the implementation of the interface itself. Apart from easy access to new features in the client programs and the MD driver, this approach avoids code duplication and substantially reduces the maintenance effort to keep the MiMiC ecosystem operational.

To fulfill our high-performance objective, we aim to minimize computational overheads associated with using multiple loosely coupled client programs. To avoid their repeated startup and shutdown, we came up with an interoperable approach that does not interfere with the parallelization of these programs while keeping their setup and execution as close as possible to the original. For this purpose, we developed MCL that facilitates data exchange between MiMiC and the clients through a clean interface, agnostic of the underlying communication mechanism, and which allows us to make use of the features and optimizations offered by the client programs. MCL is written in C++ with simple C and Fortran APIs, thus ensuring interoperability with basically any programming language. Currently, it supports two MPI-based modes of communication, but other mechanisms can be implemented essentially without any changes to the interfaces in the client programs.

A. Interfaces

An essential feature of the loose-coupling paradigm is the design of a well-defined and general interface between individual programs, as opposed to several program-specific interfaces. For practical purposes, the MiMiC framework is split into two separate libraries, namely, the main MiMiC library and MCL, each providing an API intended for a specific function. The first is linked to the simulation driver program, which also acts as the server that manages communication between the individual clients, keeps track of all simulation-related information, and computes interactions between the subsystems. However, in order to become a client, an external program responsible for a particular subsystem requires only MCL, which facilitates data exchange with the server and can

be introduced with only minimal changes in the original code.

1. Client-Server Communication

To go along with our minimal interference approach for clients, MCL's API consists of only a handful of procedures, quite similar to those of MPI. Here, we illustrate client-server communication of a generic run within the context of MPI MPMD, see Figure 2.

The simulation starts by launching all programs as an MPI MPMD job, meaning that MPI launches these programs simultaneously. All of them now share a single `MPI_COMM_WORLD` communicator that is intercepted by invoking `MCL_Init()` right after the MPI initialization with `MPI_Init()`. This splits the communicator and returns new local "world" communicators, specific for the individual programs, with all communication between the clients and the server being channeled through intercommunicators. The next step is to establish the client-server connections by calling the `MCL_Handshake()` procedure in all running programs to specify the executable that will be the server and, at the same time, to identify individual clients by assigning them IDs.

Algorithm 1 Pseudocode example of the MiMiC loop.

```
while ( not is_last_step) do
    request = mimic_receive_request()
    if (request == MCL_RUN_ENERGY_FORCES) then
        energy, forces = calculate_energy_forces()
    else if (request == MCL_SEND_FORCES) then
        mimic_send_forces(forces)
    else if (request == MCL_...) then
            ⋮
    else if (request == MCL_EXIT) then
        is_last_step = true
    else
        Abort("Unrecognized MiMiC request!")
    end
end
```

At this stage, the server (MD driver) does not have any information about the subsystems, as these are specified in the inputs of the client programs. Therefore, it first collects this information from the clients, including details relevant to the allocation of data structures in the MD driver (numbers of atoms and species/atom types), as well as details about the particular objects within the simulation needed for computing the subsystem interactions and propagating the system in time (masses, atomic numbers, etc.). Once everything is set, the program enters the main MD loop, which, in terms of communication, predominantly consists of distributing coordinates to and collecting forces from the clients.

To ensure minimal interference with the infrastructure of a client program, while at the same time maintaining maximum flexibility, we coordinate actions between the clients and the server with a request-based approach. In general, a client initializes as it normally would and then enters a construct that we refer to as the *MiMiC loop*, which is merely a conditional loop in which two actions are performed every iteration. First, it receives a request via `MCL_Recv()`, and then it executes the specified action given that it has already been implemented. For an implementation example, see Algorithm 1.

The entire MiMiC loop basically reduces to the execution of one of four possible types of instructions. Requests prepended by `MCL_SEND` instruct a client to send some sort of data. This could involve allocating a buffer, filling it with requested data, and then sending it using `MCL_Send()`. The reciprocal action is a receive data request, which is prepended by `MCL_RECV`. The last type of practical request, denoted by the prefix `MCL_RUN`, instructs a client to execute a calculation, e.g., to compute the energy, forces, or both. Finally, the server can order clients to stop the program execution by issuing the `MCL_EXIT` request. Once it is received, the client program wraps up the run, deallocates memory, and ultimately terminates the program. This typically occurs at the end of a simulation or when an error is raised by one of the programs.

2. MD Driver API

So far, we have described only the client-side API, where MCL is used as a communication channel. However, a program that acts as a server in terms of communication and as a driver in terms of an MD simulation is also needed. In comparison to the client-side interface, this program has to be linked against the MiMiC library, which then provides access to the MiMiC infrastructure via a simple API. It consists of a collection of setter-, getter-, and runner-type procedures that can be easily incorporated into a typical MD loop. At the moment, only a Fortran API is available, but we intend to extend it to other languages.

Let us demonstrate this with a short example of the procedure calls during a single time step of an MD simulation. Once the MD driver updates the particle positions, they are passed to MiMiC using the `set_coordinates()` function. Then, `run_calculation()` is called at the point in the code where the MD driver would typically perform a force calculation. By invoking this function, MiMiC carries out all the necessary steps to provide the MD driver with the forces on all particles, i.e., it distributes coordinates to individual clients via MCL, issues commands to start energy and force calculations, collects relevant quantities, and computes interactions between the subsystems. Finally, forces are collected and updated in the MD driver via `get_forces()`, and the simulation then proceeds with the rest of the MD time step.

B. Data Structures

The MiMiC main module comprises two parts, a public API for MD drivers and a private instance of the *system* class (which here is a Fortran module with a derived type) that contains all the information about the simulation, such as the simulation box parameters, *species* objects with particle properties, and, most importantly, *subsystem* and *interaction* objects.

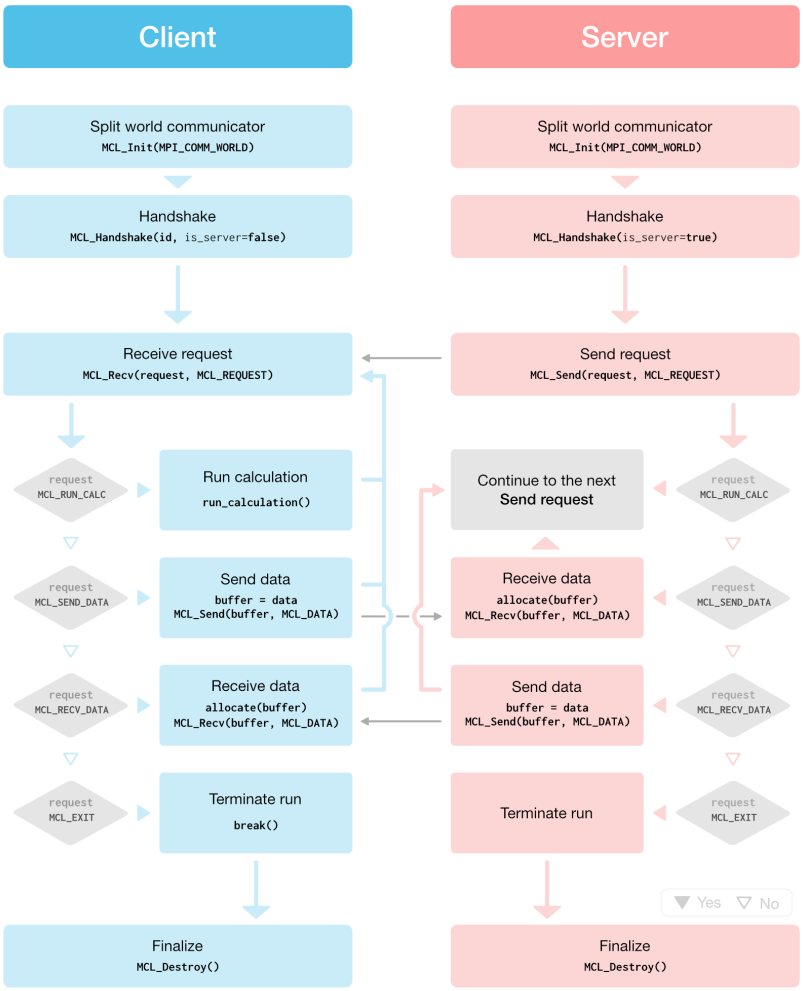

FIG. 2. Schematic representation of the communication between the server and an external client program during a simulation.

All system data are thus stored in a hierarchy of objects illustrated in Figure 3. This data structure ladder plays a crucial role in the modularity of MiMiC, particularly *subsystem* and *interaction* classes, which are the focus of this section.

The fundamental objects in MiMiC are the instances of the *subsystem* class, as each of them is associated with an external program that calculates its properties. In its essence, the base *subsystem* class contains the energy of a given subsystem, its box parameters (if present), and a set of particles, which can be clustered in fragments. Because of its association with a client program, it also includes several methods (called type-bound procedures in Fortran) that are used for data transfer and communication via MCL, and to which we collectively refer to as a communicator. In the initial phase of a simulation, they are used to collect data about the subsystem, like the number of particles and their species, and then, during the MD loop, to distribute particle coordinates to individual clients, instruct them to execute a calculation, and, once it is finished, collect the final energies, forces, and possibly other quantities. So far, we have implemented two derived *subsystem* classes, namely an MM subsystem and a QM subsystem. The latter can be straightforwardly used for programs that have a real-space grid representation of the electron density and external potentials, such as those based on plane waves (PWs). For other basis sets, such as Gaussian-type orbitals (GTOs), it may be necessary to implement additional functionality, such as mapping the electron density onto a real-space grid or including the external potential to the QM Hamiltonian, e.g., from MM point charges.

The *interaction* class groups objects required for the calculation of subsystem interactions. Here, we use an electrostatic embedding QM/MM scheme as an example to explain the essence of this class. Among its attributes, we include pointers to the involved subsystems, two arrays referencing the short-range (sr) and long-range (lr) MM atoms, and settings related to the atom sorting into sr/lr subdomains (for the theoretical basis, see Section III A). Finally, its methods constitute an interface to compute different quantities that represent the interaction, be it the potential of MM atoms acting on the QM subsystem, QM/MM energy, or QM/MM forces.

Possible extensions of the functionality, for example, to perform MM/CG simulations, can be achieved by a few ad-

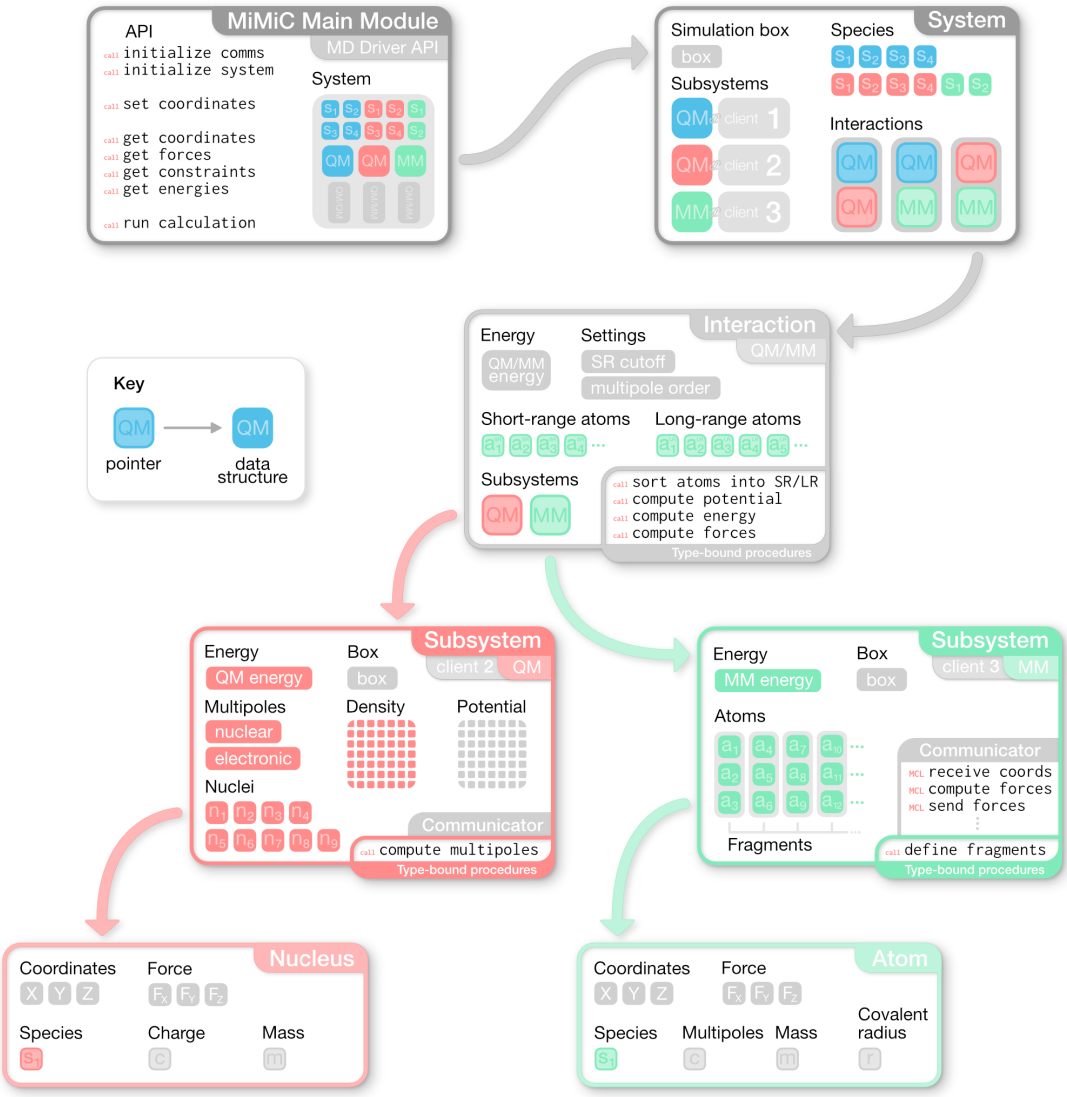

FIG. 3. Illustration of the internal structure of the MiMiC main module as a hierarchy of classes. Note that this structure is by no means exhaustive, and not all of these features are implemented.

ditions. First, it is necessary to implement a new *subsystem* class to accommodate a CG client program and all its characteristics. One can, for instance, introduce a new class for CG beads by extending the base *particle* class, which would then represent the particles in a new CG *subsystem*. Moreover, a new *interaction* class needs to be implemented to calculate the interactions between the CG beads and MM atoms. Further extensions to enable, e.g., a three-layer multiscale model, proceed in a similar way.

C. Parallelization Strategy

MiMiC computes various quantities related to the interactions between subsystems, and it is therefore assigned its own computational resources. To avoid introducing a bottleneck, we use parallelization strategies that are flexible enough to run efficiently on diverse hardware architectures, while at the same time being highly scalable. Therefore, for computationally intensive routines, most data is stored in flat, contiguous arrays, ensuring that the number of nested loops is kept to a minimum. This allows to potentially tailor parallelization to specific needs that might emerge in the future, for example, due to the addition of new subsystem and interaction types.

In terms of employed parallelization schemes, MiMiC currently features a hybrid MPI-based distributed-memory and OpenMP-based shared-memory approach, which is particularly important in the calculation of electrostatic QM/MM interactions, as described in Section III. Within this scheme, the most computationally intensive operation is the numerical integration over the grid storing either an electron density or an external potential, which has to be performed for each MM atom interacting with the QM subsystem. In a typical application, the number of atoms N_{atom} is on the order of 10^4 to 10^5,

while the number of grid points N_{grid} is on the order of 10^6 to 10^7. However, the number of atoms treated explicitly can be reduced to the order of 10^3 by employing the sr/lr scheme (see Section III A), thus resulting in a total operation count proportional to $N_{\text{atom}} \times N_{\text{grid}}$, i.e., between 10^9 and 10^{10}. In the current implementation, it then reduces to two nested loops: the outer loop over the MM atoms is distributed over MPI processes and both loops are further parallelized employing OpenMP-based shared-memory approaches.

D. Preprocessing Toolkit

The loose-coupling, minimally-invasive strategy adopted by MiMiC requires the preparation of inputs for multiple external programs that are run concurrently. Sometimes, setting up such simulations can be challenging, especially in view of the partitioning of complex biomolecular systems into two or more subsystems that need to be set up independently. For this purpose, we provide MiMiCPy[35], a Python-based modular preprocessing toolkit for MiMiC, that supports various formats of topology, coordinates, and input files and is easily extendable to new ones.

This set of tools facilitates the preparation of input files for MiMiC-based simulations by determining the mappings of the particles that are treated simultaneously by different programs, like QM atoms in a QM/MM simulation, where the MM client is responsible for calculating bonded and van der Waals QM/MM interactions. MiMiCPy can be used from the command line, but also like a Python library and, notably, we implemented user-friendly plugins for commonly used visualization programs for biomolecular simulations like VMD[36] and PyMOL[37]. This feature is especially helpful for selecting complex QM regions that require careful visual inspection.

MiMiCPy supports three major file types widely used in computational chemistry, namely, topology, coordinates, and input files. Internally, each of these has a corresponding module that further contains multiple parsers specific for every supported file format. As parsers are only accessible from within their enclosing modules, each module offers a dedicated wrapper that serves as a unified interface for data exchange. This means that adding support for a new file format only involves adding a parser class that interfaces with the appropriate wrapper, with no further changes to the core program. This allows for easy extensibility and quick support for new programs added to MiMiC.

E. Software Development Practices

The MiMiC framework is free and open-source software distributed under the GNU Lesser General Public License (LGPL) version 3.0 or later. The license was chosen to support the free and open-source software movement without imposing severe restrictions on the client programs since proprietary software is still common in the molecular simulation community. Thus, MCL may be used or integrated into essentially any software without being required to release its source code. The MiMiC source code is hosted on GitLab[38] with releases deposited on Zenodo[39,40] using versioning according to the semantic versioning specification[41]. We strive to maintain a high code quality through code review and continuous integration testing that includes unit and integration tests.

Community contributions are highly appreciated, whether through bug reporting, feature suggestions, or direct code contributions. To facilitate community engagement, we offer user support through a dedicated GitLab issue tracker (`https://gitlab.com/mimic-project/user-support`), apart from the project-specific GitLab issue trackers, and a discussion group hosted on Google Groups (`https://groups.google.com/g/mimic-project`). General information about MiMiC, including installation instructions and tutorials, is available on our website `https://mimic-project.org`.

III. WORKFLOW EXAMPLES

The overall workflow for multiscale simulations varies depending on the system under study and the specific multiscale method employed. For multiscale QM/MM MD simulations, we typically start with a system that has been equilibrated at the classical MM level. This MM equilibration phase involves the usual steps for classical MD simulations, such as energy minimization and equilibration to target conditions (e.g., ambient temperature) in the selected thermodynamic ensemble. Subsequently, input files for all programs involved in the simulation are prepared. This includes providing MiMiC with the information needed for client communication, the definition of subsystems, and the computation of subsystem interactions. The MiMiCPy preprocessing toolkit, described in Section II D, is designed to facilitate this part of the workflow[35]. Following the preparation, the multiscale simulation is performed. Like in any classical or ab inito MD workflow, this includes benchmarking to ensure optimal use of computational resources, equilibration, and production simulations. The generated trajectory data can be post-processed to evaluate various properties of interest. Although we currently rely on the output files produced by the external programs that can be analyzed using standard tools for MD analysis, we plan to introduce our own output to facilitate and homogenize the post-processing phase.

MiMiC-based multiscale simulations can proceed in several different ways depending on the method and its implementation. In this section, we describe in detail three such workflows: electrostatic embedding QM/MM in Section III A, polarizable embedding QM/MM in Section III B, and MTS QM/MM in Section III C.

A. Electrostatic Embedding QM/MM MD

Theoretical Basis

In electrostatic embedding QM/MM, the fixed point charges (or multipoles) of the MM subsystem generate an external field that polarizes the electron density of the QM sub-

system. The QM equations are thus solved in the presence of this field, but since the MM point charges are fixed, the MM subsystem is not explicitly polarized. A possible way of implementing QM/MM is to employ an additive scheme, in which the total time-independent Hamiltonian of the system,

$$H_{\mathrm{tot}} = H_{\mathrm{QM}} + H_{\mathrm{MM}} + H_{\mathrm{QM/MM}} \,, \tag{1}$$

yields the energies of the QM and MM subsystems and the interactions between them.

In QM/MM MD, all simulated particles, i.e., QM nuclei and MM atoms, are propagated with classical equations of motion, which implies invoking the Born–Oppenheimer approximation for the QM subsystem. Consequently, the nuclei move on a potential energy surface corresponding to a particular solution of the electronic QM equations for each nuclear configuration. In the context of Kohn-Sham density functional theory (KS-DFT), the energy contributions arising from the QM subsystem include the kinetic energy of the electrons and the potential energy arising from classical Coulombic electron–electron, nuclear–electron, and nuclear–nuclear interactions, as well as the exchange–correlation energy. On the other hand, the MM subsystem energy is determined by evaluating a force field, whose contributions are customarily divided into bonded (stretching, bending, and torsional energies) and non-bonded terms (van der Waals and electrostatic energies).

The QM/MM interaction Hamiltonian,

$$H_{\mathrm{QM/MM}} = V^{\mathrm{bonded}}_{\mathrm{QM/MM}} + V^{\mathrm{vdW}}_{\mathrm{QM/MM}} + V^{\mathrm{es}}_{\mathrm{QM/MM}} \,, \tag{2}$$

includes all interactions between the two subsystems, i.e., bonded and non-bonded van der Waals (vdW) and electrostatic (es) interactions. The MiMiC framework includes an implementation of the Hamiltonian electrostatic coupling scheme by Laio, VandeVondele, and Rothlisberger[42], where the electrostatic QM/MM interactions are split into short-range (sr) and long-range (lr) contributions,

$$V^{\mathrm{es}}_{\mathrm{QM/MM}} = V^{\mathrm{es,sr}}_{\mathrm{QM/MM}} + V^{\mathrm{es,lr}}_{\mathrm{QM/MM}} \,. \tag{3}$$

Short-range interactions are calculated using an exact expression

$$V^{\mathrm{es,sr}}_{\mathrm{QM/MM}} = \sum_{i=1}^{N^{\mathrm{sr}}_{\mathrm{MM}}} q_i \left(\int \mathrm{d}\mathbf{r}\, T^{(0)}_{\mathrm{mod}}(\mathbf{R}_{\mathrm{MM},i}, \mathbf{r}) \rho(\mathbf{r}) + \sum_{j=1}^{N_{\mathrm{QM}}} T^{(0)}_{\mathrm{mod}}(\mathbf{R}_{\mathrm{MM},i}, \mathbf{R}_{\mathrm{QM},j}) Z_j \right) \,, \tag{4}$$

where q_i is the point charge of the i-th MM atom, Z_j is the nuclear (or core) charge of the j-th QM atom, ρ is the electron density of the QM subsystem, and

$$T^{(0)}_{\mathrm{mod}}(\mathbf{R}_a, \mathbf{R}_b) = \frac{r^4_{\mathrm{cov},a} - |\mathbf{R}_b - \mathbf{R}_a|^4}{r^5_{\mathrm{cov},a} - |\mathbf{R}_b - \mathbf{R}_a|^5} \tag{5}$$

is the modified zeroth-order interaction tensor, where $r_{\mathrm{cov},a}$ corresponds to the covalent radius of atom a. This form of the interaction tensor effectively prevents electron spill-out, an artifact due to the lack of electronic Pauli repulsion between the QM and MM subsystems and the resulting over-polarization of electron density by nearby MM point charges[42].

Before proceeding to the treatment of the long-range interactions, we introduce a multi-index notation, which allows us to express the equations involving multipoles in a more compact manner. A multi-index α is a tuple consisting of three non-negative integers associated with the Cartesian components, i.e. $\alpha = (\alpha_x, \alpha_y, \alpha_z)$. The multi-index norm is defined as $|\alpha| = \alpha_x + \alpha_y + \alpha_z$, the multi-index factorial as $\alpha! = \alpha_x! \cdot \alpha_y! \cdot \alpha_z!$, and the multi-index partial derivative operator as $\partial^{\alpha}_{\mathbf{R}_b} = \partial^{|\alpha|} / \partial^{\alpha_x}_{R_{b,x}} \partial^{\alpha_y}_{R_{b,y}} \partial^{\alpha_z}_{R_{b,z}}$. Using this notation, the α-component of a tensor is denoted using square brackets as, e.g., $T^{[\alpha]}$, while a tensor is denoted with its rank in round parentheses, e.g., $T^{(0)}$.

Our implementation of the long-range interactions generalizes the original formulation[42] to include open-ended multipole expansions of the QM electrostatic potential[33]. For this, the QM multipoles are calculated as

$$M^{[\alpha]}_{\mathrm{QM}} = \int \mathrm{d}\mathbf{r}\, \rho(\mathbf{r}) \left(\mathbf{r} - \bar{\mathbf{R}}_{\mathrm{QM}}\right)^{\alpha} + \sum_{i=1}^{N_{\mathrm{QM}}} Z_i \left(\mathbf{R}_{\mathrm{QM},i} - \bar{\mathbf{R}}_{\mathrm{QM}}\right)^{\alpha} \,, \tag{6}$$

where α is a multi-index indicating the Cartesian component and $\bar{\mathbf{R}}_{\mathrm{QM}}$ is the origin of the expansion, e.g., the centroid of the QM subsystem $\sum_{j=1}^{N_{\mathrm{QM}}} \mathbf{R}_{\mathrm{QM},j} / N_{\mathrm{QM}}$. The electrostatic energy for the long-range interactions can then be expressed as

$$V^{\mathrm{es,lr}}_{\mathrm{QM/MM}} = \sum_{i=1}^{N^{\mathrm{lr}}_{\mathrm{MM}}} \sum_{|\alpha|=0}^{A_{\mathrm{QM}}} \frac{(-1)^{|\alpha|}}{\alpha!} q_i T^{[\alpha]}(\mathbf{R}_{\mathrm{MM},i}, \bar{\mathbf{R}}_{\mathrm{QM}}) M^{[\alpha]}_{\mathrm{QM}} \,, \tag{7}$$

where A_{QM} is the order of the expansion and $T^{[\alpha]}$ is a non-modified interaction tensor, defined through

$$T^{[\alpha]}(\mathbf{R}_a, \mathbf{R}_b) = \partial^{\alpha}_{\mathbf{R}_b} \frac{1}{|\mathbf{R}_b - \mathbf{R}_a|} \,. \tag{8}$$

This approach drastically reduces the overall computational cost by increasing the accuracy of the long-range contributions, allowing the selection of smaller short-range regions, whose interactions are substantially more expensive to compute. In particular, with a proper selection of the short-range cutoff, which dictates how many MM atoms are treated in short- and long-range domains, and the order for the multipole expansion, it is possible to reach the same accuracy as if all the electrostatic interactions were treated exactly, i.e. according to Eq. 4, yet, at a fraction of the cost[33].

A detailed theoretical treatment of the electrostatic embedding QM/MM scheme implemented using MiMiC can be found in Ref. 33. It includes explicit expressions for the polarizing potential acting on the QM subsystem due to MM atoms, i.e., the functional derivative of the QM/MM interaction energy with respect to the electron density, and for the forces acting on QM and MM atoms, i.e., the derivatives of the QM/MM interaction energy with respect to the atomic positions.

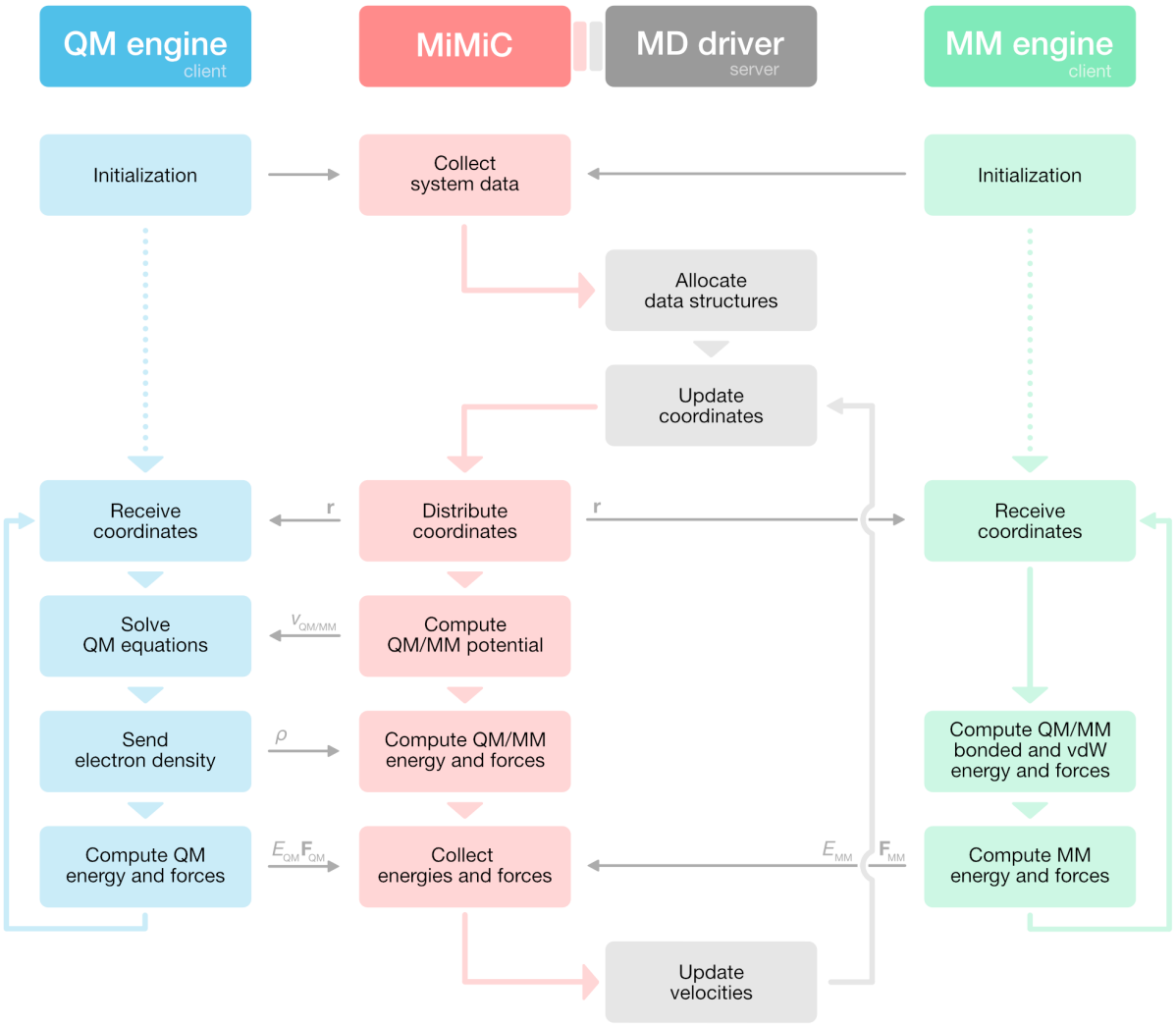

FIG. 4. Schematic workflow for electrostatic embedding QM/MM MD using MiMiC. Thick lines represent the workflow, while thin horizontal lines represent communication via MCL.

Simulation Workflow

Electrostatic embedding QM/MM is implemented using MiMiC by interfacing two external programs, a QM and an MM client, to an MD driver. The general simulation workflow is illustrated in Figure 4. After the initialization, consisting of data collection and allocation, the simulation enters the main MD loop. At each MD step, MiMiC distributes atomic coordinates among the clients, which then concurrently compute energy and forces that are collected by MiMiC and passed to the MD driver to update velocities and subsequently calculate a new set of coordinates.

The input to the MM client includes both QM and MM atoms, but all QM–QM and electrostatic QM–MM interactions must be excluded, as they are treated at the QM and QM/MM levels and thus calculated by a QM client and MiMiC, respectively. The MM client thus computes the potential energy of the MM subsystem and the remaining QM/MM interaction terms (Eq. 2), which are described purely at the MM level. The latter corresponds to van der Waals interactions between QM–MM atoms as well as all bonded terms that cross the QM/MM boundary and involve at least one MM atom. To saturate broken bonds on the QM side, boundary MM atoms bonded to a boundary QM atom are augmented with monovalent pseudopotentials.

The QM client solves the QM equations in the presence of the external MM potential computed by MiMiC. The client then sends the converged electron density to MiMiC, and finally proceeds to compute the QM forces, while MiMiC computes the corresponding electrostatic QM/MM contributions. Finally, MiMiC collects energies and forces from all client programs and forwards them to the MD driver, which then updates velocities and positions, and starts a new MD step.

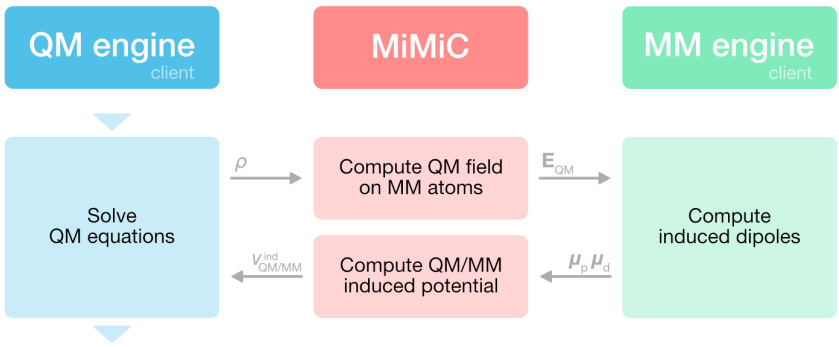

FIG. 5. Schematic workflow of the iterative cycle used in polarizable embedding QM/MM through MiMiC. Thin horizontal lines represent communication via MCL.

B. Polarizable Embedding QM/MM MD

Theoretical Basis

In electrostatic embedding, the QM system is polarized by the electric field generated by the fixed point charges of the MM subsystem. As such, this approach does not account for the explicit polarization of the MM subsystem, which is important in many cases and even crucial in some, e.g., in processes involving absorption or emission of photons and electron or proton transfer[43]. Polarizable embedding QM/MM schemes address this issue by using an MM force field that includes a polarizable component. For instance, in the frequently used induced point dipole model, the additional dynamic degrees of freedom are modeled using electric polarizabilities that describe an electronic charge distribution distorted by an electric field. Here, we briefly summarize the theoretical basis for polarizable embedding QM/MM MD using the polarizable AMOEBA[44] force field to model the MM subsystem (QM/AMOEBA). For further details, we refer to Refs. 45,46.

The electrostatic interactions in the AMOEBA force field are modeled using permanent multipoles, up to and including quadrupoles, and induced dipoles, from which electrostatic and polarization QM/MM interaction energies arise. The former is computed similarly as in electrostatic embedding QM/MM using modified interaction tensors (Eq. 5). The polarization energy is expressed in terms of induced dipoles and electric fields, thus making it necessary to first determine the induced dipoles through coupled polarization equations. A characteristic aspect of AMOEBA is its non-variational nature stemming from the fact that the electric field from the permanent multipoles used to determine the induced dipoles differs from the one used to calculate the polarization energy. The former is referred to as the *direct field* and the latter as the *polarization field*. The difference appears due to the use of different scaling factors and exclusion rules for each of these fields. Consequently, it is not possible to derive the polarization equations by simply requiring the stationarity of the energy functional with respect to the polarizable degrees of freedom. To avoid the complexity of non-variational optimization, a variational formulation has been adopted, resulting in two sets of polarization equations and, therefore, two sets of induced dipoles[45,46].

For the polarization QM/MM interaction energy in the QM/AMOEBA model, we follow the formulation by Loco *et al.*[45] in which it is expressed as

$$V^{\mathrm{pol}}_{\mathrm{QM/MM}} = \frac{1}{2}\boldsymbol{\mu}_{\mathrm{d}}^{\mathrm{T}}\mathbf{A}\boldsymbol{\mu}_{\mathrm{p}} - \frac{1}{2}\left(\boldsymbol{\mu}_{\mathrm{p}}^{\mathrm{T}}\mathbf{E}_{\mathrm{d}} + \boldsymbol{\mu}_{\mathrm{d}}^{\mathrm{T}}\mathbf{E}_{\mathrm{p}}\right) - \frac{1}{2}(\boldsymbol{\mu}_{\mathrm{p}} + \boldsymbol{\mu}_{\mathrm{d}})^{\mathrm{T}}\mathbf{E}_{\mathrm{QM}}\,, \tag{9}$$

where $\boldsymbol{\mu}_{\mathrm{d}}$ and $\boldsymbol{\mu}_{\mathrm{p}}$ are the AMOEBA induced dipoles due to the direct field, $\mathbf{E}_{\mathrm{d}}$, and the polarization field, $\mathbf{E}_{\mathrm{p}}$, respectively, as well as the field from the electrons and nuclei in the QM subsystem, $\mathbf{E}_{\mathrm{QM}}$, and $\mathbf{A}$ is the classical response matrix, which contains inverse polarizabilities on the diagonal blocks and Thole damped interaction tensors[44,47] on the off-diagonal blocks. This contribution thus includes the energies from the polarization of the MM subsystem and the mutual polarization interactions between the QM and MM subsystems.

On top of providing the MM energy and forces as in the electrostatic embedding workflow, the MM client now also has to solve two sets of polarization equations

$$\mathbf{A}\boldsymbol{\mu}_{\mathrm{d}} = \mathbf{E}_{\mathrm{d}} + \mathbf{E}_{\mathrm{QM}}\,, \tag{10a}$$

$$\mathbf{A}\boldsymbol{\mu}_{\mathrm{p}} = \mathbf{E}_{\mathrm{p}} + \mathbf{E}_{\mathrm{QM}}\,, \tag{10b}$$

to obtain the induced dipoles. In polarizable embedding QM/MM models, the induced dipoles depend on the QM electric field as seen in Eqs. 10 and thus have to be solved iteratively until self-consistency for electron density and induced dipoles is reached. This ensures fully self-consistent polarization between the QM and MM subsystems.

Simulation Workflow

Enabling fully polarizable QM/MM MD simulations using MiMiC requires modifications of the algorithm used to solve the QM equations. Achieving full self-consistency entails that the induced dipoles in the MM subsystem and the electron density in the QM subsystem are iteratively adjusted until both

are converged with respect to the field generated by the other subsystem. This iterative process is characteristic of most polarizable embedding QM/MM models. The general workflow of a polarizable embedding QM/MM MD simulation using MiMiC is quite similar to the one for electrostatic embedding QM/MM shown in Figure 4. It requires extending the *Solve QM equations* step with the iterative cycle illustrated in Figure 5.

The extended workflow can be summarized as follows. MiMiC receives an electron density from the QM client and uses it to compute the QM electric field at all MM atoms. The resulting field is sent to the MM client, which solves the polarization equations (Eqs. 10). The induced dipoles are then passed to MiMiC, which computes the corresponding induced energy and potential and relays them to the QM client, which proceeds with the next iteration. This procedure is repeated until the convergence criteria for the electron density and induced dipoles are met, thus solving the QM equations fully self-consistently with the polarizable MM subsystem.

The presented workflow is unique for the AMOEBA force field, but other polarizable models can be straightforwardly enabled with minimal or no changes in MiMiC.

C. Multiple Time Step QM/MM MD

Theoretical Basis

MTS algorithms reduce the cost of MD simulations by separating forces into *fast* and *slow* components, and integrating the latter less frequently, i.e., with a larger time step. Assuming a time-scale separation between these components, it is possible to split the classical Liouville operator of an N particle system as

$$\begin{aligned} iL &= \sum_{j=1}^{N} \left(\dot{\mathbf{q}}_j \cdot \nabla_{\mathbf{q}_j} + \mathbf{F}_j^{\text{fast}} \cdot \nabla_{\mathbf{p}_j} + \mathbf{F}_j^{\text{slow}} \cdot \nabla_{\mathbf{p}_j} \right) \\ &= iL_{\text{q}} + iL_{\text{p}}^{\text{fast}} + iL_{\text{p}}^{\text{slow}} \, . \end{aligned} \tag{11}$$

As introduced by Tuckerman, Berne, and Martyna[48], this allows the Trotter factorization of the classical time propagator

$$e^{iL\Delta t} = e^{iL_{\text{p}}^{\text{slow}}\left(\frac{\Delta t}{2}\right)} \left[e^{iL_{\text{p}}^{\text{fast}}\left(\frac{\Delta t}{2n}\right)} e^{iL_{\text{q}}\left(\frac{\Delta t}{n}\right)} e^{iL_{\text{p}}^{\text{fast}}\left(\frac{\Delta t}{2n}\right)} \right]^n e^{iL_{\text{p}}^{\text{slow}}\left(\frac{\Delta t}{2}\right)} \, , \tag{12}$$

where the terms of higher than second order are neglected. The operators in square brackets correspond to n inner propagation steps for the fast components with a time step $\Delta t/n$, and the ones outside to an outer step for the slow components, thus completing a full time step Δt.

The MTS approach can also be employed in Born–Oppenheimer MD, for which it can be reformulated in terms of higher and lower levels of theory owing to the observation that the differences between forces calculated with lower- or higher-level methods vary smoothly and slowly over time[49–51]. This is because quantum chemical methods usually only differ in the treatment of electron correlation (as well as exchange in DFT). In this QM$^{\text{low}}$–QM$^{\text{high}}$ MTS approach, the forces calculated by a low-level method (QM$^{\text{low}}$) are used as fast components, while the slow components are represented by a correction from a higher-level method (QM$^{\text{high}}$)

$$\mathbf{F}^{\text{fast}} = \mathbf{F}^{\text{low}} \, , \tag{13a}$$

$$\mathbf{F}^{\text{slow}} = \mathbf{F}^{\text{high}} - \mathbf{F}^{\text{low}} \, . \tag{13b}$$

Therefore, as QM$^{\text{high}}$ forces need to be calculated only so often, adopting this scheme effectively reduces the overall computational cost of high-level methods by at least a factor of five[50].

In the context of electrostatic embedding QM/MM, the scheme introduced above can be implemented by choosing

$$\mathbf{F}^{\text{low}} = \mathbf{F}_{\text{QM}}^{\text{low}} + \mathbf{F}_{\text{QM/MM}}^{\text{low}} + \mathbf{F}_{\text{QM/MM}}^{\text{bonded+vdW}} + \mathbf{F}_{\text{MM}} \, , \tag{14a}$$

$$\mathbf{F}^{\text{high}} = \mathbf{F}_{\text{QM}}^{\text{high}} + \mathbf{F}_{\text{QM/MM}}^{\text{high}} + \mathbf{F}_{\text{QM/MM}}^{\text{bonded+vdW}} + \mathbf{F}_{\text{MM}} \, , \tag{14b}$$

where $\mathbf{F}_{\text{QM}}^{\text{low}}$ and $\mathbf{F}_{\text{QM/MM}}^{\text{low}}$ are the low-level forces arising from the QM and QM/MM terms, respectively, just as $\mathbf{F}_{\text{QM}}^{\text{high}}$ and $\mathbf{F}_{\text{QM/MM}}^{\text{high}}$ represent their high-level counterparts. Note that in the calculation of Eq. 13b, within an electrostatic embedding QM/MM scheme, the high- and low-level bonded and vdW QM/MM forces as well as the MM forces cancel out. The latter implies that the MM atoms always evolve with the inner time step $\Delta t/n$.

MTS-generated trajectories are, by construction, at the high level of theory with outer time step Δt, regardless of the low-level method used. The choice of the latter only determines the magnitude of the correction and hence the oscillations of the slow force component over time, which dictate the maximal ratio between the inner and the outer steps that can be applied to generate stable dynamics.

Simulation Workflow

Performing a full MTS MD step requires executing n inner steps at a lower level of theory (as shown in Eq. 12) followed by a single high-level outer step. Figure 6 depicts a schematic workflow of an MTS QM/MM MD algorithm using MiMiC. At every step, MiMiC collects the low-level forces from the QM and MM clients and computes the low-level QM/MM force contributions (Eq. 14a). If the number of steps is not a multiple of n, velocities and coordinates are updated at the low level with a time step $\Delta t/n$, otherwise, at every n-th step, the outer loop is undertaken. In the outer loop, MiMiC additionally collects the high-level forces and computes the high-level QM/MM force contributions (Eq. 14b), and proceeds to calculate the slow forces. These are used by the MD driver to update the velocities, thus completing a full MD cycle with the time step Δt at the high level of theory. Note that with MiMiC, it is straightforward to use two different QM client programs for the low- and high-level forces.

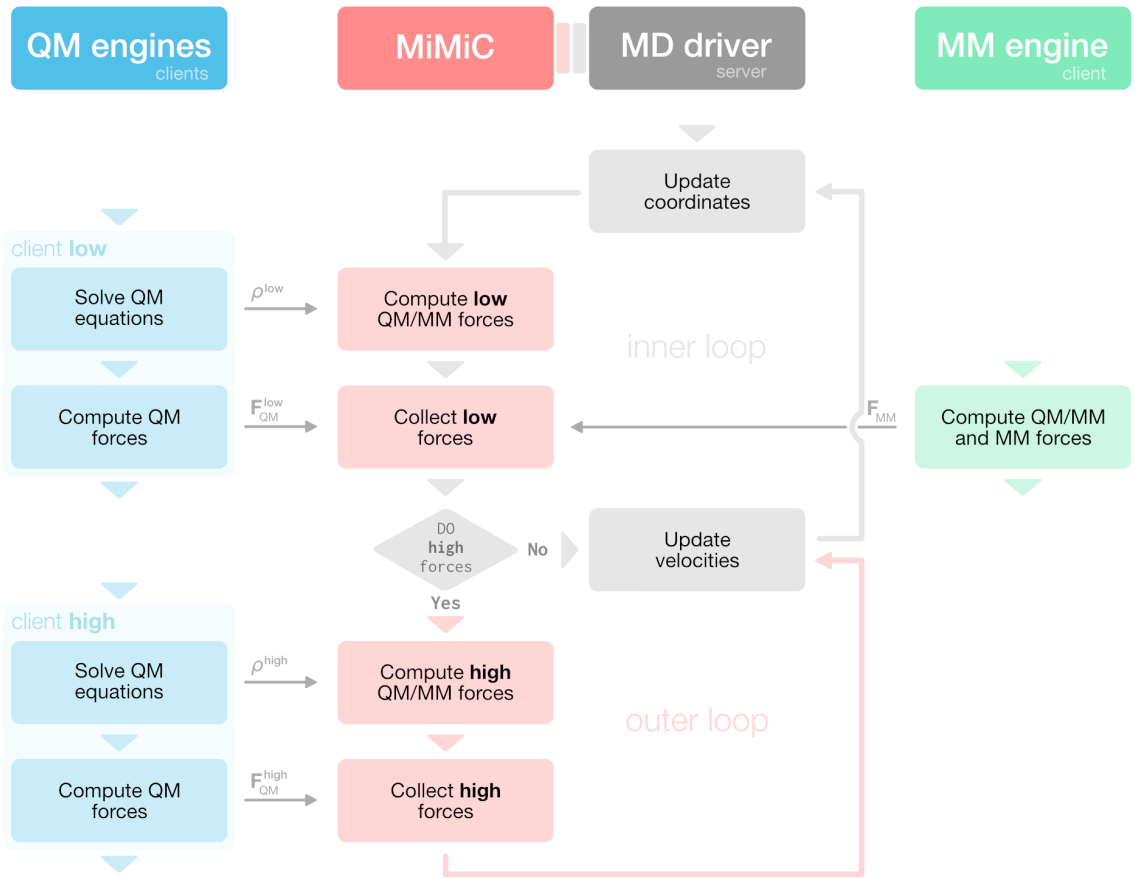

FIG. 6. Schematic workflow for MTS QM/MM MD using MiMiC.

IV. FEATURES AND CLIENT PROGRAMS

In this section, we briefly discuss the present and upcoming features of MiMiC. As multiscale QM/MM simulations remain the focus of our immediate development plans, we place particular emphasis on new QM and MM client programs that are being coupled to the framework.

The first development version of the MiMiC framework was released in 2022 on the occasion of the CECAM Flagship School *Multiscale Molecular Dynamics with MiMiC*[52] held at the CECAM Headquarters at EPFL in Lausanne, Switzerland. It implements the DFT-based electrostatic embedding QM/MM MD method illustrated in Subsection III A, using GROMACS[53,54] as the MM client and CPMD[55] as the QM client and MD driver. Furthermore, it also supports MTS QM/MM MD (Subsection III C), and, through the PLUMED library, QM/MM-based enhanced sampling simulations and free energy calculations[56–58].

The *CPMD* program[55], distributed under the MIT license since 2023, presents a number of interesting features that made it our initial choice as a QM client and MD driver. It is a highly parallelized PW/pseudopotential implementation of KS-DFT that provides access to a wide variety of exchange–correlation functionals[59–61], together with a computationally efficient treatment of hybrid functionals[62,63], and second-order Møller–Plesset perturbation theory (MP2)[64]. Moreover, it supports density functional perturbation theory[65], linear response[66] and real-time time-dependent DFT (TDDFT), and the restricted open-shell KS (ROKS) formalism[67]. As an MD driver, CPMD offers a broad range of features, as it implements both Born–Oppenheimer and Car–Parrinello MD, as well as nonadiabatic MD, using either the Ehrenfest scheme[68] or Tully's fewest-switches surface hopping[69]. The latter can also be performed in the presence of an explicit external field[70] with the possibility of performing local control simulations[71]. In addition, nuclear quantum effects can be included within a path-integral formalism[72]. CPMD also provides access to various accurate and efficient thermostats[73–77] and an MTS acceleration algorithm via an extension of the microcanonical reversible reference system propagation algorithm (rRESPA) by Tuckerman, Berne, and Martyna[48], which also allows performing NVT-MTS propagation[50,78].

Our initial choice for an MM client was *GROMACS*[54], which is a free and open-source software suite for classical MD, distributed under LGPL version 2.1, whose developers aimed to provide the highest possible performance and efficiency on any hardware[53]. It implements a native heterogeneous parallelization setup, using both CPUs and GPUs, with the possibility to offload all force components to GPUs in single-precision calculations. The latest

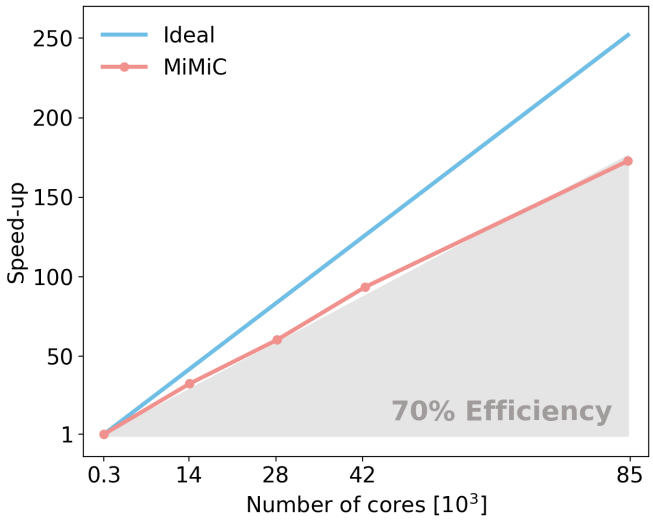

FIG. 7. Parallel scalability of a MiMiC-based QM/MM implementation using GROMACS as the MM client and CPMD as the QM client and MD driver. The system comprised 130 828 atoms, out of which 142 were treated at the QM level using the hybrid B3LYP functional. The simulations were run on the CPU partition of the JUWELS cluster at the Jülich Supercomputing Centre. Adapted from Raghavan *et al.*[84], licensed under CC BY 4.0.

versions[79] also introduced new direct GPU–GPU communication and extended GPU integration, enabling excellent performance across multiple GPUs and efficient multi-node parallelization. Moreover, GROMACS supports a large variety of popular biomolecular force fields, including AMBER[80], CHARMM[81], GROMOS[82], and OPLS[83], and provides a wide array of well-documented pre- and post-processing tools. New users can also benefit from an extensive range of tutorials, documentation, and an active user community.

Combining the benefits of these external programs, the initial MiMiC-based QM/MM implementation achieved high performance in MD simulations of large and complex biomolecular systems. A recent MiMiC-based QM/MM MD study[84] demonstrated strong scaling up to 84 672 CPU cores of the JUWELS cluster[85] at the Jülich Supercomputing Centre (corresponding to ∼77% of the available nodes) while maintaining a parallel efficiency of about 70% (see Fig. 7). The study compared the attainable throughput for QM/MM MD of two systems containing different QM subsystem sizes. It focused on the solvated isocitrate dehydrogenase-1 (IDH1) enzyme (130 828 atoms), which contained 142 atoms in its QM subsystem, and the p38α mitogen-activated protein kinase (169 550 atoms), which contained a smaller QM subsystem of 46 atoms. For both of the systems studied, two simulations were launched in which either the BLYP or the hybrid B3LYP exchange–correlation functional was used to treat the QM subsystem. The QM/MM MD simulation of the system with a larger QM subsystem (IDH1) reached a throughput of 0.74 ps/day on 84 672 CPU cores while employing the B3LYP hybrid functional. In comparison, the simulation of the system with the smaller QM subsystem (p38α) achieved a throughput of 4.8 ps/day with B3LYP and maintained a parallel efficiency above 70% up to 12 288 CPU cores. Furthermore, using the BLYP functional, the large QM subsystem simulation realized a throughput of 5.4 ps/day using 5 184 CPU cores, while the smaller QM subsystem saw a throughput of 21 ps/day using just 384 CPU cores. These simulations demonstrate that the current MiMiC-based QM/MM implementation allows for subnanosecond-scale QM/MM MD thanks to the unhindered scaling capabilities of CPMD. Given sufficient computational resources, converged free energy values of biological systems similar to those highlighted here could be attained. For further examples of biomolecular system studies with MiMiC, including more information about IDH1, see Section V.

A. Upcoming Features

An important advantage of MiMiC is the possibility to implement new features directly within the framework itself, making them client-agnostic, and thus available to use together with every supported client program. As a number of development endeavors are currently in progress, here we describe a few features that are expected to be available soon.

These efforts include the implementation of dynamically generated restrained electrostatic potential (D-RESP) charges[86], extending the original approach to also fit higher-order atom-centered multipoles. Such an extension will also make it possible to implement different coupling schemes for electrostatic interactions, for example, by introducing a three-level approach[87], which would significantly increase electrostatic embedding efficiency, particularly, for elongated QM subsystems. In addition, the implementation of D-RESP charges will also serve as an on-the-fly analysis tool for tracking ongoing changes in the electronic structure and will also enable charge-driven enhanced sampling[88]. Another important addition, building on D-RESP, will be the development of QM/MM force-matching schemes, such as those in Refs. 89 and 90, in which the total forces on the QM atoms, as well as the QM electrostatic potential and field, are extracted during a QM/MM MD simulation, for a set of sampled configurations. These quantities are then used to parameterize standard or polarizable biophysical force fields for the QM subsystem. Such QM-matched force fields can then be employed to achieve more extensive sampling in a fully classical MD simulation of the whole system.

A different extension of capabilities in MiMiC will be based on enhancing the MTS workflow illustrated in Section III C, which unlocks a plethora of possible combinations of different high- and low-level methods, including wave function, DFT, semiempirical, force field, and ML models[51]. A future release of MiMiC will also include an adaptive ML-accelerated MTS QM/MM scheme where the ML model is retrained on the fly whenever the system accesses new regions of configurational space (e.g. during a chemical reaction) for which the original ML model has insufficient predictive power. In addition, this adaptive scheme allows for a dynamic training set build-up during the run, thus, circumventing the need for extensive and computationally demanding a priori generation of high-level reference data.

Concerning the current developments of new interfaces to external client programs, we now primarily focus on the following three objectives. First, we strive to enable large-scale QM/MM MD simulations on current and future (pre-)exascale supercomputers by coupling to highly scalable GPU-enabled QM client programs, namely, CP2K[91], DFT-FE[92,93], and Quantum ESPRESSO[94,95], as well as making available more accurate wave function methods for QM/MM, as implemented in CP2K and CFOUR[96,97]. Finally, we aim to implement high-performance, GPU-accelerated polarizable embedding QM/MM (see Section III B) via polarizable force fields as implemented in Tinker-HP[98,99] and OpenMM[100–102].

To achieve these goals, work on interfaces with several client programs is currently underway. In the remainder of this section, we review the distinctive features that make them ideal candidates for integration into the framework. As new programs will be coupled to MiMiC, the pre-processing toolkit MiMiCPy will also be augmented accordingly.

CP2K[91] is a free and open-source software package, distributed under GNU General Public License (GPL) version 2.0 or later, for simulations of solid-state and (bio)molecular systems, giving access to a broad range of electronic structure methods. It focuses on ensuring excellent performance, thanks to algorithms developed for modern HPC systems, also including GPU support. In particular, the electronic structure module Quickstep[91,103,104] provides an efficient infrastructure of integral routines and optimization algorithms for a wide spectrum of methods, including orbital-free DFT, KS-DFT, and MP2. Quickstep is based on mixed Gaussian and PW (GPW) basis sets and their augmented extension (GAPW), where auxiliary PW basis sets are used within a GTO scheme. Recently, CP2K's linear scaling DFT implementation was used for calculations on the SARS-CoV-2 spike proteins with almost 83 million atoms, for which it scaled up to 4 400 NVIDIA A100 GPUs with a parallel efficiency of over 80%[105].

DFT-FE[92,93] is a recently developed free and open-source code, distributed under LGPL version 2.1 or later, that implements real-space KS-DFT based on a spatially adaptive and systematically convergent higher-order spectral finite-element (FE) basis. It accommodates both norm-conserving pseudopotentials and all-electron calculations, with ongoing implementation efforts to support ultrasoft projector-augmented-wave pseudopotentials. DFT-FE enables fast and accurate large-scale DFT calculations on generic materials systems involving many tens of thousands of electrons on both many-core CPU and hybrid CPU-GPU architectures. The good parallel scalability and efficient use of GPUs has enabled systematically convergent DFT calculations with low wall times, e.g., an ab initio MD step of systems containing 10 000-20 000 electrons can be completed in wall times of <1 minute[93]. In addition to the support of NVIDIA GPUs through CUDA, DFT-FE's GPU porting layer was recently extended to support AMD GPUs through HIP, with its exascale capability recently demonstrated through scaling up to 8 000 GPU nodes (32 000 AMD MI250X GPUs) of the Oak Ridge Leadership Computing Facility Frontier exascale supercomputer[106].

Quantum ESPRESSO[107] is an integrated and free suite of programs, distributed under GPL version 2.0 or later, designed for first-principles electronic structure calculations and materials modeling at the nanoscale. In recent years, it has been refactored and organized in multiple layers of codes[95], consisting of loosely coupled modules and libraries, tailored to be independently extensible and maintainable. Recent versions are composed of a core computational layer for DFT simulations, based on PWs and pseudopotentials, containing a quantum-engine package to solve self-consistent KS equations for periodic systems (PWscf). To achieve high performance on different architectures, Quantum ESPRESSO features a layered, fine-tuneable parallelization scheme based on MPI and OpenMP, but also GPU support. Porting to GPU has been achieved through CUDA Fortran and, to an increasing degree, directive-based programming models (OpenACC and OpenMP) for offloading to heterogeneous platforms, making it as architecture-agnostic as possible. The optimal combination of all available parallelism levels currently allows the Quantum ESPRESSO suite to achieve a substantial increase in performance and efficiency[95].

CFOUR[96,97] is a program package, distributed freely for academic users, specialized in high-accuracy wave function-based methods, with applications in thermodynamic, spectroscopic, and kinetic phenomena of small to medium-sized molecules. It implements Hartree–Fock (HF), multiconfigurational complete active space (CAS) SCF, and a multitude of single-reference post-HF methods. These include Møller–Plesset perturbation theory up to fourth-order and coupled cluster (CC) approximations ranging from CC2 to CCSDTQ. A particular strength of CFOUR is its ability to compute analytic energy derivatives for most of the implemented methods. The program is also able to treat excited states via the equation-of-motion CC theory (EOM-CC) with analytic energy derivatives for some of the approximations. An initial interface between CFOUR and MiMiC was presented recently[108], showcasing MP2, CCSD(T), and CASSCF within both ab initio QM and electrostatic embedding QM/MM MD. This work also demonstrated the utility of the QM^{low}–QM^{high} MTS implemented in CPMD by performing BLYP–CCSD(T) MTS MD simulations. The next steps for this interface include further development of the hierarchical QM^{low}–QM^{high} MTS schemes, also exploiting the ML acceleration capabilities.

Tinker-HP[98,99] is a powerful, massively MPI parallelized MD package, distributed freely for academic institutions, that supports simulations using standard fixed point-charge and advanced polarizable force fields. One of its core strengths is the ability to run long timescale simulations of very large biomolecular systems using polarizable force fields. This is facilitated by a parallel 3D spatial decomposition tailored for point-dipole polarizable models and the use of efficient iterative and non-iterative polarization solvers. Additionally, thanks to recent expansion to GPUs through OpenACC and CUDA, Tinker-HP performs efficiently on both CPU and GPU architectures[99].

OpenMM[100–102] is a free and open source classical MD package, distributed under LGPL version 3.0 and MIT licenses, designed to be simple and easy to use, with a focus

on extensibility. It offers many common classical force fields, and users can easily add support for new types of force fields via plugins in a modular and extensible way. Notable plugins include polarizable force fields (AMOEBA[44] and Drude oscillator[109] models) and, more recently, ML potentials[102]. Moreover, it also provides several custom force classes that make it possible for users to specify arbitrary algebraic expressions for force components. It performs efficiently on different hardware platforms, especially GPU-based architectures, for both single- and double-precision calculations.

V. ILLUSTRATIVE APPLICATIONS

Recent applications of MiMiC to study biological systems demonstrate its utility in multiscale modeling[34]. All results of the studies presented in this section were acquired using the development version of MiMiC that features an electrostatic embedding QM/MM implementation using CPMD and GROMACS.

One exemplary class of systems that has been extensively investigated through MiMiC are transporter proteins, which play a key role in cellular homeostasis. These systems are inherently multiscale in nature: transporter proteins are embedded in lipid bilayers, resulting in a large system, but they are often involved in the movement of ions, for which an accurate QM description is required. The first study focused on the CLC-type Cl^-/H^+ transporter protein from *Escherichia coli*, CLC-ec1, embedded in a solvated lipid bilayer[110] (Fig. 8a). The protein exchanges protons with chlorides and many other anions, but is inhibited by fluoride. Chiariello *et al.*[110] gained insight on the molecular basis of fluoride inhibition of the protein by MiMiC-based QM/MM MD and well-tempered metadynamics (via PLUMED[57,111]), at the BLYP and B3LYP levels of theory. Since the fluoride binding mechanism involves proton transfer phenomena, the use of a quantum mechanical approach is imperative[110]. The QM subsystem consisted of 106 atoms encompassing the glutamate gating region while the remainder of the protein, water, ions, and lipid bilayer (together 150 925 atoms) were described by a fixed point-charge force field enabling electrostatic embedding. Subsequent MiMiC-based MD and metadynamics simulations considered a related protein, the CLC^F antiporter (solvated and embedded in a lipid bilayer for a total of 165 732 atoms), though with a smaller QM subsystem of 36 atoms, using the same levels of theory to model the full exchange between anion and proton (Fig. 8b)[112,113]. The simulations provided clues to the intriguing question of why a protein structurally similar to CLC-ec1, such as CLC^F, transports fluoride instead of being inhibited by it. More recently, MiMiC-based QM/MM MD simulations have been extended to DgoT, a bacterial homolog of human SLC17 organic anion transporters[114,115]. Here, the simulations shed light on a key proton transfer process occurring in DgoT's transport cycle in which bond breaking and formation occur.

Ion channels represent another class of large membrane-protein systems. A recent study[116] employed MiMiC to investigate calcium conduction in the α-amino-3-hydroxy-

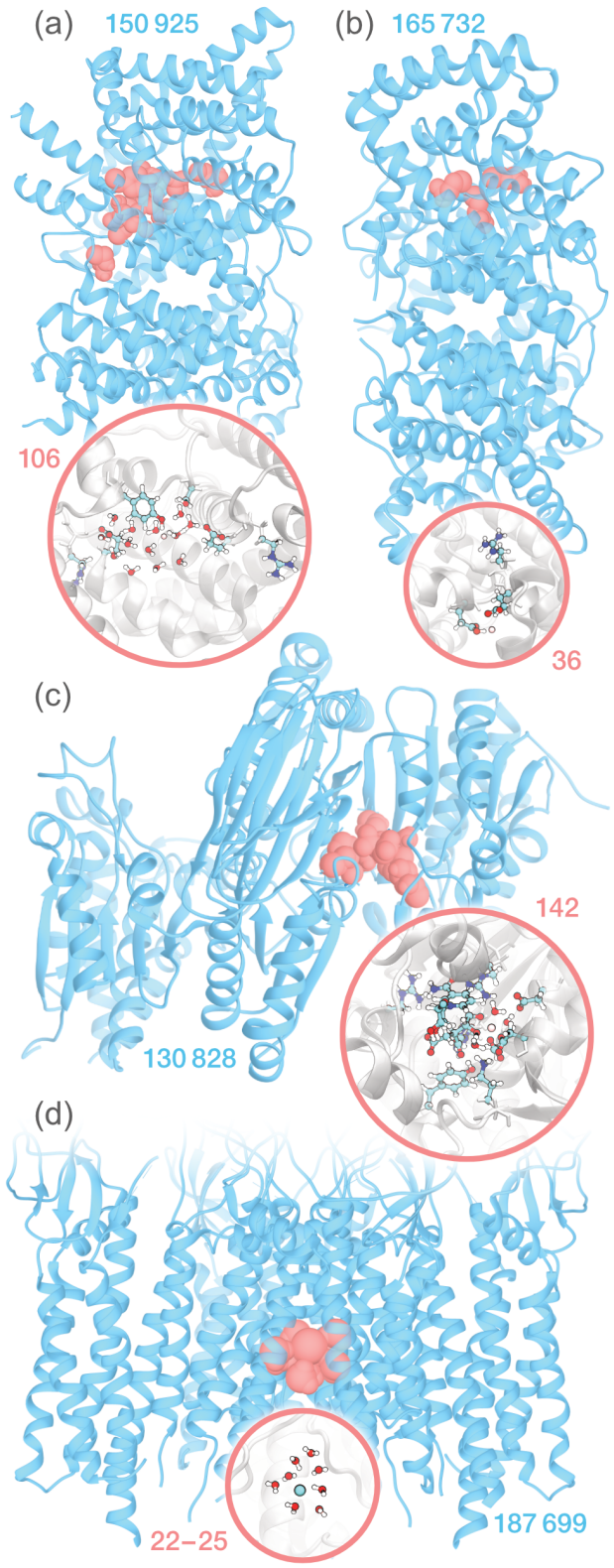

FIG. 8. Biologically relevant systems simulated using MiMiC: (a) the CLC-ec1 (Cl^-/H^+) antiporter protein[34] (b) the CLC^F antiporter protein[112]; (c) the IDH1 enzyme[84] and (d) the AMPA receptor[116]. QM subsystems are shown within the circular insets, with coral and blue numbers representing the number of QM and MM atoms, respectively.

5-methyl-4-isoxazolepropionic acid receptors (AMPARs), neurotransmitter-activated cation channels ubiquitously expressed in vertebrate brains (Fig. 8d). In particular, a Ca^{2+} ion and its first hydration shell were treated at the BLYP or B3LYP levels giving a QM subsystem of between 22 and 25 atoms depending on the simulation, while all other water molecules, as well as protein and lipids, were described classically for a total of 187 699 atoms. In this study, thanks to the QM/MM partitioning, the large electric field of the protein was modeled, as well as the structure and dynamics of calcium-bound water molecules. Moreover, the study provided data to assess the accuracy of a newly developed force field for Ca^{2+}, an ion that is notoriously difficult to describe at the force field level.

MiMiC-based QM/MM simulations have also been used in drug design studies. A combined force field metadynamics and QM/MM simulation study performed in 2020[117] investigated the residence time of iperoxo to the human muscarinic acetylcholine receptor 2 (M_2). Iperoxo serves as a known M_2 antagonist, and therefore its study yields insight into potential therapeutics aimed at M_2 receptors in the heart where they regulate the sinus rhythm of the heartbeat rate. The ligand unbinding rate constant k_{off}, and its inverse, the residence time, are crucial pharmacological parameters[118]. An accurate estimation of k_{off} requires a precise calculation of the free energy transition state, and not just of the binding free energies and affinities. The QM/MM model places the ligand, key side chains in the binding pocket, and nearby water molecules in the QM subsystem described at the B3LYP level. The MiMiC-based calculations here point to limitations of the force field in the prediction of this important quantity that is due to the lack of charge polarization at the transition state.

A more recent study[84] further highlights the potentiality of MiMiC for drug design. It focused on the IDH1 enzyme (Fig. 8c), which is a crucial actor within the Krebs cycle of cellular respiration as it serves as a catalyst in the conversion of isocitrate to α-ketoglutarate[119]. The IDH1 active site includes a Mg^{2+} ion, the NADH+ cofactor, the ligand, and important amino acid residues that all together make up the Michaelis complex. To investigate the enzymatic reaction, which invariably involves bond breaking and formation processes, quantum mechanics is required. The QM subsystem encompasses the active site, comprising 142 atoms, and the MM subsystem includes the remainder of the enzyme, water, and ions (all represented through a fixed point-charge force field) for a total of 130 828 atoms. The conclusions drawn from these simulations turned out to be consistent with previous studies on the same system.

VI. OUTLOOK

We presently devote the majority of our efforts towards multiscale simulations capable of exploiting the vast computational resources available on current and upcoming exascale supercomputers[120–123]. MiMiC is prepared for the future not only in terms of scalability, but also, the possibility of running the client programs independently and concurrently on separate resources makes it an ideal framework for modular, heterogeneous HPC architectures[124–126]. Specifically, each program can use the type and amount of resources that are optimal for its performance, and the different programs can easily run on different partitions or modules of a supercomputer. Hybrid CPU/GPU technology is already prevalent on most state-of-the-art supercomputers, while modular architectures have been adopted by multiple HPC centers. A prominent example is the first European exascale supercomputer JUPITER that will feature multiple modules, including a general-purpose CPU partition and a high-performance GPU-accelerated partition[127]. Moreover, it is expected that future technologies, such as quantum or neuromorphic computing, will eventually be integrated into traditional HPC architectures. As these novel computing technologies become available, MiMiC is virtually ready to incorporate them into its simulation workflows.

In terms of new features, we aim to extend the range of multiscale techniques to address scientific challenges beyond the scope of current QM/MM methods. One of our development directions is to introduce additional versatility in the QM subsystem treatment by enabling larger, more accurate, and yet efficient simulations through fragment-based methods and multilayered QM/QM/MM models. To model larger systems over longer time scales, we will also consider CG and continuum models (CM), enabling, for example, the implementation of MM/CG, QM/MM/CG, and QM/MM/CM schemes. Another promising way to achieve more efficient calculations is the introduction of ML-based force fields[128,129], which will give access to, e.g., ML/MM simulations. Along the same lines is the development of ML-based schemes where a subsystem is described by an ML model that is retrained on the fly based on, e.g., QM calculations. Among the other desired extensions is the implementation of schemes that allow subsystems to adjust to changes over the course of a simulation, such as adaptive QM/MM[130]. Finally, we would like to take MiMiC beyond the scope of biomolecular systems, by implementing necessary features for systems within the material science domain. Here, multiscale simulations pose different challenges, but they also open up numerous opportunities to provide a fundamental understanding of various processes, for which current models would not be applicable.

We are confident that our efforts to push the performance of multiscale simulations through advanced parallel algorithms and methods, in conjunction with the advances in computational chemistry software interfaced with MiMiC, will eventually open up new avenues of exploration to address exceptionally challenging scientific problems.

ACKNOWLEDGMENTS

The authors are grateful to Viacheslav Bolnykh for his essential foundational contributions during the initial phase of the project. The authors also thank Till Kirsch, Michele Cascella, and Jürgen Gauss for the CFOUR interface, and Maria Gabriella Chiariello, Florian Schackert, and Wenping Lyu for their contributions to the studies showcased in the Il-

lustrative Applications Section. BR, DM, EI, and PC thank the Helmholtz European Partnering program (*Innovative high-performance computing approaches for molecular neuromedicine*) for funding. PC thanks the Deutsche Forschungsgemeinschaft (DFG) RU2518 DynIon and the computing time granted by the JARA Vergabegremium on the JARA partition of the supercomputer JURECA at Forschungszentrum Jülich. UR acknowledges funding from the Swiss National Science Foundation via the NCCR MUST and individual grants Nos 200020-185092 and 200020-219440, as well as computing time from the Swiss National Computing Centre CSCS. JMHO gratefully acknowledges the financial support from VILLUM FONDEN (Grant No. VIL29478).

AUTHOR DECLARATIONS

A. Conflict of Interest

The authors have no conflicts to disclose.

DATA AVAILABILITY

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

REFERENCES

[1]W. Shinoda, R. DeVane, and M. L. Klein, "Computer simulation studies of self-assembling macromolecules," Curr. Opin. Struct. Biol. **22**, 175–186 (2012).

[2]L. R. Kjølbye, G. P. Pereira, A. Bartocci, M. Pannuzzo, S. Albani, A. Marchetto, B. Jiménez-García, J. Martin, G. Rossetti, M. Cecchini, S. Wu, L. Monticelli, and P. C. T. Souza, "Towards design of drugs and delivery systems with the Martini coarse-grained model," QRB Discovery **3**, e19 (2022).

[3]J. Jin, A. J. Pak, A. E. P. Durumeric, T. D. Loose, and G. A. Voth, "Bottom-up Coarse-Graining: Principles and Perspectives," J. Chem. Theory Comput. **18**, 5759–5791 (2022).

[4]L. Borges-Araújo, I. Patmanidis, A. P. Singh, L. H. S. Santos, A. K. Sieradzan, S. Vanni, C. Czaplewski, S. Pantano, W. Shinoda, L. Monticelli, A. Liwo, S. J. Marrink, and P. C. T. Souza, "Pragmatic Coarse-Graining of Proteins: Models and Applications," J. Chem. Theory Comput. **19**, 7112–7135 (2023).

[5]A. Warshel and M. Levitt, "Theoretical studies of enzymic reactions: Dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme," J. Mol. Biol. **103**, 227–249 (1976).

[6]U. C. Singh and P. A. Kollman, "A combined ab initio quantum mechanical and molecular mechanical method for carrying out simulations on complex molecular systems: Applications to the $CH_3Cl + Cl^-$ exchange reaction and gas phase protonation of polyethers," J. Comput. Chem. **7**, 718–730 (1986).

[7]M. J. Field, P. A. Bash, and M. Karplus, "A combined quantum mechanical and molecular mechanical potential for molecular dynamics simulations," J. Comput. Chem. **11**, 700–733 (1990).

[8]H. M. Senn and W. Thiel, "QM/MM Methods for Biomolecular Systems," Angew. Chem. Int. Ed. **48**, 1198–1229 (2009).

[9]E. Brunk and U. Rothlisberger, "Mixed Quantum Mechanical/Molecular Mechanical Molecular Dynamics Simulations of Biological Systems in Ground and Electronically Excited States," Chem. Rev. **115**, 6217–6263 (2015).

[10]U. N. Morzan, D. J. Alonso de Armiño, N. O. Foglia, F. Ramírez, M. C. González Lebrero, D. A. Scherlis, and D. A. Estrin, "Spectroscopy in Complex Environments from QM–MM Simulations," Chem. Rev. **118**, 4071–4113 (2018).

[11]F. Lipparini and B. Mennucci, "Hybrid QM/classical models: Methodological advances and new applications," Chem. Phys. Rev. **2** (2021), 10.1063/5.0064075.

[12]A. W. Götz, M. A. Clark, and R. C. Walker, "An extensible interface for QM/MM molecular dynamics simulations with AMBER," J. Comput. Chem. **35**, 95–108 (2014).

[13]M. C. R. Melo, R. C. Bernardi, T. Rudack, M. Scheurer, C. Riplinger, J. C. Phillips, J. D. C. Maia, G. B. Rocha, J. V. Ribeiro, J. E. Stone, F. Neese, K. Schulten, and Z. Luthey-Schulten, "NAMD goes quantum: an integrative suite for hybrid simulations," Nat. Methods **15**, 351–354 (2018).

[14]V. W. D. Cruzeiro, Y. Wang, E. Pieri, E. G. Hohenstein, and T. J. Martínez, "TeraChem protocol buffers (TCPB): Accelerating QM and QM/MM simulations with a client–server model," J. Chem. Phys. **158**, 044801 (2023).

[15]H. L. Woodcock, B. T. Miller, M. Hodoscek, A. Okur, J. D. Larkin, J. W. Ponder, and B. R. Brooks, "MSCALE: A General Utility for Multiscale Modeling," J. Chem. Theory Comput. **7**, 1208–1219 (2011).

[16]C. Ma, L. Martin-Samos, S. Fabris, A. Laio, and S. Piccinin, "QM-MMW: A wrapper for QM/MM simulations with Quantum ESPRESSO and LAMMPS," Comput. Phys. Commun. **195**, 191–198 (2015).

[17]J. Torras, E. Deumens, and S. B. Trickey, "Software Integration in Multiscale Simulations: the PUPIL System," J. Comput.-Aided Mater. Des. **13**, 201–212 (2006).

[18]J. Torras, B. P. Roberts, G. M. Seabra, and S. B. Trickey, "PUPIL: A Software Integration System for Multi-Scale QM/MM-MD Simulations and Its Application to Biomolecular Systems," in *Combined Quantum Mechanical and Molecular Mechanical Modelling of Biomolecular Interactions*, Adv. Protein Chem. Struct. Biol., Vol. 100, edited by T. Karabencheva-Christova (Elsevier, 2015) pp. 1–31.

[19]J. Řezáč, "Cuby: An integrative framework for computational chemistry," J. Comput. Chem. **37**, 1230–1237 (2016).

[20]A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—a Python library for working with atoms," J. Phys.: Condens. Matter. **29**, 273002 (2017).

[21]P. Altoè, M. Stenta, A. Bottoni, and M. Garavelli, "A tunable QM/MM approach to chemical reactivity, structure and physico-chemical properties prediction," Theor. Chem. Acc. **118**, 219–240 (2007).

[22]O. Weingart, A. Nenov, P. Altoè, I. Rivalta, J. Segarra-Martí, I. Dokukina, and M. Garavelli, "COBRAMM 2.0 — A software interface for tailoring molecular electronic structure calculations and running nanoscale (QM/MM) simulations," J. Mol. Model. **24** (2018), 10.1007/s00894-018-3769-6.

[23]S. Metz, J. Kästner, A. A. Sokol, T. W. Keal, and P. Sherwood, "ChemShell—a modular software package for QM/MM simulations," Wiley Interdiscip. Rev. Comput. Mol. Sci. **4**, 101–110 (2014).

[24]Y. Lu, M. R. Farrow, P. Fayon, A. J. Logsdail, A. A. Sokol, C. R. A. Catlow, P. Sherwood, and T. W. Keal, "Open-Source, Python-Based Redevelopment of the ChemShell Multiscale QM/MM Environment," J. Chem. Theory Comput. **15**, 1317–1328 (2019), pMID: 30511845.

[25]E. G. Kratz, A. R. Walker, L. Lagardère, F. Lipparini, J.-P. Piquemal, and G. Andrés Cisneros, "LICHEM: A QM/MM program for simulations with multipolar and polarizable force fields," J. Comput. Chem. **37**, 1019–1029 (2016).

[26]H. Gökcan, E. A. Vázquez-Montelongo, and G. A. Cisneros, "LICHEM 1.1: Recent Improvements and New Capabilities," J. Chem. Theory Comput. **15**, 3056–3065 (2019).

[27]B. Zhang, D. Altarawy, T. Barnes, J. M. Turney, and H. F. I. Schaefer, "Janus: An Extensible Open-Source Software Package for Adaptive QM/MM Methods," J. Chem. Theory Comput. **15**, 4362–4373 (2019).

[28]T. A. Barnes, E. Marin-Rimoldi, S. Ellis, and T. D. Crawford, "The MolSSI Driver Interface Project: A framework for standardized, on-the-fly

interoperability between computational molecular sciences codes," Comput. Phys. Commun. **261**, 107688 (2021).
[29]S. Martí, "QMCube (QM^3): An all-purpose suite for multiscale QM/MM calculations," J. Comput. Chem. **42**, 447–457 (2021).
[30]H. Lin, Y. Zhang, S. Pezeshki, A. W. Duster, B. Wang, X.-P. Wu, S.-W. Zheng, L. Gagliardi, and D. G. Truhlar, "QMMM 2023: A program for combined quantum mechanical and molecular mechanical modeling and simulations," Comput. Phys. Commun. **295**, 108987 (2024).
[31]C. M. Isborn, A. W. Götz, M. A. Clark, R. C. Walker, and T. J. Martínez, "Electronic Absorption Spectra from MM and ab Initio QM/MM Molecular Dynamics: Environmental Effects on the Absorption Spectrum of Photoactive Yellow Protein," J. Chem. Theory Comput. **8**, 5092–5106 (2012).
[32]V. W. D. Cruzeiro, M. Manathunga, K. M. J. Merz, and A. W. Götz, "Open-Source Multi-GPU-Accelerated QM/MM Simulations with AMBER and QUICK," J. Chem. Inf. Model. **61**, 2109–2115 (2021).
[33]J. M. H. Olsen, V. Bolnykh, S. Meloni, E. Ippoliti, M. P. Bircher, P. Carloni, and U. Rothlisberger, "MiMiC: A Novel Framework for Multiscale Modeling in Computational Chemistry," J. Chem. Theory Comput. **15**, 3810–3823 (2019).
[34]V. Bolnykh, J. M. H. Olsen, S. Meloni, M. P. Bircher, E. Ippoliti, P. Carloni, and U. Rothlisberger, "Extreme Scalability of DFT-Based QM/MM MD Simulations Using MiMiC," J. Chem. Theory Comput. **15**, 5601–5613 (2019).
[35]B. Raghavan, F. K. Schackert, A. Levy, S. K. Johnson, E. Ippoliti, D. Mandelli, J. M. H. Olsen, U. Rothlisberger, and P. Carloni, "MiMiCPy: An Efficient Toolkit for MiMiC-Based QM/MM Simulations," J. Chem. Inf. Model. **63**, 1406–1412 (2023).
[36]W. Humphrey, A. Dalke, and K. Schulten, "VMD: Visual molecular dynamics," J. Mol. Graphics **14**, 33–38 (1996).
[37]Schrödinger, LLC, "The PyMOL Molecular Graphics System, Version 1.8," (2015).
[38]"MiMiC Project on GitLab," `https://gitlab.com/mimic-project` (2024), date accessed: 2024-03-27.
[39]J. M. H. Olsen, V. Bolnykh, S. Meloni, E. Ippoliti, P. Carloni, and U. Röthlisberger, "MiMiC: A Framework for Multiscale Modeling in Computational Chemistry," Zenodo (2022), 10.5281/zenodo.5024022.
[40]V. Bolnykh, J. M. H. Olsen, S. Meloni, E. Ippoliti, N. Malapally, U. Röthlisberger, and P. Carloni, "MiMiC Communication Library," Zenodo (2023), 10.5281/zenodo.5035084.
[41]T. Preston-Werner, "Semantic Versioning 2.0.0," `https://semver.org/` (2013).
[42]A. Laio, J. VandeVondele, and U. Rothlisberger, "A Hamiltonian electrostatic coupling scheme for hybrid Car–Parrinello molecular dynamics simulations," J. Chem. Phys. **116**, 6941–6947 (2002).
[43]M. Bondanza, M. Nottoli, L. Cupellini, F. Lipparini, and B. Mennucci, "Polarizable embedding QM/MM: The future gold standard for complex (bio)systems?" Phys. Chem. Chem. Phys. **22**, 14433–14448 (2020).
[44]J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. Distasio, M. Head-Gordon, G. N. I. Clark, M. E. Johnson, and T. Head-Gordon, "Current Status of the AMOEBA Polarizable Force Field," J. Phys. Chem. B **114**, 2549–2564 (2010).
[45]D. Loco, L. Lagardère, S. Caprasecca, F. Lipparini, B. Mennucci, and J.-P. Piquemal, "Hybrid QM/MM Molecular Dynamics with AMOEBA Polarizable Embedding," J. Chem. Theory Comput. **13**, 4025–4033 (2017).
[46]M. Nottoli and F. Lipparini, "General formulation of polarizable embedding models and of their coupling," J. Chem. Phys. **153**, 224108 (2020).
[47]B. T. Thole, "Molecular polarizabilities calculated with a modified dipole interaction," Chem. Phys. **59**, 341–350 (1981).
[48]M. Tuckerman, B. J. Berne, and G. J. Martyna, "Reversible multiple time scale molecular dynamics," J. Chem. Phys. **97**, 1990–2001 (1992).
[49]R. P. Steele, "Multiple-timestep ab initio molecular dynamics with electron correlation," J. Chem. Phys. **139**, 011102 (2013).
[50]E. Liberatore, R. Meli, and U. Rothlisberger, "A Versatile Multiple Time Step Scheme for Efficient ab Initio Molecular Dynamics Simulations," J. Chem. Theory Comput. **14**, 2834–2842 (2018).
[51]F. Mouvet, J. Villard, V. Bolnykh, and U. Rothlisberger, "Recent Advances in First-Principles Based Molecular Dynamics," Acc. Chem. Res. **55**, 221–230 (2022).
[52]"CECAM Flagship School: Multiscale Molecular Dynamics with MiMiC," `https://www.cecam.org/workshop-details/1119` (2022), date accessed: 2024-03-27.
[53]M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," SoftwareX **1-2**, 19–25 (2015).
[54]S. Páll, M. J. Abraham, C. Kutzner, B. Hess, and E. Lindahl, "Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS," in *Solving Software Challenges for Exascale* (Springer International Publishing, 2015) pp. 3–27.
[55]"CPMD, copyright 1990-2023 by IBM Corp. and copyright 1994-2001 by Max Planck Institute, Stuttgart." `http://www.cpmd.org/` (2023), date accessed: 2024-03-27.
[56]M. Bonomi, D. Branduardi, G. Bussi, C. Camilloni, D. Provasi, P. Raiteri, D. Donadio, F. Marinelli, F. Pietrucci, R. A. Broglia, and M. Parrinello, "PLUMED: A portable plugin for free-energy calculations with molecular dynamics," Comput. Phys. Commun. **180**, 1961–1972 (2009).
[57]G. A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, and G. Bussi, "PLUMED 2: New feathers for an old bird," Comput. Phys. Commun. **185**, 604–613 (2014).
[58]D. A. Colón-Ramos, P. La Riviere, H. Shroff, and R. Oldenbourg, "Promoting transparency and reproducibility in enhanced molecular simulations," Nat. Methods **16**, 670–673 (2019).
[59]M. P. Bircher and U. Rothlisberger, "Plane-Wave Implementation and Performance of à-la-Carte Coulomb-Attenuated Exchange-Correlation Functionals for Predicting Optical Excitation Energies in Some Notorious Cases," J. Chem. Theory Comput. **14**, 3184–3195 (2018).
[60]M. P. Bircher, P. López-Tarifa, and U. Rothlisberger, "Shedding Light on the Basis Set Dependence of the Minnesota Functionals: Differences Between Plane Waves, Slater Functions, and Gaussians," J. Chem. Theory Comput. **15**, 557–571 (2018).
[61]J. Villard, M. P. Bircher, and U. Rothlisberger, "Structure and dynamics of liquid water from ab initio simulations: adding Minnesota density functionals to Jacob's ladder," Chem. Sci. (2024), 10.1039/d3sc05828j.
[62]V. Weber, C. Bekas, T. Laino, A. Curioni, A. Bertsch, and S. Futral, "Shedding Light on Lithium/Air Batteries Using Millions of Threads on the BG/Q Supercomputer," in *28th International Parallel and Distributed Processing Symposium* (IEEE, 2014).
[63]M. P. Bircher and U. Rothlisberger, "Exploiting Coordinate Scaling Relations To Accelerate Exact Exchange Calculations," J. Phys. Chem. Lett. **9**, 3886–3890 (2018).
[64]M. P. Bircher, J. Villard, and U. Rothlisberger, "Efficient Treatment of Correlation Energies at the Basis-Set Limit by Monte Carlo Summation of Continuum States," J. Chem. Theory Comput. **16**, 6550–6559 (2020).
[65]A. Putrino, D. Sebastiani, and M. Parrinello, "Generalized variational density functional perturbation theory," J. Chem. Phys. **113**, 7102–7109 (2000).
[66]J. Hutter, "Excited state nuclear forces from the Tamm–Dancoff approximation to time-dependent density functional theory within the plane wave basis set framework," J. Chem. Phys. **118**, 3928–3934 (2003).
[67]I. Frank, J. Hutter, D. Marx, and M. Parrinello, "Molecular dynamics in low-spin excited states," J. Chem. Phys. **108**, 4060–4069 (1998).
[68]I. Tavernelli, U. F. Röhrig, and U. Rothlisberger, "Molecular dynamics in electronically excited states using time-dependent density functional theory," Mol. Phys. **103**, 963–981 (2005).
[69]E. Tapavicza, I. Tavernelli, and U. Rothlisberger, "Trajectory surface hopping within linear response time-dependent density-functional theory," Phys. Rev. Lett. **98**, 023001 (2007).
[70]I. Tavernelli, B. F. Curchod, and U. Rothlisberger, "Mixed quantum-classical dynamics with time-dependent external fields: A time-dependent density-functional-theory approach," Phys. Rev. A **81**, 052508 (2010).
[71]B. F. Curchod, T. J. Penfold, U. Rothlisberger, and I. Tavernelli, "Local control theory in trajectory-based nonadiabatic dynamics," Phys. Rev. A **84**, 042507 (2011).
[72]D. Marx and M. Parrinello, "Ab initio path integral molecular dynamics: Basic ideas," J. Chem. Phys. **104**, 4077–4082 (1996).
[73]H. J. Berendsen, J. v. Postma, W. F. Van Gunsteren, A. DiNola, and J. R. Haak, "Molecular dynamics with coupling to an external bath," J. Chem. Phys. **81**, 3684–3690 (1984).

[74]S. Nosé, "A unified formulation of the constant temperature molecular dynamics methods," J. Chem. Phys. **81**, 511–519 (1984).
[75]S. Nosé, "A molecular dynamics method for simulations in the canonical ensemble," Mol. Phys. **52**, 255–268 (1984).
[76]W. G. Hoover, "Canonical dynamics: Equilibrium phase-space distributions," Phys. Rev. A **31**, 1695 (1985).
[77]M. Ceriotti, G. Bussi, and M. Parrinello, "Colored-noise thermostats à la carte," J. Chem. Theory Comput. **6**, 1170–1180 (2010).
[78]G. J. Martyna, M. E. Tuckerman, D. J. Tobias, and M. L. Klein, "Explicit reversible integrators for extended systems dynamics," Mol. Phys. **87**, 1117–1157 (1996).
[79]S. Páll, A. Zhmurov, P. Bauer, M. Abraham, M. Lundborg, A. Gray, B. Hess, and E. Lindahl, "Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS," J. Chem. Phys. **153**, 134110 (2020).
[80]J. W. Ponder and D. A. Case, "Force fields for protein simulations," Adv. Protein Chem. **66**, 27–85 (2003).
[81]B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus, "CHARMM: The biomolecular simulation program," J. Comput. Chem. **30**, 1545–1614 (2009).
[82]W. R. Scott, P. H. Hünenberger, I. G. Tironi, A. E. Mark, S. R. Billeter, J. Fennen, A. E. Torda, T. Huber, P. Krüger, and W. F. Van Gunsteren, "The GROMOS Biomolecular Simulation Program Package," J. Phys. Chem. A **103**, 3596–3607 (1999).
[83]M. J. Robertson, J. Tirado-Rives, and W. L. Jorgensen, "Improved Peptide and Protein Torsional Energetics with the OPLS-AA Force Field," J. Chem. Theory Comput. **11**, 3499–3509 (2015).
[84]B. Raghavan, M. Paulikat, K. Ahmad, L. Callea, A. Rizzi, E. Ippoliti, D. Mandelli, L. Bonati, M. De Vivo, and P. Carloni, "Drug Design in the Exascale Era: A Perspective from Massively Parallel QM/MM Simulations," J. Chem. Inf. Model. **63**, 3647–3658 (2023).
[85]D. Alvarez, "JUWELS Cluster and Booster: Exascale Pathfinder with Modular Supercomputing Architecture at Juelich Supercomputing Centre," JLSRF **7**, A183 (2021).
[86]A. Laio, J. VandeVondele, and U. Rothlisberger, "D-RESP: Dynamically Generated Electrostatic Potential Derived Charges from Quantum Mechanics/Molecular Mechanics Simulations," J. Phys. Chem. B **106**, 7300–7307 (2002).
[87]A. Laio, F. L. Gervasio, J. VandeVondele, M. Sulpizi, and U. Rothlisberger, "A Variational Definition of Electrostatic Potential Derived Charges," J. Phys. Chem. B **108**, 7963–7968 (2004).
[88]M. Sulpizi, U. Rothlisberger, and A. Laio, "Electron transfer induced dissociation of chloro-cyano-benzene radical anion: Driving chemical reactions via charge restraints," J. Theor. Comput. Chem. **4**, 985–999 (2005).
[89]P. Maurer, A. Laio, H. W. Hugosson, M. C. Colombo, and U. Rothlisberger, "Automated Parametrization of Biomolecular Force Fields from Quantum Mechanics/Molecular Mechanics (QM/MM) Simulations through Force Matching," J. Chem. Theory Comput. **3**, 628–639 (2007).
[90]M. Doemer, P. Maurer, P. Campomanes, I. Tavernelli, and U. Rothlisberger, "Generalized QM/MM Force Matching Approach Applied to the 11-cis Protonated Schiff Base Chromophore of Rhodopsin," J. Chem. Theory Comput. **10**, 412–422 (2014).
[91]T. D. Kühne, M. Iannuzzi, M. Del Ben, V. V. Rybkin, P. Seewald, F. Stein, T. Laino, R. Z. Khaliullin, O. Schütt, F. Schiffmann, D. Golze, J. Wilhelm, S. Chulkov, M. H. Bani-Hashemian, V. Weber, U. Borštnik, M. Taillefumier, A. S. Jakobovits, A. Lazzaro, H. Pabst, T. Müller, R. Schade, M. Guidon, S. Andermatt, N. Holmberg, G. K. Schenter, A. Hehn, A. Bussy, F. Belleflamme, G. Tabacchi, A. Glöß, M. Lass, I. Bethune, C. J. Mundy, C. Plessl, M. Watkins, J. VandeVondele, M. Krack, and J. Hutter, "CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations," J. Chem. Phys. **152**, 194103 (2020).
[92]P. Motamarri, S. Das, S. Rudraraju, K. Ghosh, D. Davydov, and V. Gavini, "DFT-FE – A massively parallel adaptive finite-element code for large-scale density functional theory calculations," Comput. Phys. Commun. **246**, 106853 (2020).
[93]S. Das, P. Motamarri, V. Subramanian, D. M. Rogers, and V. Gavini, "DFT-FE 1.0: A massively parallel hybrid CPU-GPU density functional theory code using finite-element discretization," Comput. Phys. Commun. **280**, 108473 (2022).
[94]P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials," J. Phys. Condens. Matter **21**, 395502 (2009).
[95]I. Carnimeo, F. Affinito, S. Baroni, O. Baseggio, L. Bellentani, R. Bertossa, P. D. Delugas, F. F. Ruffino, S. Orlandini, F. Spiga, and P. Giannozzi, "Quantum ESPRESSO: One Further Step toward the Exascale," J. Chem. Theory Comput. **19**, 6992–7006 (2023).
[96]D. A. Matthews, L. Cheng, M. E. Harding, F. Lipparini, S. Stopkowicz, T.-C. Jagau, P. G. Szalay, J. Gauss, and J. F. Stanton, "Coupled-cluster techniques for computational chemistry: The CFOUR program package," J. Chem. Phys. **152**, 214108 (2020).
[97]J. F. Stanton, J. Gauss, L. Cheng, M. E. Harding, D. A. Matthews, and P. G. Szalay, "CFOUR, Coupled-Cluster techniques for Computational Chemistry, a quantum-chemical program package," With contributions from A. Asthana, A.A. Auer, R.J. Bartlett, U. Benedikt, C. Berger, D.E. Bernholdt, S. Blaschke, Y. J. Bomble, S. Burger, O. Christiansen, D. Datta, F. Engel, R. Faber, J. Greiner, M. Heckert, O. Heun, M. Hilgenberg, C. Huber, T.-C. Jagau, D. Jonsson, J. Jusélius, T. Kirsch, M.-P. Kitsaras, K. Klein, G.M. Kopper, W.J. Lauderdale, F. Lipparini, J. Liu, T. Metzroth, L.A. Mück, D.P. O'Neill, T. Nottoli, J. Oswald, D.R. Price, E. Prochnow, C. Puzzarini, K. Ruud, F. Schiffmann, W. Schwalbach, C. Simmons, S. Stopkowicz, A. Tajti, T. Uhlirova, J. Vázquez, F. Wang, J.D. Watts, P. Yergün. C. Zhang, X. Zheng, and the integral packages MOLECULE (J. Almlöf and P.R. Taylor), PROPS (P.R. Taylor), ABACUS (T. Helgaker, H.J. Aa. Jensen, P. Jørgensen, and J. Olsen), and ECP routines by A. V. Mitin and C. van Wüllen. For the current version, see `https://www.cfour.de`.
[98]L. Lagardère, L. H. Jolly, F. Lipparini, F. Aviat, B. Stamm, Z. F. Jing, M. Harger, H. Torabifard, G. A. Cisneros, M. J. Schnieders, N. Gresh, Y. Maday, P. Y. Ren, J. W. Ponder, and J. P. Piquemal, "Tinker-HP: A massively parallel molecular dynamics package for multiscale simulations of large complex systems with advanced point dipole polarizable force fields," Chem. Sci. **9**, 956–972 (2018).
[99]O. Adjoua, L. Lagardère, L.-H. Jolly, A. Durocher, T. Very, I. Dupays, Z. Wang, T. J. Inizan, F. Célerse, P. Ren, J. W. Ponder, and J.-P. Piquemal, "Tinker-HP: Accelerating Molecular Dynamics Simulations of Large Complex Systems with Advanced Point Dipole Polarizable Force Fields Using GPUs and Multi-GPU Systems," J. Chem. Theory Comput. **17**, 2034–2053 (2021).
[100]P. Eastman, M. S. Friedrichs, J. D. Chodera, R. J. Radmer, C. M. Bruns, J. P. Ku, K. A. Beauchamp, T. J. Lane, L.-P. Wang, D. Shukla, T. Tye, M. Houston, T. Stich, C. Klein, M. R. Shirts, and V. S. Pande, "OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation," J. Chem. Theory Comput. **9**, 461–469 (2012).
[101]P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, and V. S. Pande, "OpenMM 7: Rapid development of high performance algorithms for molecular dynamics," PLoS Comput. Biol. **13**, e1005659 (2017).
[102]P. Eastman, R. Galvelis, R. P. Peláez, C. R. A. Abreu, S. E. Farr, E. Gallicchio, A. Gorenko, M. M. Henry, F. Hu, J. Huang, A. Krämer, J. Michel, J. A. Mitchell, V. S. Pande, J. P. Rodrigues, J. Rodriguez-Guerra, A. C. Simmonett, S. Singh, J. Swails, P. Turner, Y. Wang, I. Zhang, J. D. Chodera, G. De Fabritiis, and T. E. Markland, "Openmm 8: Molecular dynamics simulation with machine learning potentials," J. Phys. Chem. B **128**, 109–116 (2023).
[103]M. Krack and M. Parrinello, "Quickstep: Make the Atoms Dance," in *High performance computing in chemistry*, Vol. 25, edited by J. Grotendorst

(NIC-Directors, 2004) pp. 29–51.
[104]J. VandeVondele, M. Krack, F. Mohamed, M. Parrinello, T. Chassaing, and J. Hutter, "Quickstep: Fast and accurate density functional calculations using a mixed Gaussian and plane waves approach," Comput. Phys. Commun. **167**, 103–128 (2005).
[105]R. Schade, T. Kenter, H. Elgabarty, M. Lass, T. D. Kühne, and C. Plessl, "Breaking the exascale barrier for the electronic structure problem in ab-initio molecular dynamics," Int. J. High Perform. Comput. Appl. **37**, 530–538 (2023).
[106]S. Das, B. Kanungo, V. Subramanian, G. Panigrahi, P. Motamarri, D. Rogers, P. Zimmerman, and V. Gavini, "Large-Scale Materials Modeling at Quantum Accuracy: Ab Initio Simulations of Quasicrystals and Interacting Extended Defects in Metallic Alloys," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '23 (ACM, 2023) pp. 1–12.
[107]P. Giannozzi, O. Baseggio, P. Bonfà, D. Brunato, R. Car, I. Carnimeo, C. Cavazzoni, S. de Gironcoli, P. Delugas, F. Ferrari Ruffino, A. Ferretti, N. Marzari, I. Timrov, A. Urru, and S. Baroni, "Quantum ESPRESSO toward the exascale," J. Chem. Phys. **152**, 154105 (2020).
[108]T. Kirsch, J. M. H. Olsen, V. Bolnykh, S. Meloni, E. Ippoliti, U. Rothlisberger, M. Cascella, and J. Gauss, "Wavefunction-Based Electrostatic-Embedding QM/MM Using CFOUR through MiMiC," J. Chem. Theory Comput. **18**, 13–24 (2022).
[109]G. Lamoureux and B. Roux, "Modeling induced polarization with classical Drude oscillators: Theory and molecular dynamics simulation algorithm," J. Chem. Phys. **119**, 3025–3039 (2003).
[110]M. G. Chiariello, V. Bolnykh, E. Ippoliti, S. Meloni, J. M. H. Olsen, T. Beck, U. Rothlisberger, C. Fahlke, and P. Carloni, "Molecular Basis of CLC Antiporter Inhibition by Fluoride," J. Am. Chem. Soc. **142**, 7254–7258 (2020).
[111]The PLUMED consortium, "Promoting transparency and reproducibility in enhanced molecular simulations," Nat. Methods **16**, 670–673 (2019).
[112]M. G. Chiariello, M. Alfonso-Prieto, E. Ippoliti, C. Fahlke, and P. Carloni, "Mechanisms Underlying Proton Release in CLC-type F^-/H^+ Antiporters," J. Phys. Chem. Lett. **12**, 4415–4420 (2021).
[113]S. Asgharpour, L. A. Chi, M. Spehr, P. Carloni, and M. Alfonso-Prieto, "Fluoride Transport and Inhibition Across CLC Transporters," in *Handb. Exp. Pharmacol.* (Springer International Publishing, 2022) pp. 81–100.
[114]N. Dmitrieva, C. Alleva, M. Alfonso Prieto, P. Carloni, and C. M. Fahlke, "Exploring the transport cycle of DgoT, a bacterial homolog of human vesicular glutamate transporters," Biophys. J. **122** (2023), 10.1016/j.bpj.2022.11.1363.
[115]N. Dmitrieva, S. Gholami, C. Alleva, P. Carloni, M. Alfonso-Prieto, and C. Fahlke, "Transport mechanism of DgoT, a bacterial homolog of SLC17 organic anion transporters," bioRxiv (2024), 10.1101/2024.02.07.579339.
[116]F. K. Schackert, J. Biedermann, S. Abdolvand, S. Minniberger, C. Song, A. J. R. Plested, P. Carloni, and H. Sun, "Mechanism of Calcium Permeation in a Glutamate Receptor Ion Channel," J. Chem. Inf. Model. **63**, 1293–1300 (2023).
[117]R. Capelli, W. Lyu, V. Bolnykh, S. Meloni, J. M. H. Olsen, U. Rothlisberger, M. Parrinello, and P. Carloni, "Accuracy of Molecular Simulation-Based Predictions of k_{off} Values: A Metadynamics Study," J. Phys. Chem. Lett. **11**, 6373–6381 (2020).
[118]K. Ahmad, A. Rizzi, R. Capelli, D. Mandelli, W. Lyu, and P. Carloni, "Enhanced-Sampling Simulations for the Estimation of Ligand Binding Kinetics: Current Status and Perspective," Front. Mol. Biosci. **9**, 899805 (2022).
[119]X. Xu, J. Zhao, Z. Xu, B. Peng, Q. Huang, E. Arnold, and J. Ding, "Structures of Human Cytosolic NADP-dependent Isocitrate Dehydrogenase Reveal a Novel Self-regulatory Mechanism of Activity," J. Biol. Chem. **279**, 33946–33957 (2004).
[120]A. Remmel, "What exascale computing could mean for chemistry," C&EN Global Enterp. **100** (2022).
[121]V. Bolnykh, G. Rossetti, U. Rothlisberger, and P. Carloni, "Expanding the boundaries of ligand–target modeling by exascale calculations," Wiley Interdiscip. Rev. Comput. Mol. Sci. **11**, e1535 (2021).
[122]W. Martin, G. Sheynkman, F. C. Lightstone, R. Nussinov, and F. Cheng, "Interpretable artificial intelligence and exascale molecular dynamics simulations to reveal kinetics: Applications to Alzheimer's disease," Curr. Opin. Struct. Biol. **72**, 103–113 (2022).
[123]T. L. Beck, P. Carloni, and D. N. Asthagiri, "All-Atom Biomolecular Simulation in the Exascale Era," J. Chem. Theory Comput. **20**, 1777–1782 (2024).
[124]P. Carpenter, U.-H. Utz, S. Narasimhamurthy, and E. Suarez, "Heterogeneous High Performance Computing," Zenodo (2022), 10.5281/zenodo.6090425.
[125]E. Suarez, N. Eicker, T. Moschny, S. Pickartz, C. Clauss, V. Plugaru, A. Herten, K. Michielsen, and T. Lippert, "Modular Supercomputing Architecture," Zenodo (2022), 10.5281/zenodo.6508394.
[126]D. Carrasco-Busturia, E. Ippoliti, S. Meloni, U. Rothlisberger, and J. M. H. Olsen, "Multiscale Biomolecular Simulations in the Exascale Era," arXiv preprint arXiv:2403.01511 (2024), 10.48550/arXiv.2403.01511.
[127]"JUPITER Technical Overview," `https://www.fz-juelich.de/en/ias/jsc/jupiter/tech` (2024), date accessed: 2024-03-27.
[128]O. T. Unke, S. Chmiela, H. E. Sauceda, M. Gastegger, I. Poltavsky, K. T. Schütt, A. Tkatchenko, and K.-R. Müller, "Machine Learning Force Fields," Chem. Rev. **121**, 10142–10186 (2021).
[129]E. Kocer, T. W. Ko, and J. Behler, "Neural Network Potentials: A Concise Overview of Methods," Annu. Rev. Phys. Chem. **73**, 163–186 (2022).
[130]J. Mato, A. W. Duster, E. B. Guidez, and H. Lin, "Adaptive-Partitioning Multilayer Dynamics Simulations: 1. On-the-Fly Switch between Two Quantum Levels of Theory," J. Chem. Theory Comput. **17**, 5456–5465 (2021).