QXMD: An open-source program for nonadiabatic quantum molecular dynamics

Fuyuki Shimojo, Shogo Fukushima, Hiroyuki Kumazoe, Masaaki Misawa, Satoshi Ohmura, Pankaj Rajak, Kohei Shimamura, Lindsay Bassman, Subodh Tiwari, Rajiv K. Kalia, Aiichiro Nakano, Priya Vashishta

Department of Physics, Kumamoto University, Kumamoto 860-8555, Japan
Faculty of Science and Engineering, Kyushu Sangyo University, Fukuoka 813-8503, Japan
Research Center for Condensed Matter Physics, Hiroshima Institute of Technology, Hiroshima 731-5193, Japan
Leadership Computing Facility, Argonne National Laboratory, Argonne, IL 60439, United States
Graduate School of System Informatics, Kobe University, Kobe 657-8501, Japan
Collaboratory for Advanced Computing and Simulations, University of Southern California, Los Angeles, CA 90089-0242, United States

**Abstract**

QXMD is a scalable, parallel program for Quantum Molecular Dynamics simulations with various extensions. Its simulation engine is based on (time-dependent) density functional theory using pseudopotentials and a plane-wave basis set, while extensions include nonadiabatic electron–nuclei dynamics and multiscale shock technique. QXMD serves as a community-development platform for new methods and algorithms, a research platform on high-end parallel supercomputers, and an educational platform for hands-on training.

© 2019 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Motivation and significance

Quantum molecular dynamics (QMD) is a widely used simulation method to study the dynamic behavior of materials [1–3]. QMD follows the trajectories of each atom in a system, while computing interatomic forces quantum mechanically within the framework of density functional theory (DFT) [4,5]. In particular, nonadiabatic QMD (NAQMD) describes electronic excitations and transitions between excited electronic states assisted by atomic motions, thereby describing excitation dynamics involving electrons and nuclei [6–8].

The ever-increasing computing power of high-end parallel supercomputers [9] is enabling the study of complex material processes encompassing unprecedentedly large spatiotemporal scales, while incorporating higher levels of physical and chemical fidelity [10]. However, it also poses enormous algorithmic and computational challenges for scaling QMD simulations up...
to emerging computing architectures [10]. QXMD is a software platform for developing new eXtensions and eXperimental features to make QMD simulations metascalable, i.e., “design once, scale on new architectures” [11,12]. The core QXMD simulation engine is based on DFT using pseudopotentials and a plane-wave basis set [2,13]. Extensions implemented thus far include (1) linear-response time-dependent density functional theory (LR-TDDFT) [12] with range-separated exact-exchange correction to describe excitonic binding [13], (2) excited-state forces to describe photo-excited energy landscapes [7], (3) multiscale shock technique (MSST) to describe shock-front dynamics [16], (4) Berry-phase computation of electric polarizability [17,18], and (5) divide-and-conquer DFT (DC-DFT) algorithm [19,20] for O(N) QMD simulations [21], in which the computation scales linearly with the number of electrons, \( N \). With these extensions, QXMD has also been used to study a wide range of quantum-dynamical processes in various materials and energy applications on massively parallel supercomputers [22–28].

In addition to serving as both community-development and high-end computational research platforms as described above, elementary features of the QXMD software have been used in our MAGICS (MAterials Genome Innovation for Computational Software) workshops to train researchers on the basics of QMD and NAQMD simulations. In this paper, we outline the architecture of the QXMD software. An earlier version of the software is described in Ref. [29], while subsequent extensions are detailed in Ref. [11]. The present paper focuses on the principal features of the QXMD software and corresponding tutorial material, which can be used in various workshop and classroom settings. Related basics of pseudopotential and plane-wave implementations of DFT are found, e.g., in Ref. [13].

2. Software description

QXMD is a highly scalable, parallel program written in Fortran for performing QMD simulations. The program is parallelized based on hybrid space-band decomposition [11] and uses the message passing interface (MPI) library [30] to perform internode communication. QXMD also requires a fast Fourier transform (FFT) library, such as FFTW3 [31]. Strong-scaling tests show that time-to-solution is reduced by a factor of 2.5 on 64 processor cores compared to 16 cores for a 1,284-atom system. This represents decent scaling for typical QMD applications. While the current QXMD software distribution is for tutorial purposes only, elementary features of the QXMD software have been used in our MAGICS (MAterials Genome Innovation for Computational Software) workshops to train researchers on the basics of QMD and NAQMD simulations. In this paper, we outline the architecture of the QXMD software. An earlier version of the software is described in Ref. [29], while subsequent extensions are detailed in Ref. [11]. The present paper focuses on the principal features of the QXMD software and corresponding tutorial material, which can be used in various workshop and classroom settings. Related basics of pseudopotential and plane-wave implementations of DFT are found, e.g., in Ref. [13].

2.2. Configuration and compilation

QXMD is configured and compiled using the makefile provided with the distribution. The makefile includes preconfigured setups for various supercomputing architectures. A complete list of supported machines can be obtained by running the command ‘make help’. QXMD is configured by running ‘make [machine_name]’, where the user should specify the name of the particular machine on which QXMD is to be run. After successful completion of the configuration, QXMD is compiled by running ‘make qxmd’ for a serial executable, or ‘make qxmdmpi’ for a parallel executable. Note that a parallelized make is enabled by default.

2.3. Software functionalities

QXMD performs QMD simulations under three main paradigms: (i) adiabatic QMD, where the electrons stay in their ground state; (ii) NAQMD based on TD-DFT, in which electrons can nonradiatively hop between energy levels; and (iii) MSST to simulate shock-front dynamics. QXMD also supports in-situ analyses, such as stress tensor and atomic charge calculations, as well as multiple data dumps such as wave functions, charge density and local potential. To minimize I/O, each data dump is turned off by default as these files are very large, though they can optionally be performed every nth ionic step based on the user’s choice.
Fig. 1. Tree structure of the directories and files necessary to run a QXMD simulation.

3. Illustrative examples

3.1. Running a generic simulation

In order to run QXMD, there are a few mandatory directories and files that must be present in the correct hierarchy (Fig. 1). The working directory, from which a QXMD job is launched, must contain a control/ and data/ directory. The control/ directory contains setup files/directories as described below:

- **CONFIG**: configuration file detailing the atomic coordinates of the system in real or normalized coordinates.
- **filename**: simple text file containing the path to the main input file.
- **IN.PARAM**: main input file containing various input settings and parameters for the simulation.
- **VELOC**: optional file containing the initial velocities for each atom in the system.
- **NCPP/**: directory containing pseudopotential files for atomic species.

The data/ directory stores all output files generated during the simulation. Fig. 1 shows an example directory tree for successfully running a simulation of a water molecule.

There are sample input files for various types of simulations in the Examples/ directory, including optimization of water, adiabatic QMD of a water molecule, NAQMD of monolayer MoSe₂, and MSST simulation of SiO₂. The NAQMD and MSST examples are explained in more detail in the sections that follow.

3.2. NAQMD simulation of monolayer MoSe₂

In this example, we perform an NAQMD simulation of monolayer MoSe₂ [26,37]. In QXMD, this is carried out by promoting a set of electrons from their ground states to excited states, which mimics the dynamics in a system after photo-excitation by a laser. A sample input file can be found in Examples/03_NAQMD/01_MoSe2/control. The key section in the input parameter file, IN.PARAM, for running an NAQMD simulation is the **TDDFT-MD** section. In this section, the "on/off" variable must first be set to true. The changes in electronic occupation numbers for the relevant energy bands must also be defined with the "occupations" variables. In this example, we promote all four electrons in the two highest occupied orbitals (energy bands 35 and 36) to the two lowest lying unoccupied orbitals (energy bands 37 and 38). Note that if spin polarization is not used, both electrons in one orbital must be moved to a second orbital, otherwise one would need to define whether a single electron being moved is spin-up or spin-down.

Once the simulation completes, there are several utility files available for post-processing of the output data. The main function of the utility files is to transform the raw data output by QXMD into a format that is easily read by existing graphing and visualization software. Here, we explain how to use two of these utility files: `eig_exocc.f` for creating a plot of the Kohn–Sham eigenenergies and their occupation numbers versus time, and `gcube.f90` for producing images of the charge densities of various energy bands. `eig_exocc.f` can be found in Examples/03_NAQMD/01_MoSe2/analysis/eig. The utility program takes the path of the output data directory as a command line argument. After compiling the utility program, the resultant executable can be run with `./eig_exocc -d ../data`. This will produce three files which contain information on the Kohn–Sham energy eigenvalues and their occupation numbers. They can be used to compile all the information into one graph by running the gnuplot script `plot_eig.gnu`, as shown in Fig. 2(a).

`gcube.f90` can be found in Examples/03_NAQMD/01_MoSe2/analysis/Gcube. This utility program takes multiple command-line arguments, including the path to the output data, as well as options to choose the frequency of times steps and the range of Kohn–Sham eigenstates for which to create Gaussian cube files. After compilation, the executable can be run with `./gcube -d ../data -n 101 -b 36 -eb 37`, which will create Gaussian cube files for every 101st time step for bands 36 through 37. The resultant data files can be used as input for many visualization software packages (e.g. VMD) to create an image, shown in Fig. 2(b).

3.3. MSST simulation of SiO₂

In this example, we use QXMD to study shock on SiO₂(α-quartz) using MSST [16]. In MSST, shock is assumed to be planar, and usually a rectangular parallelepiped computation cell is assumed. Here, we employ an omni-directional variant of MSST developed by Shimamura et al. [38,39], which allows us to study shock in any crystallographic direction.

A sample input file can be found in Examples/04_MSST/01_SiO2/control. To enable MSST, the method of dynamics in the **molecular dynamics** section must be changed to option ‘10’. Shock speed and direction must be provided by setting the “shock wave velocity” input parameter, also in the **molecular dynamics** section. In this example, we have applied shock with a velocity of 7.2 km/s in the [210] crystallographic direction by setting `shock_speed` to 7,200 m/s and `nschockv` to (2,1,0). Particle velocity, pressure and temperature obtained from the Lagrangian after each iteration is written in the 'md_hug.d' file found in the output data directory. Fig. 3, (a) and (b), shows the initial and final configurations after applying shock.

The utility program 'toPDBcell.f' in Examples/04_MSST/01_SiO2/analysis/PDB can be used to create a PDB (protein data bank) formatted file of the atomic trajectories. The program takes the path of the output data directory as a command-line argument. Output PDB files can be visualized in many existing software packages such as VMD or OVITO. After compiling the utility program, the resultant executable can be run with `./toPDBcell -d ../data`. This will produce a PDB file named ‘config.pdb’, which contains the atomic trajectories of all atoms. This data can be used to create Fig. 3.

4. Impact

Three major impacts of the QXMD software are to: (1) provide an open-source, community code-development platform for advanced QMD and NAQMD simulation methods and algorithms; (2) serve as a research platform on massively parallel supercomputers; and (3) disseminate the extensive hands-on tutorial materials to train a broad scientific community.

With often-disruptive advancements in computer architectures [9], it is necessary to continuously develop new algorithms...
and computational techniques for QMD simulations that better adapt to the new architectures [10]. One example is algebraic transformation of nonlocal pseudopotential computations to make them amenable to computationally-efficient level-3 basic linear-algebra software (BLAS3), i.e., via BLASification, thereby achieving high floating-point performance [12]. QXMD continues to serve as a platform for such new technological developments.

QXMD has also been used to study a wide range of quantum-dynamical processes on high-end supercomputers, thereby providing new scientific knowledge. Examples include (1) emergence of thermodynamically-stable interphases in ceramic interfaces [22], (2) on-demand hydrogen production from water using metallic nanoparticles [24], (3) emergent magnetism from solely non-magnetic elements in two-dimensional layered materials [27], (4) dynamics of strongly-bound [23] and free [25] electron–hole pairs, and (5) ultrafast optical control of lattice motions [26,28], just to name a few.

Finally, as mentioned in Section 1, QXMD has been used extensively as an educational platform. We have developed hands-on tutorials as part of QXMD, and as of November 2018, 108 users from 55 institutions have been trained using QXMD in the MAGICS software workshops.

5. Conclusions

To date, QXMD has been used for method and algorithmic developments, basic scientific research, and training on first-principles materials simulations, on machines of all sizes, from desktop to high-end parallel computing platforms. Importantly, QXMD also serves as a testbed for hardware–software co-development on emerging computing architectures. For example, to facilitate easy access for development and collaboration, we have ported our code to Code Ocean. QXMD is also one of the ten initial simulation codes that will run on the United States’ first exaflop/s computer (which can perform $10^{18}$ floating-point operations per second) under the US Department of Energy, Aurora early science program [9]. On these platforms, we are developing new techniques to: (1) make QXMD scalable for the unprecedented level of concurrency using globally-scalable and locally-fast (GSLF) solvers [11,12], which combine scalable tree-based algorithms for internode computations and fast BLAS3- and FFT-based solvers for intranode computations; and (2) reduce the time-to-solution using mixed-precision tensor operations by BLASifying key computational kernels. These developments will be reported in future publications.

Declaration of competing interest

The authors confirm that there is no conflict of interest associated with this publication.

Acknowledgments

This work was supported as part of the Computational Materials Sciences Program funded by the U.S. Department of Energy.
References