JCTC Journal of Chemical Theory and Computation

Article

Cite This: J. Chem. Theory Comput. 2019, 15, 1317−1328

pubs.acs.org/JCTC

Open-Source, Python-Based Redevelopment of the ChemShell Multiscale QM/MM Environment

You Lu,[†] Matthew R. Farrow,[‡] Pierre Fayon,[†,⊥] Andrew J. Logsdail,[‡,§] Alexey A. Sokol,[‡] C. Richard A. Catlow,[‡,§,∥] Paul Sherwood,[†] and Thomas W. Keal*[,†]

[†]Scientific Computing Department, STFC Daresbury Laboratory, Keckwick Lane, Daresbury, Warrington WA4 4AD, United Kingdom

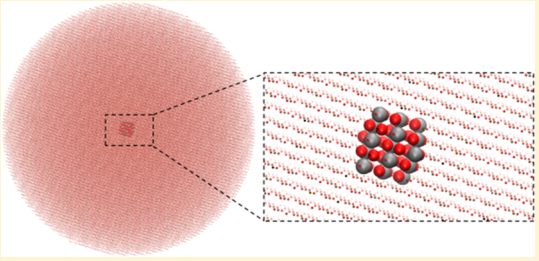
[‡]Kathleen Lonsdale Materials Chemistry, Department of Chemistry, University College London, 20 Gordon Street, London WC1H 0AJ, United Kingdom

[§]Cardiff Catalysis Institute, School of Chemistry, Cardiff University, Cardiff CF10 3AT, United Kingdom

[∥]UK Catalysis Hub, Research Complex at Harwell, STFC Rutherford Appleton Laboratory, Harwell Science and Innovation Campus, Oxon OX11 0QX, United Kingdom

Supporting Information

ABSTRACT: ChemShell is a scriptable computational chemistry environment with an emphasis on multiscale simulation of complex systems using combined quantum mechanical and molecular mechanical (QM/MM) methods. Motivated by a scientific need to efficiently and accurately model chemical reactions on surfaces and within microporous solids on massively parallel computing systems, we present a major redevelopment of the ChemShell code, which provides a modern platform for advanced QM/MM embedding models. The new version of ChemShell has been re-engineered from the ground up with a new QM/MM driver module, an improved parallelization framework, new interfaces to high performance QM and MM programs, and a user interface written in the Python programming language. The redeveloped package is capable of performing QM/MM calculations on systems of significantly increased size, which we illustrate with benchmarks on zirconium dioxide nanoparticles of over 160000 atoms.



1. INTRODUCTION

Combined quantum mechanical/molecular mechanical (QM/MM) calculations are an efficient means of calculating localized chemistry in complex molecular and solid systems. In the QM/MM approach, a region of interest in the system is identified where an electronic structure description is required, for example the site of a reaction or a defect. A quantum mechanical calculation is performed on this region, while the environment around it is described with a classical molecular mechanics force field. This approach ensures that local chemical processes are modeled with sufficient accuracy, while avoiding the expense of treating the whole system quantum mechanically.

ChemShell is a general-purpose scriptable computational chemistry environment[1,2] with an emphasis on QM/MM simulations. ChemShell implements a flexible, modular approach to QM/MM, where a variety of QM and MM programs (either built in as libraries or called through external interfaces) may be used to evaluate energies and gradients of the quantum and classical regions, while ChemShell takes on the task of coupling the results to obtain the combined QM/MM energy and gradient, including an appropriate treatment of the boundary region that couples the two subsystems. High-level tasks such as geometry optimization and molecular dynamics are also handled at the ChemShell level, ensuring that these are carried out in a consistent way regardless of the choice of energy evaluator.

Previously, ChemShell has been used to model a wide range of chemical systems using QM/MM methods, particularly in the fields of biomolecular and materials modeling. In biochemistry, QM/MM methods are well-suited to simulating enzymatic reaction mechanisms[3] and ChemShell has been used to study many important enzymatic processes, with recent investigations targeting oxidoreductases such as flavin-containing amine oxidases,[4] lignin peroxidase,[5] multicopper oxidases,[6] [Fe] hydrogenase,[7] Ni−Fe carbon monoxide dehydrogenases,[8] iron-containing dioxygenases,[9−13] and iron-containing halogenases;[14,15] transferases such as protein kinases[16] and glycosyltransferases;[17] hydrolases such as the lipase family[18] and matrix metalloproteinases;[19,20] lyases such as the radical SAM superfamily;[21] and ligases such as carbapenem synthase.[22] ChemShell continues to be particularly heavily used for studies of the cytochrome P450 superfamily of enzymes owing to their importance in metabolic processes,[23] with recent studies investigating reactivity networks,[24] active

Received: October 15, 2018
Published: December 4, 2018

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

site chemoselectivity,[25] regio- and enantioselectivity of fatty acid hydroxylation,[26,27] and redox-catalyzed drug metabolism mechanisms.[28] ChemShell has also been used to study prototypical copper complexes,[29] electron hole transport in proteins,[30] protein−protein interfaces,[31] covalent inhibitors for drug design,[32] DNA repair mechanisms,[33,34] and enzymatic reactions in crystals.[35,36]

As well as studying reaction mechanisms, ChemShell is used to determine QM/MM spectroscopic properties through use of appropriate QM programs, with recent work involving IR,[37] NMR,[38] and Mössbauer[15,39] calculations. The QM and QM/MM molecular dynamics driver can be used to study chemical reactivity and kinetics at arbitrary levels of theory,[40−42] and excited state optimization and molecular dynamics methods are also supported, which have recently been used to study chromophore emission properties in proteins,[43,44] light-oxygen-voltage domains in blue-light-sensitive proteins,[45] and OLEDs.[46]

In materials chemistry, QM/MM techniques are increasingly used to model defects, localized electronic states, sorbed species, and catalytic reactions on surfaces, where they have distinct advantages over the widely used periodic methods in their avoidance of artificial periodicity, capability to handle charged (surface) defects without *a posteriori* correction schemes, and the availability of a clearly defined vacuum level; they may also prove to be more economical in computational resources. The solid state QM/MM embedded cluster model implemented in ChemShell has recently been used to investigate defect formation in wide band gap semiconductors;[47] the band energies of TiO_2 polymorphs[48] and band alignment of mixed-phase TiO_2;[49] the energetics of Mg doping,[50] stabilization of silicon and oxygen dopants,[51] and multiband luminescence in GaN;[52] and the nature of oxygen vacancies in transparent conducting oxides.[53] The embedded cluster approach is also extensively used for the study of microporous catalysts[54,55] and has been adapted to describe one-dimensional CdS nanowires.[56] Recent surface studies have included the adsorption of CO_2 on MgO[57] and Mn-doped MgO[58] and its reactivity with H_2,[59] water oxidation on TiO_2,[60] and reactivity on ice surfaces using instanton theory.[61−63] ChemShell has also been used to study molecular crystals,[64] metal−organic frameworks (MOFs),[65] and the aggregate-induced emission and the influence of structural distortion on the optical structure of nanoparticles.[66,67] Further discussion of the methodology for solid state embedding is available in ref 68.

The original ChemShell program, with an interface written in the Tcl scripting language[69] (herein referred to as "Tcl-ChemShell"), is a well-established, mature software package, having been under active development for over 20 years. This article describes our work to redevelop ChemShell using the Python programming language[70] ("Py-ChemShell"). This work was motivated for several reasons. First, Python has become established in recent years as one of the most popular programming languages in the computational chemistry community, and the redevelopment provides a more familiar and appealing working environment for new users. Second, Python is a more powerful and flexible language than Tcl, with better support for mathematical operations, complex data structures, and a wider range of support libraries. Third, the redevelopment has offered the opportunity to rationalize the underlying codebase in order to make ongoing development of new embedding methods more straightforward. This includes the development of a significantly improved parallelization framework, with associated new functionality including a general purpose finite difference gradient module and new directly linked interfaces to QM and MM codes to enable highly scalable QM/MM materials simulations. Finally, we have taken the opportunity afforded by rewriting the software codebase to change to an open source licensing model. The new Py-ChemShell package is available for download under the GNU LGPL v3 free software license at www.chemshell.org.

2. PYTHON IMPLEMENTATION

In the original Tcl-ChemShell program, the Tcl scripting language is used mainly to provide the user interface layer, with complex data structures and manipulation implemented in the more flexible C programming language, and some modules and interfaces to external packages written in Fortran. In Py-ChemShell, we have exploited the power of the Python language and the associated mathematical library NumPy[71] to simplify this structure, with both the user interface and data structures handled at the Python level, while the data structures can be directly accessed by Fortran modules and interface code without the need for C wrappers, as described below.

2.1. User Interface. Py-ChemShell retains the flexible scripting approach taken by Tcl-ChemShell, and we intend that Py-ChemShell inputs will look familiar to experienced ChemShell users, albeit translated into a new programming language. As a simple example, we present a script that carries out a purely quantum mechanical optimization of a water molecule using the NWChem QM code with the BLYP density functional and a 6-31G basis set:

```
# Initialize ChemShell
from chemsh import *
# Load water geometry from XYZ file
my_mol = Fragment(coords='water.xyz')
# Define QM level of theory using NWChem
my_qm = NWChem(frag=my_mol,
method='dft', functional='blyp',
basis='6-31g')
# Run geometry optimization with DL-FIND
my_opt = Opt(theory=my_qm)
my_opt.run()
# Write optimized energy to output
print('Energy = ', my_opt.result.energy)
```

This script can be run either in an ordinary Python interpreter after importing the Py-ChemShell library, or through calling the user-compiled ChemShell executable as follows:

```
chemsh h2o.py
```

where `h2o.py` is the Python input script. The ChemShell executable is particularly intended for parallel calculations, as described in section 3.

2.2. Data Structures. The ChemShell object types previously implemented in C have been reimplemented as Python classes in accordance with the principles of object-oriented programming. An overview of these classes is provided in Figure 1. At the highest level are classes defining computational tasks, such as the `SP` class for single-point calculations, and the `Opt` class for geometry optimization. Different levels of theory are then defined, with classes

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

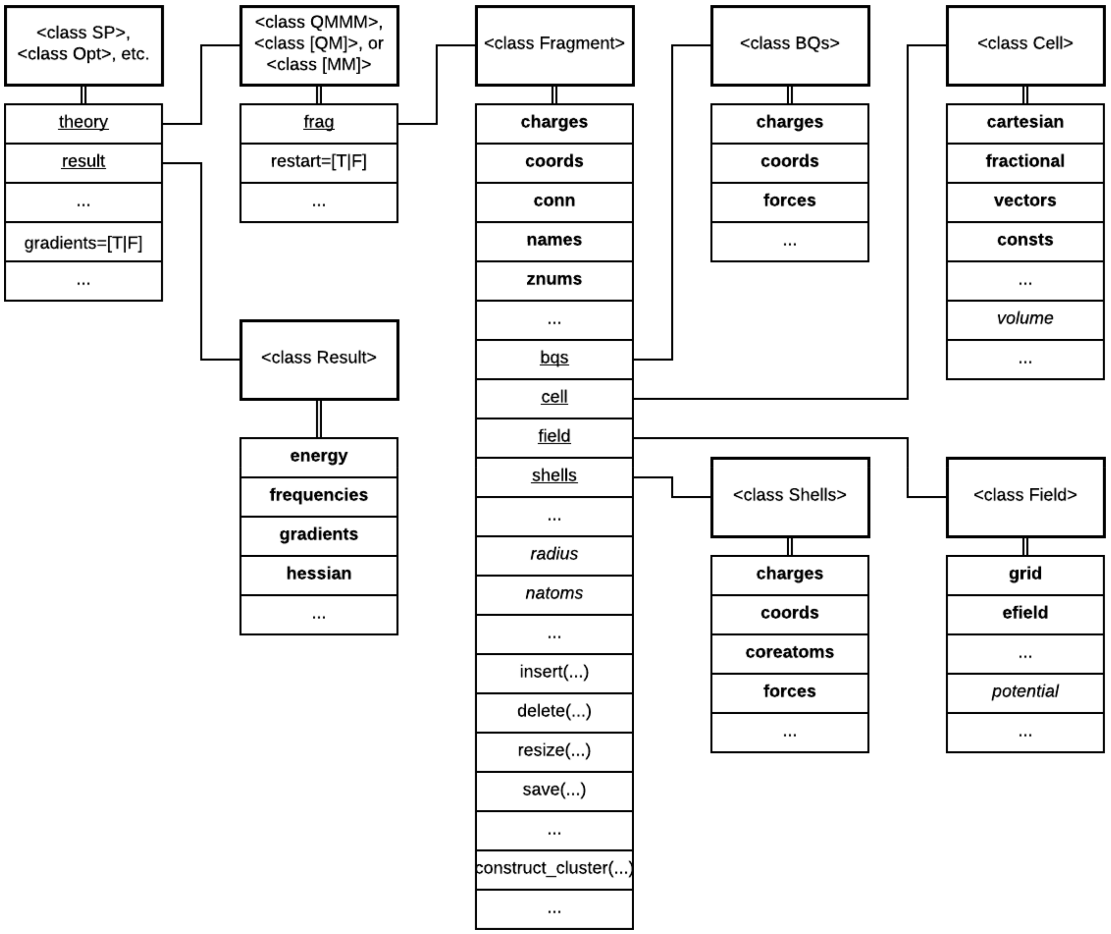


Figure 1. Python classes that define Py-ChemShell objects, their relationships to each other, and their main attributes and methods. Attributes in italics are scalar quantities, in bold are NumPy arrays, underlined are instances of associated classes, and those with parentheses are methods.

corresponding to each external software package interfaced to ChemShell. The classes are grouped by type of theory (QM or MM) and can be combined together using the `QMMM` class. Underneath this is the `Fragment` class that holds information on the chemical system such as the geometry, thus corresponding to the Fragment object in Tcl-ChemShell. In Py-ChemShell additional information such as shells (`Shells`), point charges (`BQs`), link atoms (`Links`), and periodic boundary conditions (`Cell`) are contained in subclasses that are accessible from the `Fragment` object.

Further classes include `Field`, which, as in Tcl-ChemShell, contains potential and field data on a grid, and `Result`, which collates the results of a calculation into a single object including energy, gradient, and Hessian information.

The user creates a Py-ChemShell object by instantiating one of the above classes. User instances can have arbitrary names. For example, in the script in section 2.1, `my_mol` is an instance of class `Fragment` while `my_qm` is an instance of the theory class `NWChem`. Each Py-ChemShell object instance contains data which is stored in instance variables, as with any other Python object. All array data (such as a list of atomic coordinates, for example) is stored in NumPy arrays to ensure high performance and ease of data manipulation. Each instance may also contain scalar data (such as the number of atoms, for example) and instances of other Py-ChemShell classes. The names of these attributes also serve as input keywords in the user interface, so that instances can be initialized when created. In the script in section 2.1, the line

```
my_mol = Fragment(coords='water.xyz')
```

creates an instance of the `Fragment` object named `my_mol` and initializes it by passing in the name of a file containing an XYZ geometry using the `coords` keyword, which refers to the NumPy array holding the geometry data. Once loaded, this data can then be accessed and edited using standard Python syntax as `my_mol.coords`.

Some other attributes are used as options to control the computational job. For example,

```
my_qm.restart = True
```

forces the QM calculation to restart based on a previously saved wave function from NWChem. Another set of attributes are read-only properties which are provided for the convenience of the user to enquire, such as `radius`, `volume`, and so on.

Py-ChemShell objects also contain methods, which are functions that act on the associated object; these correspond to the procedures in Tcl-ChemShell which are used to manipulate object data. For example, to remove the first atom from the `my_mol` fragment, the `delete()` method would be called as follows:

```
my_mol.delete(0)
```

2.3. Python/Fortran Coupling. While the ChemShell user interface has been reimplemented in Python and ChemShell objects are now implemented as Python classes, a Fortran layer is required for certain lower level operations.

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

These include the integration of modules written in Fortran such as the routines for cutting clusters and the DL-FIND geometry optimization library, and the ability to interface directly to external packages written in Fortran. The parallelization framework in Py-ChemShell has also been written in Fortran so that it is not necessary to launch a Python interpreter on every MPI process (see section 3).

To address this need we have developed a general-purpose library, DL_PY2F, which enables Py-ChemShell objects to be directly accessed from the Fortran layer so that data can be passed in a straightforward manner to and from Fortran modules and external libraries. When DL_PY2F is run, it creates a table of pointers to the Python object data (such as NumPy arrays and references to other objects) together with their datatypes. This table is accessed from the Fortran side through a convenient dictionary-like interface of key and value pairs, through which any data within the object can be obtained and amended. For example, for the DL-FIND optimization library described in section 2.6, the user options selected for a geometry optimization, together with the initial structure, are passed to DL-FIND by DL_PY2F. The calculated energy and gradients required over the course of the optimization are obtained via a callback mechanism. DL-FIND then updates the new sets of positions directly using DL_PY2F.

Unlike other Python/Fortran coupling approaches, DL_PY2F is specifically designed for interfacing with modern Fortran and fully supports complex datatypes such as those used by Py-ChemShell. Full technical details of DL_PY2F will be published in a companion paper.

2.4. External Interfaces. Py-ChemShell supports calculations using a range of QM and MM packages through user interfaces written in Python that automatically handle both the creation of input files and the parsing of outputs in a consistent manner across packages. Each interface is implemented as a Python class, with optional attributes to specify general settings, such as energy evaluation methods, and any program-specific settings for each software package. In the example input script in section 2.1, `my_qm` is defined as an instance of the `NWChem` class with an appropriate set of QM options.

For the initial release of Py-ChemShell, a subset of the interfaces available in Tcl-ChemShell were targeted, specifically those that are particularly well-suited for calculations on high-performance computing platforms. For each external code a loosely coupled interface is available, where the code is executed through a system call, or alternatively a directly linked interface where the external code is compiled into ChemShell as a library. The latter mode is important for parallel execution as described in section 3.

NWChem is an ab initio quantum chemistry software package developed by a consortium of scientists led by the Environmental Molecular Sciences Laboratory at the US Pacific Northwest National Laboratory.[72] It supports Hartree−Fock, density functional theory, and several high-level ab initio methods. NWChem is highly scalable with an architecture built on the Global Arrays toolkit (see section 3). The NWChem package is freely available under the open source Educational Community License 2.0. As part of the ChemShell redevelopment project, we have implemented a frozen density embedding model[73,74] in NWChem for use with ChemShell QM/MM calculations, which will be described in a follow-up paper.

GAMESS-UK is a quantum chemistry package developed by Daresbury Laboratory and collaborators[75] as part of the UK's Collaborative Computational Project for the Electronic Structure of Molecules (CCP1). It supports Hartree−Fock, DFT, and MCSCF calculations together with a variety of post Hartree−Fock methods. GAMESS-UK is highly scalable through MPI parallelism or Global Arrays builds. The GAMESS-UK package has a proprietary license but is available free of charge to UK-based researchers.

LSDalton is a quantum chemistry program which forms part of the Dalton suite[76] led by developers at the University of Oslo. LSDalton contains a highly efficient, linear-scaling implementation of Hartree−Fock and density functional theory suitable for large molecular systems with robust wave function and response optimization procedures. The LSDalton program is open source and available for download under the GNU LGPL v3 license. QM/MM calculations with ChemShell/LSDalton support hybrid MPI/OpenMP execution and have been shown to give excellent scaling on high performance platforms.[77]

GULP (General Utility Lattice Program) is a molecular mechanics software package developed at Curtin University.[78] ChemShell fully supports all forms of QM/MM embedding with GULP, including polarized embedding using shell-model force fields. It is particularly well-suited for QM/MM materials chemistry calculations as it offers excellent support for materials force fields. GULP has a proprietary license but is free of charge for academic users.

DL_POLY 4 is a general purpose molecular dynamics program[79] developed by a team led by Daresbury Laboratory under the UK's Collaborative Computational Project for Computer Simulation of Condensed Phases (CCP5). In Tcl-ChemShell, the original replicated data version of DL_POLY ("DL_POLY Classic") was available as a built-in module, which offered support for QM/MM calculations of up to roughly 50000 atoms. In Py-ChemShell, we have implemented an interface to the more modern DL_POLY 4 program, which is parallelized based on a domain decomposition strategy allowing much larger MM environments to be included in the QM/MM calculation (see section 3). DL_POLY 4 has a proprietary license but is free of charge for academic users.

Further QM and MM programs will be interfaced to Py-ChemShell in future releases.

2.5. QM/MM Driver. The original Tcl/C "hybrid" module for QM/MM calculations in Tcl-ChemShell has been rewritten in Python with an emphasis on flexibility to support ongoing embedding model developments. In the initial release of Py-ChemShell, the standard additive QM/MM scheme has been implemented as described in the original QUASI project publication,[1] with full support for both link-atom and ionic boundary methods for handling the QM/MM boundary region. Mechanical, electrostatic, and polarized (shell model) embedding are all implemented in full. The charge shifting scheme for avoiding overpolarization of link atoms is available and automatically applied when electrostatic embedding is selected. Link atom forces are resolved as in the original implementation.

QM/MM calculations are performed in an analogous manner to Tcl-ChemShell, with a `QMMM` class used to define the overall calculation, which contains subclasses to define the QM and MM components. The new QM/MM driver has been rigorously validated against the original Tcl-ChemShell implementation. In the initial release, MM force fields must

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

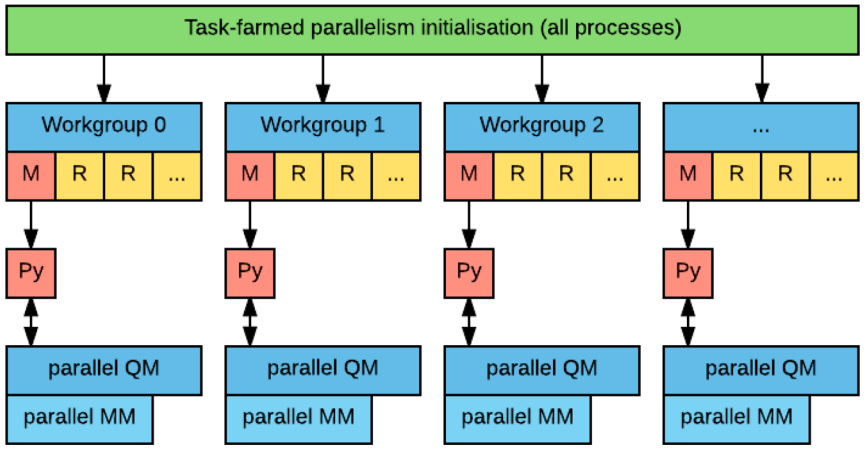


Figure 2. Parallel MPI framework in Py-ChemShell following implementation of task-farming parallelization. M, Master; R, Replica; and Py, Python interpreter.

be defined in either GULP or DL_POLY format, but future releases will include a mechanism for automated import of standard MM force fields to support biomolecular QM/MM calculations.

2.6. Geometry Optimization with DL-FIND. DL-FIND is an open-source geometry optimization library[80] developed at Daresbury Laboratory. It is the recommended optimizer module in Tcl-ChemShell and Py-ChemShell, implementing a wide range of optimization algorithms. Local energy minimization methods include steepest descent, conjugate gradients, Newton−Raphson, Quasi-Newton (BFGS), and damped molecular dynamics. Transition states can be located via partitioned rational function optimization (P-RFO) or the dimer method,[81] and full minimum energy reaction paths can be optimized using the nudged elastic band (NEB) method.[82,83] Global optimization can be performed using a genetic algorithm or stochastic search, and multiple electronic state crossings can be found using three conical intersection optimization algorithms.[84,85] Reaction rates can be calculated using harmonic transition-state theory and quantum tunneling paths optimized using instanton theory.[86]

During a DL-FIND optimization, the main optimization cycle is controlled by the chosen optimizer, and energies and gradients are requested from ChemShell as required. The modular design of ChemShell allows any choice of QM, MM, or combined QM/MM method to be used in conjunction with DL-FIND, which needs no information on the underlying method used. Several techniques have however been implemented within DL-FIND to facilitate the optimization of large QM/MM systems. For example, the calculation of a Hessian matrix rapidly becomes intractable for large systems, but this can be avoided by using the limited memory variant of the BFGS algorithm[87] (L-BFGS) for minimization and dimer method for transition state optimization. A number of coordinate systems are implemented, which are converted automatically within DL-FIND to/from Cartesian coordinates, including the highly efficient hybrid delocalized coordinate system (HDLC), which calculates delocalized internal coordinates using a divide-and-conquer approach that splits a complex system into manageable residues.[88] Microiterative methods have also been implemented for minimization, transition state searches, and nudged elastic band, in order to increase the efficiency of QM/MM optimization.[89] These methods relax the MM environment fully after every QM step, relying on the fact that MM energy and gradient evaluations are usually much quicker than QM to reduce the overall time to optimization.

In Py-ChemShell, DL-FIND is invoked through the `Opt` Python class, which is designed to be extensible to enable linking in other optimization modules in future. The `Opt` class interfaces with DL-FIND using the DL_PY2F library as described in section 2.3, with optimized structures returned as Py-ChemShell objects.

2.7. Generation of Finite Cluster Models. ChemShell uses an embedded cluster approach for QM/MM calculations of materials, in which a finite cluster is cut from a periodic system in order to model localized states such as point defects and adsorbed molecules.[1] This approach has a number of advantages over periodic models: the method is well-suited to QM packages that do not support periodic MM background charges, there are no inaccuracies due to periodically repeating defects, charged states can be readily calculated, and a common reference energy can be defined for ionization potentials of different systems. The disadvantage is that periodic electrostatic interactions are lost, but these can be compensated for by fitting point charges around the outside of the cluster to match the electrostatic potential and derivatives in a sampling region of the original periodic system.

In Tcl-ChemShell, a cluster cutting module is available that automates much of the cluster setup process, including the charge fitting scheme. This module has been ported to Py-ChemShell with a new Python wrapper interfacing the code via DL_PY2F.

The point charge fitting scheme can be computationally intensive as Ewald summations over the periodic system are required to calculate the electrostatic potential and its derivatives over a set of atomic centers and grid points in the sampling region. These calculations are however independent and so amenable to parallelization. An MPI replicated data version of the module is currently under development and will be released in a future version of Py-ChemShell.

3. TASK-FARMING PARALLELISM

3.1. Methodology. Task-farming parallelism is useful in situations where a set of independent calculations have to be performed.[90,91] The task farm consists of all available processors, which are then divided into subsets of processors

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

called workgroups. The tasks are split between the workgroups, which then work on the separate tasks in parallel. As the calculations are independent, no information needs to be exchanged between workgroups during this time, and sharing of results can be postponed until all the tasks have completed. In computational chemistry, it is common to carry out calculations containing a number of single-point energy evaluations that are independent from each other. Typical examples of applications include the nudged elastic band method for reaction path optimization, finite-difference numerical gradient and Hessian calculations, and optimization methods based on a population of structures (e.g., genetic algorithms).

Figure 2 illustrates the task-farmed MPI parallel environment of Py-ChemShell. The MPI processes are evenly grouped into workgroups, each containing one Master process and zero or more Replica processes. Each Master process launches a Python interpreter instance, which parses the user input script and invokes external programs to execute computational chemistry tasks. If the size of the workgroup is greater than one, these tasks will be run in parallel across the Master and Replica processes. This results in a two-level parallelization framework that allows Py-ChemShell to execute multiple parallel energy evaluations simultaneously.

This approach follows the previously implemented task-farming framework in Tcl-ChemShell[91] but goes beyond it in several important respects.

First, as indicated in Figure 2, we allow parallel tasks within workgroups to be executed either over the full workgroup or on a subset of the workgroup processes. This is useful for QM/MM simulations where the QM part of the calculation is usually the dominant computational expense and the most amenable to parallelization, whereas the MM region is generally relatively small and quick to compute, such that using a subset of the processes may reduce the time to solution due to avoiding unnecessary parallel communication overheads.

Second, task-farming with the Global Arrays (GA) library is fully supported. The GA library is used in a variety of popular quantum chemistry packages, including NWChem and GAMESS-UK, and our implementation ensures that task-farmed QM/MM calculations with NWChem can be performed in ChemShell. In order to support task-farming, GA processor groups are set up at the initialization of the calculation to match exactly the MPI workgroups, such that all GA and MPI calls apply to the same set of processes.

Third, we have implemented new parallel tasks within Py-ChemShell, including a new numerical finite difference gradient module, which can be helpful when using levels of theory for which analytic gradients are not available. The finite difference gradient module is seamlessly integrated into ChemShell so that it can be used in place of an analytical gradient wherever the user requests it and can be run either in serial or parallel as desired. One-point and two-point finite difference gradients are available.

3.2. QM/MM Task-Farming Benchmarks. To validate and benchmark the task-farming implementation, we performed a series of two-point finite difference gradient calculations. An MgO cluster containing 2263 atoms and 1133 shell centers was constructed (see Figure 3), and numerical gradients were calculated over the 34 QM atoms (102 degrees of freedom, i.e., 204 QM/MM single-point calculations). In each single-point calculation, a full relaxation of the shell centers is undertaken until the QM and MM regions are self-consistently converged (tolerance: 1.0×10^{-4} a.u.), after which a final energy evaluation is carried out for estimating the gradients.

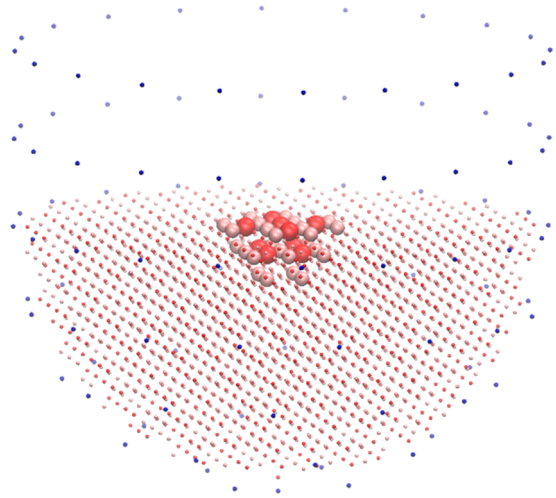


Figure 3. QM/MM-embedded MgO system used for two-point finite-difference numerical gradient calculations. The cluster consists of 2263 atoms in total, with the QM region comprising 25 Mg (pink VDW representation) and 9 O (red VDW representation) atoms and the MM region comprising 2229 atoms, with 107 background point charges surrounding the cluster (in blue). Graphic produced with VMD.[92]

In Table 1, we present example QM/MM benchmark calculations using GAMESS-UK as the external QM driver and GULP as the MM driver. The QM calculations were performed using density functional theory with the BLYP functional.[93,94] The atoms in the QM region were described using a modified def2-SVP basis set[95] with Stuttgart effective core potentials[96] placed on the Mg ions in the QM region and on the cations in the boundary region. The GULP force field used was as described in previous studies of MgO.[57] Task-farming scalability has been benchmarked by varying the number of workgroups in a fixed allocation of 48 cores on the ARCHER supercomputing service (a Cray XC30 platform with two 2.7 GHz 12-core Ivy Bridge processors per node). The table lists the wall time for these calculations with respect to changing the number of workgroups ($n_{\text{workgroups}}$), and Figure 4 visualizes the profiles. Overall, we observe a 5.7-fold speedup for a 24 workgroup calculation compared to the non-task-farmed reference ($n_{\text{workgroups}} = 1$). Note that the scaling profile of the task-farmed calculations is nonlinear as $n_{\text{workgroups}}$ grows, and it always has a minimum point. This is due to the increase of computational time for a single-point GAMESS-UK or GULP execution as n_{cores} per workgroup decreases (see dashed lines in Figure 4).

To demonstrate the scalability of task-farming parallelization to higher core counts, we also carried out similar two-point finite difference gradient calculations while increasing both $n_{\text{workgroups}}$ and the total number of processor cores (n_{cores}), while fixing n_{cores} per workgroup at 48. Results are shown in Table 2 and Figure 5. The speedup obtained in this way on 8 workgroups (384 ARCHER cores) is 7.26, illustrating the inherent linearity in the task-farming parallelization of the finite difference task. In this case, the time used for a single-point GAMESS-UK or GULP execution remains roughly equal because the available n_{cores} per workgroup is a constant

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

Table 1. Two-Point Finite Difference QM/MM Gradient Task-Farming Benchmark of MgO on a Fixed total of 48 Cores Using GAMESS-UK/GULP[a]

$n_{workgroups}$	1	2	4	8	16	24	48
n_{cores}				48			
$n_{cores/workgroup}$	48	24	12	6	3	2	1
n_{SP}	412	416	424	440	472	504	600
t_{SP}/s (GAMESS-UK)	2.9	4.2	5.7	9.8	18.0	25.8	57.6
t_{SP}/s (GULP)	2.1	2.5	2.6	3.0	4.1	5.0	8.1
t_{total} (h)	1.48	0.85	0.50	0.33	0.27	0.26	0.32
speedup vs 1 workgroup	1.0	1.7	2.9	4.5	5.4	5.7	4.6

[a]The total wall time (t_{total} in h) for the calculation quoted is the average over workgroups, and the time for single-point (SP) calculations is averaged over all calculations (t_{SP} in sec). $n_{workgroups}$, n_{cores}, $n_{cores/workgroup}$, and n_{SP} refer to the number of workgroups, processor cores, processor cores per workgroup, and single-point calculations, respectively. The active region for the calculation is the QM region (102 degrees of freedom). System size: 2263 atoms, of which 34 are QM, plus 1133 shell centers and 107 fitted point charges. Shell relaxation tolerance: 1.0×10^{-4} a.u. Calculations performed on 2 nodes of ARCHER (48 cores).

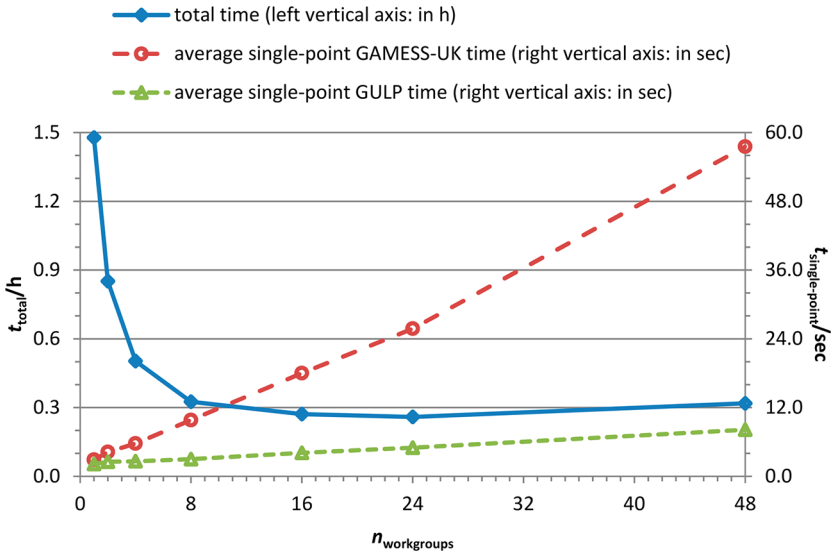


Figure 4. Benchmark of task-farmed two-point finite-difference numerical gradients calculation on QM/MM embedded MgO system against the number of workgroups on a fixed total number of 48 cores. The total wall time (t_{total} in h) for the calculation quoted is the average over workgroups, and the time for single-point (SP) calculations is averaged over all calculations (t_{SP} in sec). The active region for the calculation is the QM region (102 degrees of freedom). System size: 2263 atoms, of which 34 are QM, plus 1133 shell centers and 107 fitted point charges. Shell relaxation tolerance: 1.0×10^{-4} a.u. Calculations performed on 2 nodes of ARCHER (48 cores) using GAMESS-UK/GULP.

Table 2. Two-Point Finite Difference QM/MM Gradient Task-Farming Benchmark of MgO for Fixed Size Workgroups Using GAMESS-UK/GULP[a]

$n_{workgroups}$	1	2	4	8
n_{cores}	48	96	192	384
$n_{cores/workgroup}$		48		
n_{SP}	412	416	424	440
t_{SP}/s (GAMESS-UK)	2.9	2.6	2.8	3.1
t_{SP}/s (GULP)	2.1	1.9	2.0	2.0
t_{total} (h)	1.48	0.72	0.37	0.20
parallel scaling	1.00	2.05	4.04	7.26

[a]Each workgroup consists of 48 cores. The total wall time (t_{total} in h) for the calculation quoted is the average over workgroups, and the time for single-point (SP) calculations is averaged over all calculations (t_{SP} in sec). $n_{workgroups}$, n_{cores}, $n_{cores/workgroup}$, and n_{SP} are defined as in Table 1. The active region for the calculation is the QM region (102 degrees of freedom). System size: 2263 atoms, of which 34 are QM, plus 1133 shell centers and 107 fitted point charges. Shell relaxation tolerance: 1.0×10^{-4} a.u. Calculations performed on ARCHER.

number. Note that superlinear scaling is possible because of subtle variations in the shell relaxation starting points, but this is an artifact of the benchmark rather than a systematic effect.

Although these benchmarks cannot be compared in a straightforward manner with Tcl-ChemShell (as the finite difference gradient module is not available in that program), the observed performance is broadly in line with the previously implemented task-farming parallelization framework,[91] as would be expected given that the overall timings are determined for the most part by the parallel scaling performance of the GAMESS-UK and GULP packages. In the next section, we consider a system which is beyond the capability of Tcl-ChemShell to calculate.

3.3. QM/MM Nanoparticle Benchmarks. To assess the performance of Py-ChemShell for large QM/MM systems, we have carried out NWChem/DL_POLY 4 single-point QM/MM energy evaluations for a ZrO_2 nanoparticle of radius 75.0 Å, containing 162994 atoms, as illustrated in Figure 6. It is not possible to treat such a system in Tcl-ChemShell because the built-in version of DL_POLY Classic used for MM energy evaluations uses replica data parallelism with an upper limit of

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

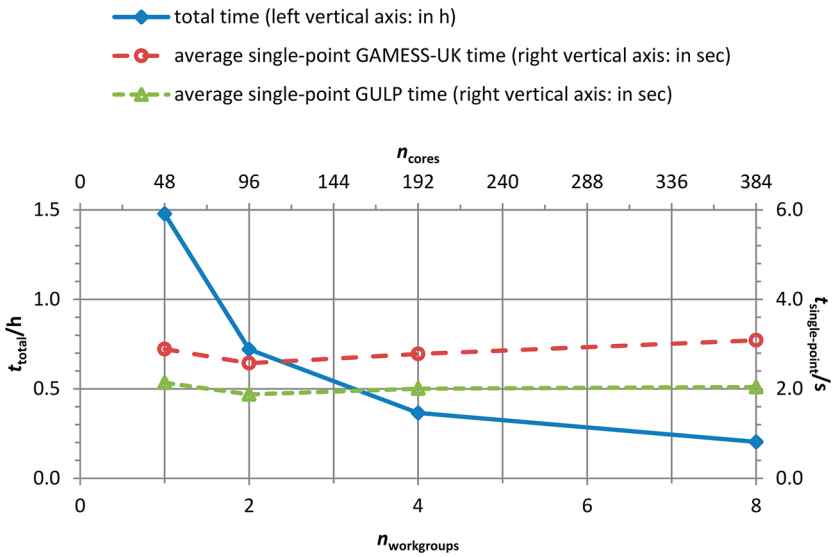


Figure 5. Benchmark of task-farmed two-point finite-difference numerical gradients calculation on QM/MM embedded MgO system against number of cores and workgroups: each workgroup consists of 48 cores. The total wall time (t_{total} in h) for the calculation quoted is the average over workgroups, and the time for single-point (SP) calculations is averaged over all calculations (t_{SP} in sec). The active region for the calculation is the QM region (102 degrees of freedom). System size: 2263 atoms, of which 34 are QM, plus 1133 shell centers and 107 fitted point charges. Shell relaxation tolerance: 1.0×10^{-4} a.u. Calculations performed on ARCHER using GAMESS-UK/GULP.

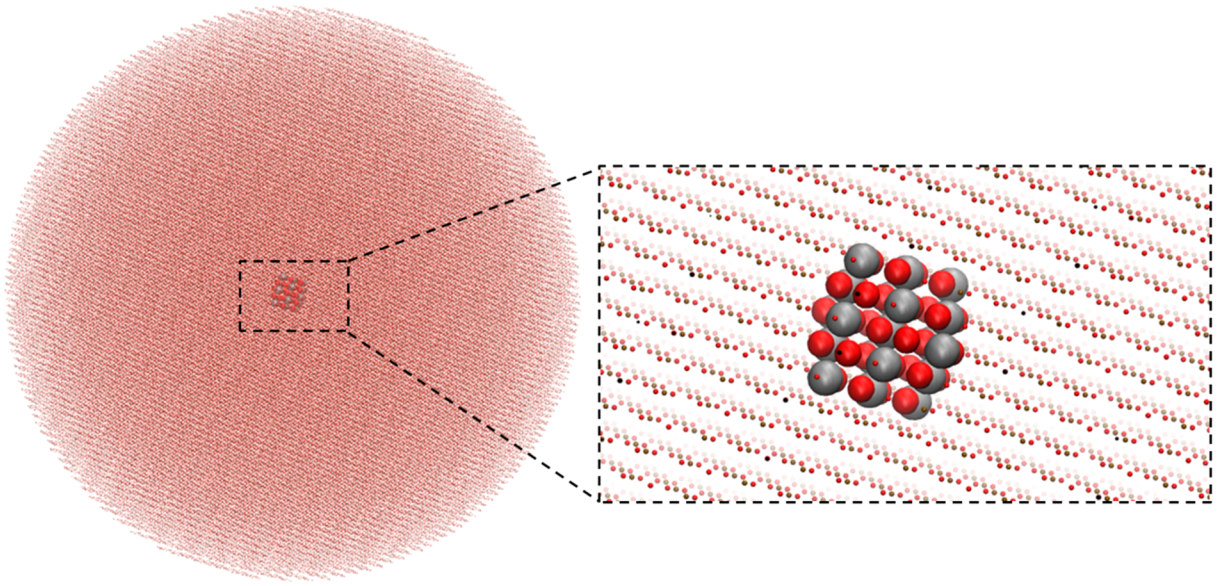


Figure 6. ZrO_2 nanoparticle of 162994 atoms, including 19 Zr (gray VDW representation) and 32 O (red VDW representation) atoms in the QM region. Graphics generated using VMD.[92]

roughly 50000 atoms. This limit is removed in Py-ChemShell through interfacing to DL_POLY 4. The QM calculations were carried out at the Hartree−Fock level using the def2-TZVP basis set,[95] with the corresponding ECP[97] for Zr. The MM force field used was as described in previous studies of ZrO_2.[98]

The resulting time used for the calculations are listed in Table 3, which shows that Py-ChemShell scales efficiently for this system. To demonstrate the facility for using different number of processes for the QM and MM parts of the calculation, we have restricted the DL_POLY energy evaluations to 8 cores, while the computationally far more intensive NWChem calculations uses the full set of processes. The overall wall time is therefore dominated by the NWChem QM calculation (>99%), and good scaling is observed as the number of cores increases (speedup column).

Table 3. Single-Point Energy Evaluations of a QM/MM ZrO_2 Nanoparticle System of 162994 Atoms Using NWChem/DL_POLY 4[a]

n_{cores}	n_{nodes}	n_{cores} (DL_POLY 4)	t_{NWChem} (h)	t_{DL_POLY} (s)	t_{total} (h)	speedup
24	1	8	11.99	50.1	12.21	1.00
48	2	8	6.15	49.3	6.38	1.91
96	4	8	3.26	58.8	3.47	3.52
192	8	8	1.71	50.8	1.92	6.36

[a]Calculations run on ARCHER.

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

4. CONCLUSION AND OUTLOOK

The ChemShell software package has been completely rewritten with the aim of providing a modern, scriptable platform for multiscale computational chemistry. This redevelopment of ChemShell features an easy-to-use Python-based interface and a high-performance Fortran parallelization framework. This article describes the features of the code available in the initial public release in December 2017 (version 17.0), which is focused on QM/MM calculations for materials chemistry. However, the code continues to undergo rapid development, and a number of additional features are targeted for release in the near future. These include a built-in molecular dynamics driver incorporating the open source DL_POLY Classic code[99] and support for biomolecular QM/MM calculations through automatic import of biomolecular force fields via an interface to the DL_FIELD program,[100] bringing these aspects of Py-ChemShell up to feature equivalence with Tcl-ChemShell. Further functionality not currently available in Tcl-ChemShell will continue to be introduced, including full support for embedded cluster calculations with the ORCA QM code[101] and a new interface to the density functional tight binding package DFTB+ for fast, semiempirical DFT calculations.[102] In ongoing projects, we are implementing a periodic QM/MM embedding scheme together with interfaces to periodic QM programs including CP2K[103] and CRYSTAL[104] and developing new methods for spectroscopic signatures based on the frozen density embedding model developed in NWChem. Future releases will also include methods for "adaptive" QM/MM calculations, where molecules can be exchanged between the QM and MM regions. A graphical user interface is also under development through a plug-in to the open source Aten visualizer.[105]

For further information about the ChemShell project and to download the Py-ChemShell code, please visit www.chemshell.org.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jctc.8b01036.

Summary description of archived structures, basis sets, effective core potentials, and force field definitions (PDF)

An archive of structures, basis sets, effective core potentials, and force field definitions used in the MgO and ZrO_2 benchmark calculations (ZIP)

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: thomas.keal@stfc.ac.uk.

ORCID

Matthew R. Farrow: 0000-0002-1579-3395
Andrew J. Logsdail: 0000-0002-2277-415X
Alexey A. Sokol: 0000-0003-0178-1147
Thomas W. Keal: 0000-0001-8747-3975

Present Address

⊥P.F.: Institute for Materials and Processes, School of Engineering, The University of Edinburgh, Sanderson Building, Robert Stevenson Road, The King's Buildings, Edinburgh EH9 3FB.

Funding

The redevelopment of ChemShell was funded by EPSRC under the "Scalable Quantum Chemistry with Flexible Embedding" Grants EP/I030662/1 and EP/K038419/1. Implementation of the task-farming parallelization framework was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service under Grant eCSE08-14 "Task-Farming Parallelisation of Python-ChemShell for Nanomaterials". Maintenance and ongoing development of ChemShell is supported by CoSeC, the Computational Science Centre for Research Communities, through the HEC Materials Chemistry Consortium (EP/L000202/1) and CCP5 (EP/M022617/1).

Notes

The authors declare no competing financial interest.

■ REFERENCES

(1) Sherwood, P.; de Vries, A. H.; Guest, M. F.; Schreckenbach, G.; Catlow, C. R. A.; French, S. A.; Sokol, A. A.; Bromley, S. T.; Thiel, W.; Turner, A. J.; Billeter, S.; Terstegen, F.; Thiel, S.; Kendrick, J.; Rogers, S. C.; Casci, J.; Watson, M.; King, F.; Karlsen, E.; Sjøvoll, M.; Fahmi, A.; Schäfer, A.; Lennartz, C. QUASI: A general purpose implementation of the QM/MM approach and its application to problems in catalysis. *J. Mol. Struct.: THEOCHEM* **2003**, *632*, 1−28.

(2) Metz, S.; Kästner, J.; Sokol, A. A.; Keal, T. W.; Sherwood, P. ChemShell-a modular software package for QM/MM simulations. *WIREs Comput. Mol. Sci.* **2014**, *4*, 101−110.

(3) Senn, H. M.; Thiel, W. QM/MM Methods for Biomolecular Systems. *Angew. Chem., Int. Ed.* **2009**, *48*, 1198−1229.

(4) Karasulu, B.; Thiel, W. Amine Oxidation Mediated by N-Methyltryptophan Oxidase: Computational Insights into the Mechanism, Role of Active-Site Residues, and Covalent Flavin Binding. *ACS Catal.* **2015**, *5*, 1227−1239.

(5) Castro, L.; Crawford, L.; Mutengwa, A.; Götze, J. P.; Bühl, M. Insights into structure and redox potential of lignin peroxidase from QM/MM calculations. *Org. Biomol. Chem.* **2016**, *14*, 2385−2389.

(6) Götze, J. P.; Bühl, M. Laccase Redox Potentials: pH Dependence and Mutants, a QM/MM Study. *J. Phys. Chem. B* **2016**, *120*, 9265−9276.

(7) Finkelmann, A. R.; Senn, H. M.; Reiher, M. Hydrogen-activation mechanism of [Fe] hydrogenase revealed by multi-scale modeling. *Chem. Sci.* **2014**, *5*, 4474−4482.

(8) Ciaccafava, A.; Tombolelli, D.; Domnik, L.; Fesseler, J.; Jeoung, J. H.; Dobbek, H.; Mroginski, M. A.; Zebger, I.; Hildebrandt, P. When the inhibitor tells more than the substrate: the cyanide-bound state of a carbon monoxide dehydrogenase. *Chem. Sci.* **2016**, *7*, 3162−3171.

(9) Su, H.; Sheng, X.; Zhu, W.; Ma, G.; Liu, Y. Mechanistic Insights into the Decoupled Desaturation and Epoxidation Catalyzed by Dioxygenase AsqJ Involved in the Biosynthesis of Quinolone Alkaloids. *ACS Catal.* **2017**, *7*, 5534−5543.

(10) Wang, B.; Cao, Z.; Sharon, D. A.; Shaik, S. Computations Reveal a Rich Mechanistic Variation of Demethylation of N-Methylated DNA/RNA Nucleotides by FTO. *ACS Catal.* **2015**, *5*, 7077−7090.

(11) Faponle, A. S.; Seebeck, F. P.; de Visser, S. P. Sulfoxide Synthase versus Cysteine Dioxygenase Reactivity in a Nonheme Iron Enzyme. *J. Am. Chem. Soc.* **2017**, *139*, 9259−9270.

(12) Timmins, A.; Saint-Andre, M.; de Visser, S. P. Understanding How Prolyl-4-hydroxylase Structure Steers a Ferryl Oxidant toward Scission of a Strong C-H Bond. *J. Am. Chem. Soc.* **2017**, *139*, 9855−9866.

(13) Roy, S.; Kästner, J. Synergistic Substrate and Oxygen Activation in Salicylate Dioxygenase Revealed by QM/MM Simulations. *Angew. Chem., Int. Ed.* **2016**, *55*, 1168−1172.

(14) Huang, J.; Li, C.; Wang, B.; Sharon, D. A.; Wu, W.; Shaik, S. Selective Chlorination of Substrates by the Halogenase SyrB2 Is Controlled by the Protein According to a Combined Quantum

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

Mechanics/Molecular Mechanics and Molecular Dynamics Study. *ACS Catal.* **2016**, *6*, 2694−2704.
(15) Rugg, G.; Senn, H. M. Formation and structure of the ferryl [Fe = O] intermediate in the non-haem iron halogenase SyrB2: classical and QM/MM modelling agree. *Phys. Chem. Chem. Phys.* **2017**, *19*, 30107−30119.
(16) Pérez-Gallegos, A.; Garcia-Viloca, M.; González-Lafont, À.; Lluch, J. M. SP20 Phosphorylation Reaction Catalyzed by Protein Kinase A: QM/MM Calculations Based on Recently Determined Crystallographic Structures. *ACS Catal.* **2015**, *5*, 4897−4912.
(17) Mendoza, F.; Gómez, H.; Lluch, J. M.; Masgrau, L. α1,4-N-Acetylhexosaminyltransferase EXTL2: The Missing Link for Understanding Glycosidic Bond Biosynthesis with Retention of Configuration. *ACS Catal.* **2016**, *6*, 2577−2589.
(18) Escorcia, A. M.; Sen, K.; Daza, M. C.; Doerr, M.; Thiel, W. Quantum Mechanics/Molecular Mechanics Insights into the Enantioselectivity of the O-Acetylation of (R,S)-Propranolol Catalyzed by Candida antarctica Lipase B. *ACS Catal.* **2017**, *7*, 115−127.
(19) Vasilevskaya, T.; Khrenova, M. G.; Nemukhin, A. V.; Thiel, W. Mechanism of proteolysis in matrix metalloproteinase-2 revealed by QM/MM modeling. *J. Comput. Chem.* **2015**, *36*, 1621−1630.
(20) Vasilevskaya, T.; Khrenova, M. G.; Nemukhin, A. V.; Thiel, W. Methodological aspects of QM/MM calculations: A case study on matrix metalloproteinase-2. *J. Comput. Chem.* **2016**, *37*, 1801−1809.
(21) Zhu, W.; Liu, Y. Ring Contraction Catalyzed by the Metal-Dependent Radical SAM Enzyme: 7-Carboxy-7-deazaguanine Synthase from B. multivorans. Theoretical Insights into the Reaction Mechanism and the Influence of Metal Ions. *ACS Catal.* **2015**, *5*, 3953−3965.
(22) Ma, G.; Zhu, W.; Su, H.; Cheng, N.; Liu, Y. Uncoupled Epimerization and Desaturation by Carbapenem Synthase: Mechanistic Insights from QM/MM Studies. *ACS Catal.* **2015**, *5*, 5556−5566.
(23) Shaik, S.; Cohen, S.; Wang, Y.; Chen, H.; Kumar, D.; Thiel, W. P450 enzymes: their structure, reactivity, and selectivity-modeled by QM/MM calculations. *Chem. Rev.* **2010**, *110*, 949−1017.
(24) Wang, B. J.; Li, C. S.; Dubey, K. D.; Shaik, S. Quantum Mechanical/Molecular Mechanical Calculated Reactivity Networks Reveal How Cytochrome P450cam and Its T252A Mutant Select Their Oxidation Pathways. *J. Am. Chem. Soc.* **2015**, *137*, 7379−7390.
(25) Dubey, K. D.; Wang, B. J.; Vajpai, M.; Shaik, S. MD simulations and QM/MM calculations show that single-site mutations of cytochrome $P450_{BM3}$ alter the active site's complexity and the chemoselectivity of oxidation without changing the active species. *Chem. Sci.* **2017**, *8*, 5335−5344.
(26) Dubey, K. D.; Wang, B. J.; Shaik, S. Molecular Dynamics and QM/MM Calculations Predict the Substrate-Induced Gating of Cytochrome P450 BM3 and the Regio- and Stereoselectivity of Fatty Acid Hydroxylation. *J. Am. Chem. Soc.* **2016**, *138*, 837−845.
(27) Ramanan, R.; Dubey, K. D.; Wang, B. J.; Mandal, D.; Shaik, S. Emergence of Function in P450-Proteins: A Combined Quantum Mechanical/Molecular Mechanical and Molecular Dynamics Study of the Reactive Species in the H_2O_2-Dependent Cytochrome $P450_{SPα}$ and Its Regio- and Enantioselective Hydroxylation of Fatty Acids. *J. Am. Chem. Soc.* **2016**, *138*, 6786−6797.
(28) Li, A. T.; Wang, B. J.; Ilie, A.; Dubey, K. D.; Bange, G.; Korendovych, I. V.; Shaik, S.; Reetz, M. T. A redox-mediated Kemp eliminase. *Nat. Commun.* **2017**, *8*, 14876.
(29) Metz, S. N_2O Formation via Reductive Disproportionation of NO by Mononuclear Copper Complexes: A Mechanistic DFT Study. *Inorg. Chem.* **2017**, *56*, 3820−3833.
(30) Chen, X.; Ma, G.; Sun, W.; Dai, H.; Xiao, D.; Zhang, Y.; Qin, X.; Liu, Y.; Bu, Y. Water promoting electron hole transport between tyrosine and cysteine in proteins via a special mechanism: double proton coupled electron transfer. *J. Am. Chem. Soc.* **2014**, *136*, 4515−4524.
(31) Bier, D.; Mittal, S.; Bravo-Rodriguez, K.; Sowislok, A.; Guillory, X.; Briels, J.; Heid, C.; Bartel, M.; Wettig, B.; Brunsveld, L.; Sanchez-Garcia, E.; Schrader, T.; Ottmann, C. The Molecular Tweezer CLR01 Stabilizes a Disordered Protein-Protein Interface. *J. Am. Chem. Soc.* **2017**, *139*, 16256−16263.
(32) Schirmeister, T.; Kesselring, J.; Jung, S.; Schneider, T. H.; Weickert, A.; Becker, J.; Lee, W.; Bamberger, D.; Wich, P. R.; Distler, U.; Tenzer, S.; Johe, P.; Hellmich, U. A.; Engels, B. Quantum Chemical-Based Protocol for the Rational Design of Covalent Inhibitors. *J. Am. Chem. Soc.* **2016**, *138*, 8332−8335.
(33) Wang, B.; Usharani, D.; Li, C.; Shaik, S. Theory uncovers an unusual mechanism of DNA repair of a lesioned adenine by AlkB enzymes. *J. Am. Chem. Soc.* **2014**, *136*, 13895−13901.
(34) Sadeghian, K.; Ochsenfeld, C. Unraveling the Base Excision Repair Mechanism of Human DNA Glycosylase. *J. Am. Chem. Soc.* **2015**, *137*, 9824−9831.
(35) Sen, K.; Horrell, S.; Kekilli, D.; Yong, C. W.; Keal, T. W.; Atakisi, H.; Moreau, D. W.; Thorne, R. E.; Hough, M. A.; Strange, R. W. Active-site protein dynamics and solvent accessibility in native Achromobacter cycloclastes copper nitrite reductase. *IUCrJ* **2017**, *4*, 495−505.
(36) Horrell, S.; Kekilli, D.; Sen, K.; Owen, R. L.; Dworkowski, F. S. N.; Antonyuk, S. V.; Keal, T. W.; Yong, C. W.; Eady, R. R.; Hasnain, S. S.; Strange, R. W.; Hough, M. A. Enzyme catalysis captured using multiple structures from one crystal at varying temperatures. *IUCrJ* **2018**, *5*, 283−292.
(37) Costa, P.; Trosien, I.; Fernandez-Oliva, M.; Sanchez-Garcia, E.; Sander, W. The fluorenyl cation. *Angew. Chem., Int. Ed.* **2015**, *54*, 2656−2660.
(38) Nadal-Ferret, M.; Gelabert, R.; Moreno, M.; Lluch, J. M. Are there really low-barrier hydrogen bonds in proteins? The case of photoactive yellow protein. *J. Am. Chem. Soc.* **2014**, *136*, 3542−3552.
(39) Wang, C.; Chen, H. Convergent Theoretical Prediction of Reactive Oxidant Structures in Diiron Arylamine Oxygenases AurF and CmlI: Peroxo or Hydroperoxo? *J. Am. Chem. Soc.* **2017**, *139*, 13038−13046.
(40) Bühl, M.; DaBell, P.; Manley, D. W.; McCaughan, R. P.; Walton, J. C. Bicarbonate and Alkyl Carbonate Radicals: Structural Integrity and Reactions with Lipid Components. *J. Am. Chem. Soc.* **2015**, *137*, 16153−16162.
(41) Knorr, J.; Sokkar, P.; Schott, S.; Costa, P.; Thiel, W.; Sander, W.; Sanchez-Garcia, E.; Nuernberger, P. Competitive solvent-molecule interactions govern primary processes of diphenylcarbene in solvent mixtures. *Nat. Commun.* **2016**, *7*, 12968.
(42) Ganguly, A.; Luong, T. Q.; Brylski, O.; Dirkmann, M.; Möller, D.; Ebbinghaus, S.; Schulz, F.; Winter, R.; Sanchez-Garcia, E.; Thiel, W. Elucidation of the Catalytic Mechanism of a Miniature Zinc Finger Hydrolase. *J. Phys. Chem. B* **2017**, *121*, 6390−6398.
(43) Mancini, D. T.; Sen, K.; Barbatti, M.; Thiel, W.; Ramalho, T. C. Excited-State Proton Transfer Can Tune the Color of Protein Fluorescent Markers. *ChemPhysChem* **2015**, *16*, 3444−3449.
(44) Armengol, P.; Sporkel, L.; Gelabert, R.; Moreno, M.; Thiel, W.; Lluch, J. M. Ultrafast action chemistry in slow motion: atomistic description of the excitation and fluorescence processes in an archetypal fluorescent protein. *Phys. Chem. Chem. Phys.* **2018**, *20*, 11067−11080.
(45) Chang, X. P.; Gao, Y. J.; Fang, W. H.; Cui, G.; Thiel, W. Quantum Mechanics/Molecular Mechanics Study on the Photoreactions of Dark- and Light-Adapted States of a Blue-Light YtvA LOV Photoreceptor. *Angew. Chem., Int. Ed.* **2017**, *56*, 9341−9345.
(46) Shuai, Z.; Peng, Q. Excited states structure and processes: Understanding organic light-emitting diodes at the molecular level. *Phys. Rep.* **2014**, *537*, 123−156.
(47) Walsh, A.; Buckeridge, J.; Catlow, C. R. A.; Jackson, A. J.; Keal, T. W.; Miskufova, M.; Sherwood, P.; Shevlin, S. A.; Watkins, M. B.; Woodley, S. M.; Sokol, A. A. Limits to Doping of Wide Band Gap Semiconductors. *Chem. Mater.* **2013**, *25*, 2924−2926.
(48) Buckeridge, J.; Butler, K. T.; Catlow, C. R. A.; Logsdail, A. J.; Scanlon, D. O.; Shevlin, S. A.; Woodley, S. M.; Sokol, A. A.; Walsh, A. Polymorph Engineering of TiO_2: Demonstrating How Absolute Reference Potentials Are Determined by Local Coordination. *Chem. Mater.* **2015**, *27*, 3844−3851.

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

(49) Scanlon, D. O.; Dunnill, C. W.; Buckeridge, J.; Shevlin, S. A.; Logsdail, A. J.; Woodley, S. M.; Catlow, C. R.; Powell, M. J.; Palgrave, R. G.; Parkin, I. P.; Watson, G. W.; Keal, T. W.; Sherwood, P.; Walsh, A.; Sokol, A. A. Band alignment of rutile and anatase TiO_2. *Nat. Mater.* **2013**, *12*, 798−801.
(50) Buckeridge, J.; Catlow, C. R.; Scanlon, D. O.; Keal, T. W.; Sherwood, P.; Miskufova, M.; Walsh, A.; Woodley, S. M.; Sokol, A. A. Determination of the nitrogen vacancy as a shallow compensating center in GaN doped with divalent metals. *Phys. Rev. Lett.* **2015**, *114*, 016405.
(51) Xie, Z.; Sui, Y.; Buckeridge, J.; Catlow, C. R. A.; Keal, T. W.; Sherwood, P.; Walsh, A.; Scanlon, D. O.; Woodley, S. M.; Sokol, A. A. Demonstration of the donor characteristics of Si and O defects in GaN using hybrid QM/MM. *Phys. Status Solidi A* **2017**, *214*, 1600445.
(52) Xie, Z.; Sui, Y.; Buckeridge, J.; Sokol, A. A.; Keal, T. W.; Walsh, A. Prediction of multiband luminescence due to the gallium vacancy−oxygen defect complex in GaN. *Appl. Phys. Lett.* **2018**, *112*, 262104.
(53) Buckeridge, J.; Catlow, C. R. A.; Farrow, M. R.; Logsdail, A. J.; Scanlon, D. O.; Keal, T. W.; Sherwood, P.; Woodley, S. M.; Sokol, A. A.; Walsh, A. Deep vs shallow nature of oxygen vacancies and consequent n-type carrier concentrations in transparent conducting oxides. *Phys. Rev. Mater.* **2018**, *2*, 054604.
(54) Van Speybroeck, V.; Hemelsoet, K.; Joos, L.; Waroquier, M.; Bell, R. G.; Catlow, C. R. Advances in theory and their application within the field of zeolite chemistry. *Chem. Soc. Rev.* **2015**, *44*, 7044−7111.
(55) O'Malley, A. J.; Logsdail, A. J.; Sokol, A. A.; Catlow, C. R. Modelling metal centres, acid sites and reaction mechanisms in microporous catalysts. *Faraday Discuss.* **2016**, *188*, 235−255.
(56) Buckeridge, J.; Bromley, S. T.; Walsh, A.; Woodley, S. M.; Catlow, C. R.; Sokol, A. A. One-dimensional embedded cluster approach to modeling CdS nanowires. *J. Chem. Phys.* **2013**, *139*, 124101.
(57) Downing, C. A.; Sokol, A. A.; Catlow, C. R. The reactivity of CO_2 on the MgO(100) surface. *Phys. Chem. Chem. Phys.* **2014**, *16*, 184−195.
(58) Logsdail, A. J.; Downing, C. A.; Keal, T. W.; Sherwood, P.; Sokol, A. A.; Catlow, C. R. Modelling the chemistry of Mn-doped MgO for bulk and (100) surfaces. *Phys. Chem. Chem. Phys.* **2016**, *18*, 28648−28660.
(59) Downing, C. A.; Sokol, A. A.; Catlow, C. R. The reactivity of CO_2 and H_2 at trapped electron sites at an oxide surface. *Phys. Chem. Chem. Phys.* **2014**, *16*, 21153−21156.
(60) Berger, D.; Logsdail, A. J.; Oberhofer, H.; Farrow, M. R.; Catlow, C. R.; Sherwood, P.; Sokol, A. A.; Blum, V.; Reuter, K. Embedded-cluster calculations in a numeric atomic orbital density-functional theory framework. *J. Chem. Phys.* **2014**, *141*, 024105.
(61) Meisner, J.; Lamberts, T.; Kästner, J. Atom Tunneling in the Water Formation Reaction $H_2 + OH \rightarrow H_2O + H$ on an Ice Surface. *ACS Earth Space Chem.* **2017**, *1*, 399−410.
(62) Lamberts, T.; Kästner, J. Influence of Surface and Bulk Water Ice on the Reactivity of a Water-forming Reaction. *Astrophys. J.* **2017**, *846*, 43.
(63) Song, L.; Kästner, J. Tunneling Rate Constants for H_2CO+H on Amorphous Solid Water Surfaces. *Astrophys. J.* **2017**, *850*, 118.
(64) Cormanich, R. A.; Durie, A.; Bjornsson, R.; Rittner, R.; O'Hagan, D.; Bühl, M. Density Functional Study of Interactions between Fluorinated Cyclohexanes and Arenes. *Helv. Chim. Acta* **2014**, *97*, 797−807.
(65) Doitomi, K.; Xu, K.; Hirao, H. The mechanism of an asymmetric ring-opening reaction of epoxide with amine catalyzed by a metal-organic framework: insights from combined quantum mechanics and molecular mechanics calculations. *Dalton Trans* **2017**, *46*, 3470−3481.
(66) Zheng, X.; Peng, Q.; Zhu, L.; Xie, Y.; Huang, X.; Shuai, Z. Unraveling the aggregation effect on amorphous phase AIE luminogens: a computational study. *Nanoscale* **2016**, *8*, 15173−15180.
(67) Li, W.; Peng, Q.; Ma, H.; Wen, J.; Ma, J.; Peteanu, L. A.; Shuai, Z. Theoretical Investigations on the Roles of Intramolecular Structure Distortion versus Irregular Intermolecular Packing in Optical Spectra of 6T Nanoparticles. *Chem. Mater.* **2017**, *29*, 2513−2520.
(68) Catlow, C. R. A.; Buckeridge, J.; Farrow, M. R.; Logsdail, A. J.; Sokol, A. A. Quantum Mechanical/Molecular Mechanical (QM/MM) Approaches. In *Handbook of Solid State Chemistry*, Dronskowski, R., Kikkawa, S., Stein, A., Eds.; Wiley-VCH: Weinheim, Germany, 2017; Vol. *5*, pp 647−680.
(69) Tcl Developer Xchange. https://www.tcl.tk (accessed Oct 15, 2018).
(70) The Python programming language. https://www.python.org/ (accessed Oct 15, 2018).
(71) Oliphant, T. E. *A guide to NumPy*; Trelgol Publishing, 2006.
(72) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Van Dam, H. J. J.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L.; de Jong, W. A. NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations. *Comput. Phys. Commun.* **2010**, *181*, 1477−1489.
(73) Wesolowski, T. A.; Warshel, A. Frozen density functional approach for ab initio calculations of solvated molecules. *J. Phys. Chem.* **1993**, *97*, 8050−8053.
(74) Wesolowski, T. A.; Shedge, S.; Zhou, X. Frozen-Density Embedding Strategy for Multilevel Simulations of Electronic Structure. *Chem. Rev.* **2015**, *115*, 5891−5928.
(75) Guest, M. F.; Bush, I. J.; Van Dam, H. J. J.; Sherwood, P.; Thomas, J. M. H.; Van Lenthe, J. H.; Havenith, R. W. A.; Kendrick, J. The GAMESS-UK electronic structure package: algorithms, developments and applications. *Mol. Phys.* **2005**, *103*, 719−747.
(76) Aidas, K.; Angeli, C.; Bak, K. L.; Bakken, V.; Bast, R.; Boman, L.; Christiansen, O.; Cimiraglia, R.; Coriani, S.; Dahle, P.; Dalskov, E. K.; Ekström, U.; Enevoldsen, T.; Eriksen, J. J.; Ettenhuber, P.; Fernández, B.; Ferrighi, L.; Fliegl, H.; Frediani, L.; Hald, K.; Halkier, A.; Hättig, C.; Heiberg, H.; Helgaker, T.; Hennum, A. C.; Hettema, H.; Hjertenæs, E.; Høst, S.; Høyvik, I. M.; Iozzi, M. F.; Jansik, B.; Jensen, H. J. A.; Jonsson, D.; Jørgensen, P.; Kauczor, J.; Kirpekar, S.; Kjærgaard, T.; Klopper, W.; Knecht, S.; Kobayashi, R.; Koch, H.; Kongsted, J.; Krapp, A.; Kristensen, K.; Ligabue, A.; Lutnæs, O. B.; Melo, J. I.; Mikkelsen, K. V.; Myhre, R. H.; Neiss, C.; Nielsen, C. B.; Norman, P.; Olsen, J.; Olsen, J. M. H.; Osted, A.; Packer, M. J.; Pawlowski, F.; Pedersen, T. B.; Provasi, P. F.; Reine, S.; Rinkevicius, Z.; Ruden, T. A.; Ruud, K.; Rybkin, V. V.; Salek, P.; Samson, C. C. M.; Sánchez de Meras, A.; Saue, T.; Sauer, S. P. A.; Schimmelpfennig, B.; Sneskov, K.; Steindal, A. H.; Sylvester-Hvid, K. O.; Taylor, P. R.; Teale, A. M.; Tellgren, E. I.; Tew, D. P.; Thorvaldsen, A. J.; Thøgersen, L.; Vahtras, O.; Watson, M. A.; Wilson, D. J. D.; Ziolkowski, M.; Ågren, H. The Dalton quantum chemistry program system. *WIREs Comput. Mol. Sci.* **2014**, *4*, 269−284.
(77) Reine, S.; Kjærgaard, T.; Helgaker, T.; Vahtras, O.; Rinkevicius, Z.; Frecus, B.; Keal, T. W.; Sunderland, A.; Sherwood, P.; Schliephake, M.; Aguilar, X.; Axner, L.; Iozzi, M. F.; Saastad, O. W.; Gimenez, J. *Petascaling and Performance Analysis of Dalton on Different Platforms*; PRACE White Paper, 2012.
(78) Gale, J. D.; Rohl, A. L. The General Utility Lattice Program (GULP). *Mol. Simul.* **2003**, *29*, 291−341.
(79) Todorov, I. T.; Smith, W.; Trachenko, K.; Dove, M. T. DL_POLY_3: new dimensions in molecular dynamics simulations via massive parallelism. *J. Mater. Chem.* **2006**, *16*, 1911−1918.
(80) Kästner, J.; Carr, J. M.; Keal, T. W.; Thiel, W.; Wander, A.; Sherwood, P. DL-FIND: an open-source geometry optimizer for atomistic simulations. *J. Phys. Chem. A* **2009**, *113*, 11856−11865.
(81) Henkelman, G.; Jónsson, H. A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives. *J. Chem. Phys.* **1999**, *111*, 7010−7022.
(82) Henkelman, G.; Uberuaga, B. P.; Jónsson, H. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *J. Chem. Phys.* **2000**, *113*, 9901−9904.

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328

(83) Henkelman, G.; Jónsson, H. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *J. Chem. Phys.* **2000**, *113*, 9978−9985.
(84) Keal, T. W.; Koslowski, A.; Thiel, W. Comparison of algorithms for conical intersection optimization using semiempirical methods. *Theor. Chem. Acc.* **2007**, *118*, 837−844.
(85) Keal, T. W.; Wanko, M.; Thiel, W. Assessment of semiempirical methods for the photoisomerisation of a protonated Schiff base. *Theor. Chem. Acc.* **2009**, *123*, 145−156.
(86) Rommel, J. B.; Goumans, T. P.; Kästner, J. Locating Instantons in Many Degrees of Freedom. *J. Chem. Theory Comput.* **2011**, *7*, 690−698.
(87) Liu, D. C.; Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **1989**, *45*, 503−528.
(88) Billeter, S. R.; Turner, A. J.; Thiel, W. Linear scaling geometry optimization and transition state search in hybrid delocalised internal coordinates. *Phys. Chem. Chem. Phys.* **2000**, *2*, 2177−2186.
(89) Keal, T. W. *Microiterative QM/MM Optimisation for Materials Chemistry*; HECToR dCSE Report, 2013.
(90) van Dam, H. J. J.; Guest, M. F.; Sherwood, P.; Thomas, J. M. H.; van Lenthe, J. H.; van Lingen, J. N. J.; Bailey, C. L.; Bush, I. J. Large scale electronic structure calculations in the study of the condensed phase. *J. Mol. Struct.: THEOCHEM* **2006**, *771*, 33−41.
(91) Keal, T. W.; Sherwood, P.; Dutta, G.; Sokol, A. A.; Catlow, C. R. A. Characterization of hydrogen dissociation over aluminium-doped zinc oxide using an efficient massively parallel framework for QM/MM calculations. *Proc. R. Soc. London, Ser. A* **2011**, *467*, 1900−1924.
(92) Humphrey, W.; Dalke, A.; Schulten, K. VMD: Visual molecular dynamics. *J. Mol. Graphics* **1996**, *14*, 33−38.
(93) Becke, A. D. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A: At., Mol., Opt. Phys.* **1988**, *38*, 3098−3100.
(94) Lee, C.; Yang, W.; Parr, R. G. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B: Condens. Matter Mater. Phys.* **1988**, *37*, 785−789.
(95) Weigend, F.; Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy. *Phys. Chem. Chem. Phys.* **2005**, *7*, 3297−3305.
(96) Fuentealba, P.; Szentpaly, L. v.; Preuss, H.; Stoll, H. Pseudopotential calculations for alkaline-earth atoms. *J. Phys. B: At. Mol. Phys.* **1985**, *18*, 1287−1296.
(97) Andrae, D.; Huermann, U.; Dolg, M.; Stoll, H.; Preuss, H. Energy-adjusted ab initio pseudopotentials for the second and third row transition elements. *Theor. Chim. Acta* **1990**, *77*, 123−141.
(98) Xia, X.; Oldman, R.; Catlow, R. Computational Modeling Study of Bulk and Surface of Yttria-Stabilized Cubic Zirconia. *Chem. Mater.* **2009**, *21*, 3576−3585.
(99) Smith, W.; Forester, T. R. DL_POLY_2.0: A general-purpose parallel molecular dynamics simulation package. *J. Mol. Graphics* **1996**, *14*, 136−141.
(100) Yong, C. W. Descriptions and Implementations of DL_F Notation: A Natural Chemical Expression System of Atom Types for Molecular Simulations. *J. Chem. Inf. Model.* **2016**, *56*, 1405−1409.
(101) Neese, F. The ORCA program system. *WIREs Comput. Mol. Sci.* **2012**, *2*, 73−78.
(102) Aradi, B.; Hourahine, B.; Frauenheim, T. DFTB+, a sparse matrix-based implementation of the DFTB method. *J. Phys. Chem. A* **2007**, *111*, 5678−5684.
(103) Hutter, J.; Iannuzzi, M.; Schiffmann, F.; VandeVondele, J. CP2K: atomistic simulations of condensed matter systems. *WIREs Comput. Mol. Sci.* **2014**, *4*, 15−25.
(104) Dovesi, R.; Orlando, R.; Erba, A.; Zicovich-Wilson, C. M.; Civalleri, B.; Casassa, S.; Maschio, L.; Ferrabone, M.; De La Pierre, M.; D'Arco, P.; Noel, Y.; Causa, M.; Rerat, M.; Kirtman, B. CRYSTAL14: A Program for the Ab Initio Investigation of Crystalline Solids. *Int. J. Quantum Chem.* **2014**, *114*, 1287−1317.
(105) Youngs, T. G. A. Aten-An Application for the Creation, Editing, and Visualization of Coordinates for Glasses, Liquids, Crystals, and Molecules. *J. Comput. Chem.* **2010**, *31*, 639−648.

DOI: 10.1021/acs.jctc.8b01036
J. Chem. Theory Comput. 2019, 15, 1317−1328