MANUAL

QMMM 2018

Hai Lin,[1] Yan Zhang,[1] Soroosh Pezeshki,[1] Bo Wang,[2] Xin-Ping Wu,[2] Laura Gagliardi,[2] and Donald G. Truhlar[2]

[1]*Chemistry Department, University of Colorado Denver, Denver, CO 80217-3364*

[2]*Department of Chemistry, Chemical Theory Center, and Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN 55455-0431*

Program version: 2018

Program version date: September 19, 2018

Date of release of this version: October 19, 2018

Manual version date: November 13, 2019

Abstract: QMMM is a computer program for performing single-point calculations (energies, gradients, and Hessians), geometry optimizations, and molecular dynamics using combined quantum mechanical and molecular mechanical (QM/MM) methods.

Three types of schemes are available for embedding the QM subsystem in the MM one: mechanical embedding, electronic embedding (also called electrostatic embedding), and polarizable embedding. The boundary between the QM and MM regions that passes through covalent bonds can be treated by a number of schemes, including the redistributed charge (RC) scheme, the redistributed charge and dipole (RCD) scheme, the balanced redistributed charge (BRC) scheme, the balanced redistributed charge and dipole (BRCD) scheme, the polarized-boundary RC (PBRC) scheme, the polarized-boundary RCD (PBRCD) scheme, the flexible-boundary RC (FBRC), and the flexible-boundary RCD (FBRCD) scheme. The RC and RCD schemes are link-atom-based electronic-embedding schemes, where the QM calculations are

carried out in the presence of background atomic partial charges of the MM atoms with with advanced methods to avoid the over-polarization of the QM subsystem by MM charges in the close vicinity. The BRC and BRCD schemes improve the RC and RCD schemes by including a balancing step for the MM charges to conserve the total charge of the QM/MM system. The PBRC and PBRCD schemes account for the polarization of the MM subsystem due to the QM subsystem near the boundary; the polarization of the MM subsystem is realized by adjusting the secondary-subsystem atomic partial charges in the embedded-QM calculations according to the principle of electronegativity equalization and the principle of charge conservation. The FBRC and FBRCD schemes are further developments of the PBRC and PBRCD schemes, and they account for partial charge transfer (in addition to mutual polarization) between the QM and MM subsystems.

Screened charge schemes and smeared charge schemes to delocalize the MM charges can be used in the electronically embedded QM/MM method. In the screened charge schemes, charge penetration effects are taken account of, whereas these are neglected in the traditional point-charge scheme. In the smeared charge scheme, the charges that are close to the QM/MM boundary are smeared to avoid overpolarization.

Molecular dynamics simulations can be performed at the pure-MM level, pure-QM level, fixed-partitioning QM/MM level, and adaptive-partitioning QM/MM level. The adaptive-partitioning treatments permit on-the-fly relocation of the QM/MM boundary by dynamically reclassifying atoms or groups into the QM or MM subsystems.

The dynamics simulations can be performed in a microcanonical or canonical ensemble with or without periodic boundaries. QMMM calls a QM package and an MM package to perform the required single-level calculations. QMMM was tested with GAMESS, *Gaussian*, and ORCA for the QM package and with TINKER for the MM package; it contains 158 sample runs that can be used to learn and test the program.

Short Table of Contents

Table of Contents

Chapter One

1

1. Introduction

QMMM is a computer program for performing single-point calculations (energies, gradients, and/or Hessians), geometry optimizations, and Born-Oppenheimer molecular dynamics using combined quantum mechanics (QM) and molecular mechanics (MM) methods.(*1-102*)

The boundary between QM and MM regions can be treated by a variety of methods, including the redistributed charge (RC) scheme,(*82*) the redistributed charge and dipole (RCD) scheme,(*82*) the polarized-embedding RC (PBRC) scheme,(*84*) the polarized-embedding RCD (PBRCD) scheme,(*84*) the flexible-boundary RC (FBRC),(*87*) and the flexible-boundary RCD (FBRCD) scheme,(*87*) the adaptive-partitioning RC (APRC), and the adaptive-partitioning RCD (APRCD) scheme. All schemes use link atoms (also called cap atoms) to saturate the dangling bonds for the QM subsystem and use redistributed MM point charges to mimic a generalized hybrid orbital (GHO)(*39, 41, 45, 46, 103*) on the MM host atom (called the M1 atom) that is replaced by the link atom.

In the RCD treatment, the value of the redistributed charge and the value of the charge on the M2 atom, i.e., the MM atom that is directly bonded to the M1 atom, are further modified to preserve the M1–M2 bond dipole. Both RC and RCD schemes can be viewed as classical mechanical analogs to the quantum mechanical description provided by the GHO method. The balanced RC and balanced RCD schemes are modified versions of RC and RCD in which a balancing step is added to conserve the total charge of the QM/MM system. (*159, 160*)

The PBRC and PBRCD schemes are further developments of the RC and RCD schemes. The PBRC and PBRCD schemes account for the polarization of the MM subsystem due to the QM subsystem; the polarization of the MM subsystem is realized by adjusting the background point charges in the embedded-QM calculations for the QM subsystem according to the principle of electronegativity equalization and the principle of charge conservation.

In the FBRC and FBRCD schemes, both the partial charge transfer and mutual polarization between the QM and MM subsystems are accounted for based on the principle of electronic

chemical potential equalization. The QM subsystem is viewed as an open system with a fluctuating number of electrons and is described by a statistical mixture of ensembles that consist of states with an integral number of electrons. The MM subsystem serves a reservoir that exchanges electrons with the QM subsystem. The electronic chemical potential of the MM subsystem varies when charges flow in or out until equilibrium is established for electronic chemical potentials between the QM and MM subsystems.

In the APRC and APRCD schemes, the atoms/groups (including fragmental groups of molecules) can be on-the-fly reclassified into QM or MM subsystems during dynamics simulations.

Some other schemes used for QM/MM boundary treatments are also implemented; in particular, we implement a mechanical embedding scheme (ME)(*25*) that is equivalent to the original integrated molecular-orbital molecular-mechanics (IMOMM),(*11*) the straight electronic embedding (SEE) scheme, three eliminated charge schemes,(*2*) and the shifted charge scheme (Shift).(*70*) (Please note that electronic embedding is also called electrostatic embedding.) The RC, RCD, SEE, eliminated charges, and shift schemes are examples of electronic embedding. The PBRC and PBRCD schemes belong to the so-called polarized-embedding category. The FBRC and FBRCD belong to a new type of charge-transferable-embedding schemes.

In addition to the point charge models, QMMM can also use delocalized charge models in QM/MM calculations; these include a screened charge model (161) and a smeared charge model. (160) In the screened charge model, charge penetration effects are included. In the smeared charge model, the charges that are close to the boundary are smeared to avoid the overpolarization of the QM region.

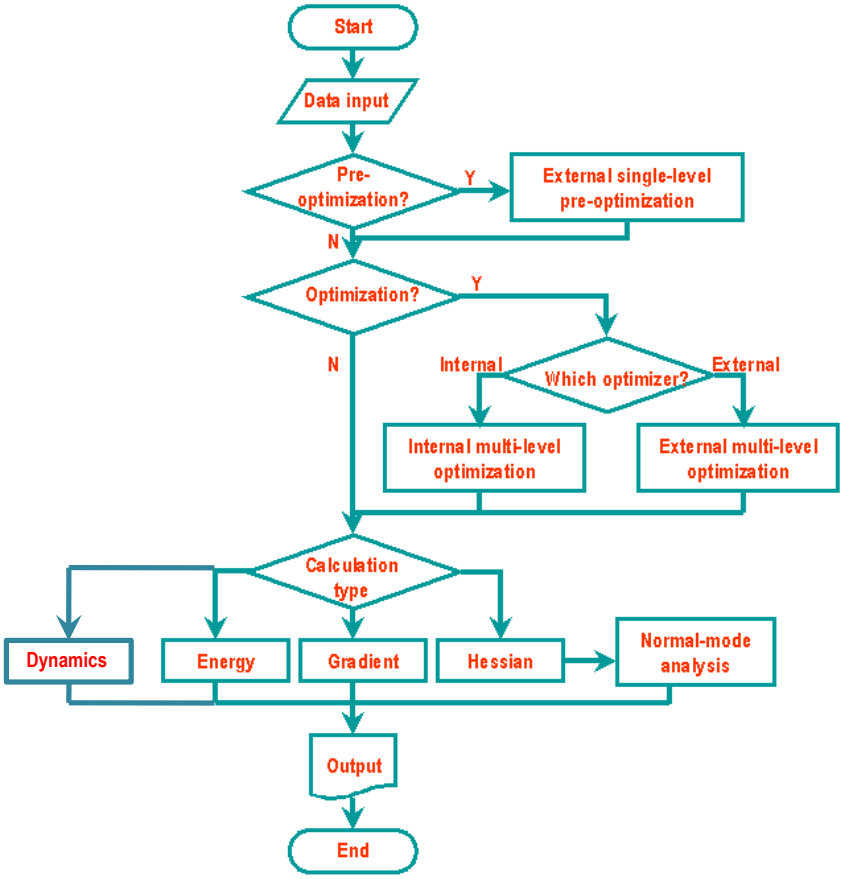
As implied by its name, the QMMM program is desgned to involve a combination of quantum mechanical electronic structure theory and molecular mechanics. The QM levels may be semi-empirical molecular orbital theory,(*104*) Hartree-Fock or correlated wave function theory,(*105*) or density functional theory.(*106*) Any MM force field that uses atom-centered point charges for electrostatic interactions can be used for electronic embedding; some popular force fields that belong to this catalog are AMBER,(*107*) CHARMM,(*108*) and OPLS-AA.(*109*) For mechanical embedding there is no limitation at all on the force field; methods like MM3(*110-112*) that use distributed dipoles may also be used. Each level of calculation is obtained by making a call to an external electronic structure or molecular mechanics package. The external programs enabled

in the current version are GAMESS,(*113*) *Gaussian*(*114*) (the 03 version and later), and ORCA(*115*) for the QM calculations and TINKER(*116*) (version 4.1 and later) for the MM calculations. The structure of the code is modular so that one could also use other packages with straightforward modifications.

Seven kinds of calculations can be done with QMMM: energies, gradients, Hessians, optimizations using QMMM routines, optimizations using the external electronic structure or molecular mechanics packages, and Born-Oppenheimer molecular dynamics. The various kinds of calculation are described as follows:

1. *External single-level pre-optimization.* This option performs geometry optimization at a single level (QM or MM) from the external electronic structure or molecular mechanics package. The use of this option is recommended to give a good starting geometry.
2. *Internal multi-level optimization.* This option allows geometry optimization to be done at the QM/MM level by one of the internal (QMMM) optimizers.
3. *External multi-level optimization.* This option allows geometry optimization to be carried out at the QM/MM level by an external optimizer (by default the Berny optimizer of *Gaussian*).
4. *Hessian calculation.* When this option is specified, the program will first calculate the energy, gradient, and Hessian, and then it will compute frequencies and normal mode coordinates.
5. *Gradient calculation.* When this option is specified, the program will calculate the energy and gradient.
6. *Energy calculation.* When this option is specified, the program will calculate the energy.
7. *Molecular dynamics*. When this option is specified, the program will perform *Born*-Oppenheimer molecular dynamics simulations.

The organization of QMMM is summarized in the flowchart on the next page:



The QMMM program has been developed primarily for QM/MM calculations; but it can also perform pure-QM or pure-MM calculations. However, although users can carry out pure-QM calculations through the QMMM interface to invoke an electronic-structure package such as *Gaussian*, GAMESS, and ORCA for doing the actual calculations, we do not recommend users to do so for most purposes. Instead, it will typically be more convenient and more efficient to use the electronic structure packages directly for pure-QM calculations since use of QMMM interface will involve unnecessary data transfer and file handling. For the same reason, we do not encourage users to carry out pure-MM calculations through the QMMM interface to invoke TINKER packages.

The functionality of the QMMM program is summarized in Table 1.1.

Table 1.1. The functionality of the QMMM program.[a]

	QM/MM	MM	QM
Energy	Y	Y	Y
Gradient	Y	Y	Y
Hessian and normal-mode analysis	Y	Y	Y
Geometry pre-optimization	N	Y	Y
Geometry optimization by an internal (QMMM) optimizer	Y	N	N
Geometry optimization by an external (*Gaussian*) optimizer	Y	N	N
Dynamics (Molecular Dynamics)	Y	Y	Y

[a] Y = yes (present); N = no (not present).

QMMM is run from one main input file, called `ml.inp`, one coordinate file, called `t41.crd`, and one MM parameter file, called `t41.prm`; and one output file, called `ml.out`, is created. Chapter 6 describes the contents of the input files and describes the files necessary to run QMMM. The files created by QMMM are described in Chapter 7. Chapter 8 provides installation details and usage instructions. Listed in Chapter 9 are supported platforms (computers and operation systems) where QMMM has been tested.

Users are encouraged to send their questions/comments to the authors:

Hai Lin: hai.lin@ucdenver.edu

Donald G. Truhlar: truhlar@chem.umn.edu

Questions about licenses, distribution, and so forth may be sent to Truhlar Group License Manager (software@comp.chem.umn.edu).

Publications resulting from using the QMMM package should cite the program. The main recommended citation is:

H. Lin, Y. Zhang, S. Pezeshki, B. Wang, X.-P. Wu, L. Gagliardi, and D. G. Truhlar, QMMM 2018 (University of Minnesota, Minneapolis, 2018). http://comp.chem.umn.edu/qmmm

See chapter 2 for complete recommended citations.

No guarantee is made that this program is bug-free or suitable for specific applications, and no liability is accepted for any limitations in the mathematical methods and algorithms used within the program. We try our best to provide helps to users, when possible and when time permits; however, no consulting or maintenance services are guaranteed or implied. The program is available free of charge, but should not be redistributed outside of a esaerch group. New groups should obtain the code from the download site whose URL is given in the recommended coiation above.

Chapter Two

2

2. References for QMMM Program

The recommended reference for the current version of the code is given below in two styles, first in *J. Chem. Phys.* style, then in *J. Amer. Chem. Soc.* style. We give the references in four different styles for those who wish to cut and paste.

J. Chem. Phys. style if QMMM is used with GAMESS:

QMMM 2018 by H. Lin, Y. Zhang, S. Pezeshki, B. Wang, X.-P. Wu, L. Gagliardi, and D. G. Truhlar , University of Minnesota, Minneapolis, 2018, based on GAMESS by M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. J. Su, T. L. Windus, M. Dupuis, and J. A. Montgomery, GAMESS, Iowa State University, Ames, Iowa, 2006, based on TINKER 4.2 by J. W. Ponder, TINKER–version 4.2, Washington University, St. Louis, MO, 2004, and based on MULTILEVEL-version 3.1/G03 by J. M. Rodgers, B. J. Lynch, P. L. Fast, Y. Zhao, J. Pu, Y.-Y. Chuang, and D. G. Truhlar, University of Minnesota, Minneapolis, 2002. http://comp.chem.umn.edu/qmmm

J. Amer. Chem. Soc. style if QMMM is used with GAMESS:

Lin, H.; Zhang, Y.; Pezeshki S.; Wang, B.; Wu, X.-P.; Gagliardi, L.; Truhlar, D. G.; QMMM 2018; University of Minnesota, Minneapolis, 2018, based on GAMESS by Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S. J.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. GAMESS, Iowa State University, Ames, Iowa, 2006, based on TINKER 4.2 by Ponder, J. W. TINKER–version 4.2, Washington University, St. Louis, MO, 2004, and based on Rodgers, J. M.; Lynch, B. J.; Fast, P. L.; Chuang, Y.-Y.; Pu, J.; Zhao, Y.; Truhlar, D. G.; MULTILEVEL-version 3.1/G03; University of Minnesota, Minneapolis, 2002.

J. Chem. Phys. style (short version) if QMMM is used with *Gaussian*:

QMMM 2018 by H. Lin, Y. Zhang, S. Pezeshki, B. Wang, X.-P. Wu, L. Gagliardi, and D. G. Truhlar, University of Minnesota, Minneapolis, 2018, based on *Gaussian*, by M. J. Frisch, et al., Gaussian, Inc., Pittsburgh PA, 2003, based on TINKER 6.3 by J. W. Ponder, TINKER–version 6.3, Washington University, St. Louis, MO, 2014, and based on MULTILEVEL-version 3.1/G03 by J. M. Rodgers, B. J. Lynch, P. L. Fast, Y. Zhao, J. Pu, Y. -Y. Chuang, and D. G. Truhlar, University of Minnesota, Minneapolis, 2002. http://comp.chem.umn.edu/qmmm

J. Amer. Chem. Soc. style (short version for printed article) if QMMM is used with *Gaussian*:

Lin, H.; Zhang, Y.; Pezeshki, S.; Wang, B.; Wu, X.-P.; Gagliardi, L.; Truhlar, D. G.; QMMM 2018; University of Minnesota, Minneapolis, 2018, based on Frisch, M. J. et al. *Gaussian*; Gaussian, Inc.; Pittsburgh PA, 2003, based on TINKER 6.3 by Ponder, J. W. TINKER–version 6.3, Washington University, St. Louis, MO, 2014, and based on Rodgers, J. M.; Lynch, B. J.; Fast, P. L.; Chuang, Y.-Y.; Pu, J.; Zhao, Y.; Truhlar, D. G.; MULTILEVEL-version 3.1/G03; University of Minnesota, Minneapolis, 2002. http://comp.chem.umn.edu/qmmm

J. Chem. Phys. style (long version) if QMMM is used with *Gaussian*:

QMMM 2018 by H. Lin, Y. Zhang, S. Pezeshki, B. Wang, X.-P. Wu, L. Gagliardi, and D. G. Truhlar , University of Minnesota, Minneapolis, 2018, based on *Gaussian*, by M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson,

W. Chen, M. W. Wong, C. Gonzalez, and J. A. Pople, Gaussian, Inc., Pittsburgh PA, 2003, based on TINKER 6.3 by J. W. Ponder, TINKER–version 6.3, Washington University, St. Louis, MO, 2014, and based on MULTILEVEL-version 3.1/G03 by J. M. Rodgers, B. J. Lynch, P. L. Fast, Y. Zhao, J. Pu,Y. -Y. Chuang, and D. G. Truhlar, University of Minnesota, Minneapolis, 2002. http://comp.chem.umn.edu/qmmm

J. Amer. Chem. Soc. style (long version for supporting information) if QMMM is used with *Gaussian*:

Lin, H.; Zhang, Y.; Pezeshki, S.; Wang, B.; Wu, X.-P.; Gagliardi, L.; Truhlar, D. G.; Pezeshki, S.; QMMM 2018; University of Minnesota, Minneapolis, 2018, based on Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Montgomery, Jr., J. A.; Vreven, T.; Kudin, K. N.; Burant, J. C.; Millam, J. M.; Iyengar, S. S.; Tomasi, J.; Barone, V.; Mennucci, B.; Cossi, M.; Scalmani, G.; Rega, N.; Petersson, G. A.; Nakatsuji, H.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Klene, M.; Li, X.; Knox, J. E.; Hratchian, H. P.; Cross, J. B.; Adamo, C.; Jaramillo, J.; Gomperts, R.; Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Ayala, P. Y.; Morokuma, K.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Zakrzewski, V. G.; Dapprich, S.; Daniels, A. D.; Strain, M. C.; Farkas, O.; Malick, D. K.; Rabuck, A. D.; Raghavachari, K.; Foresman, J. B.; Ortiz, J. V.; Cui, Q.; Baboul, A. G.; Clifford, S.; Cioslowski, J.; Stefanov, B. B.; Liu, G.; Liashenko, A.; Piskorz, P.; Komaromi, I.; Martin, R. L.; Fox, D. J.; Keith, T.; Al-Laham, M. A.; Peng, C. Y.; Nanayakkara, A.; Challacombe, M.; Gill, P. M. W.; Johnson, B.; Chen, W.; Wong, M. W.; Gonzalez, C.; Pople. J. A. *Gaussian*; Gaussian, Inc.; Pittsburgh PA, 2003, based on TINKER 6.3 by Ponder, J. W. TINKER–version 6.3, Washington University, St. Louis, MO, 2014, and based on Rodgers, J. M.; Lynch, B. J.; Fast, P. L.; Chuang, Y.-Y.; Pu, J.; Zhao, Y.; Truhlar, D. G. MULTILEVEL-version 3.1/G03; University of Minnesota, Minneapolis, 2002. http://comp.chem.umn.edu/qmmm

J. Chem. Phys. style if QMMM is used with ORCA:

QMMM 2018 by H. Lin, Y. Zhang, S. Pezeshki, B. Wang, X.-P. Wu, L. Gagliardi, and D. G. Truhlar , University of Minnesota, Minneapolis, 2018, based on ORCA version-2.5 by F. Neese, University of Bonn, Bonn, 2006, based on TINKER 4.2 by J. W. Ponder, TINKER–version 4.2,

Washington University, St. Louis, MO, 2004, and based on MULTILEVEL-version 3.1/G03 by J. M. Rodgers, B. J. Lynch, P. L. Fast, Y. Zhao, J. Pu,Y. -Y. Chuang, and D. G. Truhlar, University of Minnesota, Minneapolis, 2002. http://comp.chem.umn.edu/qmmm

J. Amer. Chem. Soc. style if QMMM is used with ORCA:

Lin, H.; Zhang, Y.; Pezeshki, S.; Wang, B.; Wu, X.-P.; Gagliardi, L.; Truhlar, D. G.; QMMM 2018; University of Minnesota, Minneapolis, 2018, based on ORCA version-2.4 by Neese, F. University of Bonn, Bonn, based on TINKER 4.2 by Ponder, J. W. TINKER–version 4.2, Washington University, St. Louis, MO, 2004, and based on Rodgers, J. M.; Lynch, B. J.; Fast, P. L.; Chuang, Y.-Y.; Pu, J.; Zhao, Y.; Truhlar, D. G.; MULTILEVEL-version 3.1/G03; University of Minnesota, Minneapolis, 2002. http://comp.chem.umn.edu/qmmm

Additional references are given in the documentation for GAMESS, *Gaussian*, ORCA, and TINKER.

Furthermore, users should give references for methods used as well as for software. The reference for the RC and RCD schemes is

Lin, H.; Truhlar, D. G. *J. Phys. Chem. B* **2005**, *109*, 3991-4004.

The reference for the PBRC and PBRCD schemes is

Zhang, Y; Lin, H.; Truhlar, D. G. *J. Chem. Theory Comput.* **2007,** *3*, 1378-1398.

The reference for the FBRC and FBRCD schemes is

Zhang, Y; Lin, H. *J. Chem. Theory Comput.* **2008,** *4*, 414-425.

Zhang, Y; Lin, H. *Theor. Chem. Acc.* **2010**, 216, 315-322**.**

The reference for the APRC and APRCD schemes is

Heyden, A.; Lin, H.; Truhlar, D. G. *J. Phys. Chem. B.* **2007,** *111*, 2231-2241.

Soroosh, P.; Lin, H. *J. Chem. Theory Comput.* **2011**, *7*, 3625-3634**.**

The references for the BRC, BRCD, BRC2, TBRC, and TBRCD schemes are

Wang, B.; Truhlar, D. G., *J. Chem. Theory Comput.*, **2013,** *9*, 1036–1042.

Wang, B.; Truhlar, D. G., *Phys. Chem. Chem. Phys.*, **2011**, *13*, 10556–10564.

Wang, B.; Truhlar, D. G., *J. Chem. Theory Comput.*, **2010**, *6*, 359–369.

The reference for the screened charge schemes is

Wang, B.; Truhlar, D. G., *J. Chem. Theory Comput.*, **2010**, *6*, 3330–3342.

The reference for the smeared redistributed charge schemes is

Wang, B.; Truhlar, D. G., *Phys. Chem. Chem. Phys.*, **2011**, *13*, 10556–10564.

These articles and the present manual also contain references for other QM, MM, and QM/MM methods used in the QMMM package.

Chapter Three

3

3. General Program Description

QMMM is developed in a similar way to MULTILEVEL 3.1,(*117*) which is written in standard Fortran 90. The QMMM program is written using modular subprograms called "hooks." In particular, the structure of hooks in QMMM is very similar to those in MULTILEVEL(*117*) and POLYRATE,(*118*) i.e., there are four main hooks for QMMM values: energy hook (MLEHOOK), gradient hook (MLGHOOK), Hessian hook (MLHHOOK), and optimization hook (MLOHOOK); such an arrangement facilitates revision and modification of the code in the future.

The MLEHOOK, MLGHOOK, and MLHHOOK routines call the QM/MM hooks, i.e., QMMMEHK, QMMMGHK, and QMMMHHK for QM/MM energy, gradient, and Hessian calculations. These QM/MM hooks have been designed in order to allow QMMM calculations with a variety of the QM/MM boundary treatments that are available. They make a minimum number of calls to the electronic structure and molecular mechanics package in order to carry out these calculations. Each of these three QM/MM hooks makes calls to certain "subhooks" for specific calculations.

The optimization hook MLOHOOK is structured differently in that it makes calls to routines for the different optimization algorithms available in QMMM. When energies, gradients, and/or Hessians are required for the optimization algorithm, three different hooks may be called for this information: OPTEHK, OPTGHK, and OPTHHK respectively. Then these three hooks will in turn call the QM/MM hooks, i.e., QMMMEHK, QMMMGHK, and QMMMHHK for QM/MM energy, gradient, and Hessian calculations.

Whenever the program directly calls MLHOOK, but not when it calls MLHOOK from OPTHHK, the program also calculates the harmonic vibrational frequencies and normal mode coordinates and writes them to the output file. This calculation is carried out in mass-scaled Cartesian coordinates, and the translations and rotations are projected out before diagonalizing the mass-scaled force constant matrix.

As the QM/MM calculations require calls to an electronic structure package, the hooks have been written to make calls to the appropriate electronic structure program and then extract the

energies (PROGEHK and PROGESTHK), gradients (PROGGHK and PROGGSTHK), and Hessians (PROGHHK and PROGHSTHK) from the output of that program.

The QM/MM calculations also require multiple calls to a molecular mechanics program (sometimes called a force field program), and the hooks have been written to make calls to the appropriate molecular mechanics program and then extract the energies (PROGMMEHK, PROGMM0EHK, and PROGMMSTEHK), gradients (PROGMMGHK, PROGMM0GHK, and PROGMMSTGHK), and Hessians (PROGMMHHK, PROGMM0HHK, and PROGMMSTHHK) from the output of that program.

Besides these hooks, there are also two hooks to do geometry optimizations (PROGOHK and PROGMMOHK) using the electronic structure methods and molecular mechanics methods of the external programs, respectively, so as to provide a starting geometry for subsequent calls to ML hooks. The QMMM code currently has program hooks for calling GAMESS, *Gaussian*, ORCA, and TINKER, but in the future it is planned that it will contain hooks that will call other electronic structure packages, for example, ACESII, and/or HONDOPLUS.

The hooks are called in the following order to create the functionality described in the introduction: PROGOHK, MLOHOOK, MLHHOOK, MLGHOOK, MLEHOOK.

See Chapter 7 for a complete listing of all the files that comprise the source code of QMMM, as well as for a description of the subprograms and modules within the source code.

Chapter Four

4

4. Theoretical Background

This chapter describes the QM/MM methods that are available in this version of the QMMM code.

4.A. The QM/MM Model

The combined quantum mechanical and molecular mechanical (QM/MM) method is a powerful tool for studying many chemical and biochemical processes such as enzyme reactions. A QM/MM model treats a relatively localized region (e.g., where bond breaking/forming or electronic excitation occur) with a QM method and includes the influence of the surroundings at the MM level. The localized region is called the primary subsystem (PS) due to the higher level employed for treating its electronic structure; this is the region whose orbitals are most likely to change significantly during a reaction. The surroundings are called the secondary subsystem (SS), and they affect the electronic structure of the primary subsystem by imposing geometric restraints and/or through polarization. The advantage of using QM/MM methods is that they include the effects of the SS while remaining computationally feasible. Combined QM/MM methods are a special case of the most general approach called multilevel methods, where two or more levels of theory are employed in treating a system.

The treatment for the boundary between the QM and MM subsystem is a critical issue in QM/MM methodology. In some cases, such as the treatment of solvation and some clusters and nano-structured materials, the boundary between the QM and MM subsystems is between a solute and an environment to which it is not bonded (e.g., solvent molecules), and no covalent bond is cut by the boundary. In many other cases, however, passing the boundary through covalent bonds is desirable, and special care is required to handle the boundary. An example occurs in the treating reactive system involving biopolymer, for example, enzyme catalysis.

Treatments of the boundary between QM and MM regions can be largely grouped into two classes. The first is called the link atom approach, where a “link atom” or “cap atom” is used to saturate the dangling bond at the “frontier atom” of the QM fragment. This link atom is usually

taken to be a hydrogen atom,(*3, 11, 12, 14, 15, 17, 18*) or a parameterized atom, e.g., a one-free-valence atom in the "connection atom",(*27*) "pesudobond",(*62*) and "quantum capping potential"(*63*) schemes, which involve a parameterized semi-empirical Hamiltonian(*27*) or a parameterized effective core potential (ECP) (*62, 63*) adjusted to mimic the properties of the original bond being cut. The link atom method is straightforward and is widely used. However, the introduction of link atoms that are not present in the original molecular system makes the definition of the QM/MM energy more complicated. In addition, it is found, at least in the original versions of the link atom method, that polarization of the bond between the QM frontier atom and the link atom is unphysical due to the nearby point charge on the MM "boundary atom" (an MM boundary atom is the atom whose bond to a frontier QM atom is cut). In early work, the point charges on the MM boundary atoms and on some of the atoms directly bonded to it were deleted, and in second-generation link-atom methods, these point charges are treated in a special manner(*25, 27, 49, 62, 70*) to avoid this unphysical polarization, for example, some (or all) of the point charges might be redistributed, scaled, or zeroed. Extensive discussions of these problems can be found in literature. (*25, 49, 66*)

The second class of QM/MM methods consists of methods that use localized orbitals instead of link atoms at the boundary of the QM and MM regions. Examples include the so-called local self-consistent field (LSCF) algorithm(*6-10*) and the generalized hybrid orbital (GHO) method.(*39, 41, 45, 46, 103*) In the GHO approach, a set of four sp^3 hybrid orbitals is assigned to each MM boundary atom. The hybridization scheme is determined by the local geometry of the three MM atoms to which the boundary atom is bonded, and the parametrization is assumed to be transferable. The hybrid orbital that is directed toward the frontier QM atom is called the active orbital, and the other three hybrid orbitals are called auxiliary orbitals. All four hybrid orbitals are included in the QM calculations, but the active hybrid orbital participates in the SCF optimizations, while the auxiliary orbitals do not.

The methods using local orbitals are theoretically more fundamental than the methods using link atoms, since they provide a quantum mechanical description for the charge distribution around the QM/MM boundary. The delocalized representation of charges in these orbitals helps to prevent or reduce the over-polarization that, as mentioned above, is sometimes found in the link-atom methods. However, the local-orbital methods are much more complicated than the

link-atom methods. Test calculations showed that reasonably good accuracy can be achieved by both approaches if they are used with special care.(*10, 30, 45, 49, 66*)

It is natural to ask if there is a way to combine some of the merits of the link-atom and local-orbital approaches. Such a combination would be attractive if it retains the simplicity of the former and the theoretical justification of the latter. Recently, we(*82*) explored simple ways to incorporate delocalization into the link-atom picture. In particular, we introduced a redistributed charge (RC) scheme and a redistributed charge and dipole (RCD) scheme, each of which can be viewed in one sense as a point charge analog to the GHO method. Both schemes use redistributed charges as classical mimics for the auxiliary orbitals associated with the MM boundary atom in the GHO method. The methods may also be considered as an attempt to further refine the procedure introduced earlier by de Vries and coworkers.(*70*)

Before proceeding to the next section, we remind users that there are intrinsic limitations to the QM/MM approach despite its success in many applications. The QM/MM interface is immune to artifacts and therefore the users should test and calibrate the QM/MM models for his or her kind of applications, if that has not already been done.

4.B. General Description of the RC and RCD Schemes

The redistributed charge scheme(*82*) can be considered as a point charge analog of the GHO method. This is illustrated in Fig. 1(a). In the GHO scheme,(*39, 41*) the valence electron distribution on The atom B (which is always a carbon atom) has a formal MM partial charge q_B consisting of an effective nuclear charge of 4 and electronic charges in valence orbitals that sum to $-4 + q_B$. To accomplish this charge distribution, the GHO method makes arrangement as follows: First, it assigns an occupation of $n_{aux} = 1 - q_B/3$ to each auxiliary orbital, which therefore carries a charge of $q_{aux} = -n_{aux}$ (a typical value of q_B is –0.18, for which $q_{aux} = -1.06$). Next, the remaining single electronic charge of B is added to the active electron pool treated by the SCF process and is primarily in the active orbital but also possibly delocalized over the quantum system. The boundary atom B is represented by the active orbital η_Q pointing toward the QM frontier atom A and by three auxiliary orbitals (η_X, η_Y, and η_Z) pointing toward the MM atoms (X, Y, and Z) directly bonded to B. These four GHO orbitals are constructed by hybridization of the atomic *s* and *p* valence basis functions of B.

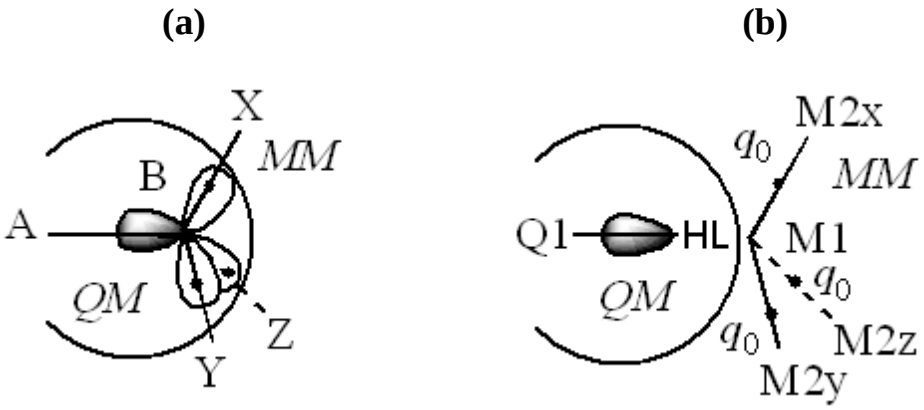


Figure 1. Schematic representations of the QM/MM boundary treatments in (a) the generalized hybrid orbital (GHO) scheme and (b) the redistributed charge (RC) scheme. The frontier atom is denoted as A in (a) and as Q1 in (b). The boundary atom is denoted as B in (a) and as M1 in (b). The MM atoms bounded to the boundary atoms are denoted as X, Y, and Z in (a) and as M2x, M2y, and M2z in (b). The link atom is denoted as HL. The redistributed charge is denoted as q_0.

The redistributed charge (RC) scheme is illustrated in Fig. 1(b). We will find it convenient to label the atoms according to "tiers" (or "layers"). Thus the B atom of GHO, i.e., the MM boundary atom, will be denoted as M1 in this paper, and the partial charge q_B becomes q_{M1}. Those MM atoms directly bonded to M1 will be called second-tier molecular mechanics atoms or M2 [e.g., M2x, M2y, and M2z in Fig. 1 (b)]. The QM atom that is directly connected to M1 is still called the frontier atom, but now it is labeled Q1. One continues the numbering in this way: M3 atoms are the third-tier molecular mechanics atoms, i.e., those MM atoms bonded to M2 atoms. Similarly, one defines Q2 and Q3 atoms in the QM subsystem. In the RC treatment, a link atom HL (which denotes hydrogen link) is used to represent the active hybrid orbital η_Q, and the MM partial charge on M1 ($q_{M1} = q_B$) is delocalized evenly as n point charges q_0 with $q_0 = q_{M1}/n$, where n is the number of M1–M2 bonds, usually three. The delocalized point charges q_0 are located on the M1–M2 bonds, as discussed in more detail in the next subsection. This is illustrated in Fig. 1(b) for the case of $n = 3$. These redistributed point charges (q_0) serve as mimics for the auxiliary hybrid orbitals. These are in approximately the same location as the q_{aux} charges of the GHO method, however, because there is no nuclear charge on M1, they are much smaller in magnitude, e.g., $q_0 = -0.06$ when $q_B = -0.18$ and $n = 3$. By construction, the HL atom

does not carry any MM point charge, which is consistent with the requirement that adding the link atom to the QM subsystem should not change the charge for the QM subsystem, e.g., should not make a neutral QM subsystem partially charged.

It should be noted that the redistributed point charges provide only classical mimics for calculating the Columbic interactions of the auxiliary orbitals, and quantum mechanical exchange interactions are not recovered. However, we hope that such a treatment will not cause unacceptably large errors.

4.C. Redistributed Charges and Dipoles

Concerning the details of charge redistribution, the first question to ask is precisely where to locate the redistributed point charges. The location for the redistributed charge q_0 along the M1−M2 bond can be characterized by a scaling factor Cq_0:

$$Cq_0 = R(\text{M1}-q_0) \,/\, R(\text{M1}-\text{M2}) \qquad (4.C.1)$$

where $R(\text{M1}-q_0)$ is the distance between the M1 atom and q_0. The most physical choice for a simple model is to place the redistributed charges at points along each M1−M2 bond, i.e., at the nominal centers of the bond charge distributions. One would expect that the precise position along each M1−M2 bond would depend on the nature of the bond; in practice, though we found that QM/MM calculations such as geometry optimizations are not very sensitive to the actual location, provided the location is sufficiently far from M1. For simplicity, we choose the locations to be the mid-points of the M1−M2 bonds (i.e., $Cq_0 = 0.5$). Schematically, these places are indicated in Fig. 1(b) by dots on the M1−M2 bonds. This is the only redistribution that occurs in the RC scheme.

The second question to ask is how large a perturbation is introduced to the MM subsystem by redistributing charge from M1 to the M1−M2 bond. Clearly, the RC method reduces the contribution of q_0 to each M1−M2 bond dipole q_0R, where R is the M1−M2 bond distance, by a factor of $(1 - Cq_0)$; if $Cq_0 = 0.5$, the contribution reduces by 50%. Therefore, we consider a second method called the RCD method,(*82*) which is the same as the RC method except that the values of redistributed charges q_0 and of the charges on M2 atoms (labeled $k = 1, 2, \ldots$) are further modified such that these contributions to the M1−M2 bond dipoles are preserved:

$$q_0^{\mathrm{RCD}} = q_0/(1 - Cq_0) \tag{4.C.2}$$

$$q_{\mathrm{M2},k}^{\mathrm{RCD}} = q_{\mathrm{M2},k} - q_0 Cq_0 /(1 - Cq_0) \tag{4.C.3}$$

In particular, if $Cq_0 = 0.5$, one has

$$q_0^{\mathrm{RCD}} = 2q_0 \tag{4.C.4}$$

$$q_{\mathrm{M2},k}^{\mathrm{RCD}} = q_{\mathrm{M2},k} - q_0 \tag{4.C.5}$$

This is the only difference between the RC and RCD treatments.

We have also proposed a third method, called the RCD2 method, where the redistributed charge and the charge for M2 atoms are the same as those in the RC scheme:

$$q_0^{\mathrm{RCD2}} = q_0 \tag{4.C.6}$$

$$q_{\mathrm{M2},k}^{\mathrm{RCD2}} = q_{\mathrm{M2},k} \tag{4.C.7}$$

and where the M1–M2 bond dipole is preserved by placing an additional pair of point charges (q_- and q_+) in the vicinity of each M2 atom. The values for these two additional point charges are determined as follows:

$$q_-^{\mathrm{RCD2}} = -q_+^{\mathrm{RCD2}} = q_0 (1 - Cq_0) / Cq_\pm \tag{4.C.8}$$

$$Cq_\pm = R(q_- - q_+) / R(\mathrm{M1–M2}) \tag{4.C.9}$$

where R(M1–M2) is the distance between the M1 and M2 atoms and $R(q_- - q_+)$ is between the (q_- and q_+) point charges. The RCD2 scheme can be considered as a combination of the RC scheme and the shifted charge (Shift) scheme(*70*) that will be discussed in Section 4.G.

The RC and RCD method are published while the RCD2 method is not. The reason for not pubishing the RCD2 scheme is that we suspect that the point charges (q_- and q_+) pair may cause instability in QM wavefunctions, which will present difficulty in vibrational analysis, e.g., produce imaginary frequencies for a minimum-energy structure optimized at the same level. Moreover, more point charges are generated in the RCD2 scheme, and this increases the demand on computational resources without increasing the accuracy enough to warrant the extra expense.

Both the redistributed charge q_0 and the pair of additional point charges q_- and q_+ are called auxiliary charges in the QMMM program.

The RC and RCD schemes are esepcially useful for treating nonpolar bonds at the QM–MM boundary. (Another way to treat nonpolar bonds is to use the flexible-boundary scheme, which accounts for the partial charge transfer between the PS and SS.) For polar bonds, such as C–O bonds, large extra charges may appear on the QM–MM boundary if RC and RCD are used. The balanced-redistributed charge (BRC) scheme was developed to improve the redistributed charge scheme for the treatment of polar bonds at the QM–MM boundary.(*159, 160*) In the balanced RC, the charge on the M1 atom is modified before the redistribution. The modified M1 charge q_0 satisfies

$$q_0 + \sum_i q_i + q^{\mathrm{QM}} = q^{\mathrm{total}} \tag{4.C.10}$$

where $\{q_i\}$ are the MM point charges of MM atoms except M1, q^{QM} is the charge of the QM region (that is, of the capped QM subsystem), and q^{total} is the charge of the original entire system. This charge-balancing step conserves the charge of the QM/MM entire system. After the balancing step, the modified M1 charge, rather than the original M1 charge, is used for the redistribution. In order to better deal with the boundary, the balanced redistributed charge scheme can be combined with parameteraized pseudo link atom. (*159, 160, 162*) The detailed description of the tuned pseudo link atom is in Sec. 4.H.

4.D. The Polarized-Boundary Treatment

4.D.1. General Description of the Polarized-Boundary Treatment

In the RC and RCD schemes,(*82*) the QM subsystem (CPS) is polarized by the MM system (SS), but the MM subsystem is not polarized by the QM subsystem, resulting in an unbalanced treatment of the electrostatic interactions between the QM and MM subsystems. The polarized-boundary RC (PBRC) and polarized-boundary RCD (PBRCD) schemes(*84*) make improvements in this regard to the RC and RCD schemes, respectively, by allowing self-consistent mutual polarizations between the QM and MM subsystems near the boundary. In the PBRC and PBRCD schemes, the polarizations of the SS due to the CPS were accomplished by adjusting the SS atomic partial charges in the embedded-QM calculations according to the principles of electronegativity equalization and charge conservations. We have implemented three literature methods based on electronegativity equalization for the determination of the SS atomic partial charges: (1) the charge equalization method proposed by Rappé and Goddard(*119*) (denoted as

QEq-SCT), (2) a modified version of charge equalization method by Bakowies and Thiel(*26*) (denoted as QEq-BT), and (3) the so-called electronegativity equalization (EEM) method by Mortier and coworkers (*120*) with the parameters by Bultinck et al. (*121*)

The procedure employed for the self-consistent polarization is as follows: First, the SS atomic partial charges are assigned to the SS atoms using the MM partial point charges, and the embedded-QM calculation is performed. Second, the electric field generated by the CPS (nuclei and electronic wavefunctions) is computed and imposed on the SS, and a new set of SS atomic partial charges is determined according to the electronegativity equalization and charge conservation. Third, the new set of partial charges replaces the old set of partial charges, and a new embedded-QM calculation is performed with the updated partial charges. Next, the electric field generated by the CPS is recalculated, and the partial charges are modified again. The loop is followed until convergence, i.e., until the variations in partial charges are smaller than preset thresholds, or until the number of iterations exceeds a preset value.

Here, we want to point out two general considerations. First, we note that the variation of charges only affects the embedded-QM calculations and does not affect pure MM calculations of E(MM;ES) and E(MM*;CPS). Although it is a common practice to use the MM charge parameters as partial atomic charges for embedded-QM calculations, and we indeed use them here in the first step to initiate the polarization treatment, the MM charges are not designed for embedded-QM calculations. The MM charges are part of the whole MM force field, which has been parameterized to reproduce reference data from experiments and/or high-level QM computations. The use of MM charges in embedded-QM calculations is in principle not appropriate, though convenient. For the same reason, it is also clear that the partial atomic charges resulting from the polarization procedure based on the embedded-QM calculations should not replace the MM charge parameters in pure MM calculations.

The second thing we want to mention is that we have adopted an “intra-group charge transfer” prescription. In this prescription, charge transfer is allowed within each group but prohibited between groups. In many MM force fields, the total charge for a functional group is often constrained to zero (or an integer) during the parameterization, so that the charge parameters can be easily transferred to other molecules with similar chemical groupings. A simple example is the CH_2 group in an *n*-butane molecule. A water molecule, which does not covalently bond to any other molecules, forms a group itself. A group can also be constructed for

special purposes by selectively putting together a number of atoms that are connected to each other via covalent bonds but not necessarily belong to the same functional group. The intra-group-charge-transfer prescription is a crude approximation since it prohibits charge transfer between two groups even if they are covalently linked to each other, but it is practical for treating large molecules such as proteins for the following reasons: First, it helps to retain the transferability of the group parameters. Second, allowing charge transfer among all atoms may cause a "catastrophe" in the charges, leading to failure of self-consistent convergence or leading to the convergence to apparently unrealistic charges. Finally, the charge equalization of a large number of atoms is computationally rather expensive, because one has to solve a large number of coupled equations. Thus, confining the charge transfers to within each group helps to reduce the computational cost significantly.

In the intra-group-charge-transfer prescription, a given group was in the presence of the electric field generated by the CPS and the electric fields generated by the other SS groups; those electric fields are combined into one electric field, which is referred to as the "external electric field" in this manual. This means that this combined electric field is "external" to the given group, which is of our primary concern. The given group thus "feels" the charge distribution of the surroundings, and it responds to the external electric field generated by the surroundings by adjusting its charge distributions. In the PBRC and PBRCD methods, the M2 and M3 atoms are in the first group, which is called the "polarized" group, although they might not belong to the same functional group. All the other charges on the SS atoms as well as the redistributed charges q_0 retained their values, and they are in the second group called the "unpolarized" group. Although the redistributed charges q_0 are very close to the QM/MM boundary, the PBRC and PBRCD schemes do not allow them to change values for two reasons: First, the redistributed charges are classical mimics of the auxiliary hybrid orbitals in the GHO theory, in which each auxiliary orbital retains its electron occupation in the QM/MM calculations. Second, the redistributed-charge sites are very close to the M2 atom, and we found that the electronegativity equalization calculations in which q_0 is variable gives unrealistically large values for both q_0 and q_{M2}.

We note that users of QMMM are not obliged to follow the PBRC and PBRCD schemes of grouping the SS atoms. For a specific system, it is possible that other ways of grouping the SS atoms will lead to better results. It is possible to have more than one polarized group, with the

charge transfer being confined in each polarized group. A conceivable example is an MM water molecule close to the PS. The close location of this water molecule to the PS will suggest substantial polarization effect due to the PS. Thus it may be desirable to put the three atoms in this water molecule into a group and to allow charge transfer within that group. But such calculations should not be called PBRC and PBRCD. More general names used for other definitions of the polarized group or groups are polarized-RC (PRC) and polarized-RCD (PRCD). The PBRC and PBRCD schemes allowing the charge transfer within the M2 and M3 atoms are for cases where the QM/MM boundary goes through a covalent bond. For the cases where the QM/MM boundary does not go through a covalent bond, it seems more suitable to put together the SS atoms of the same functional group.

4.D.2. Treatments Based on the QEq Method and Its Variance

In the QEq approach by Rappé and Goddard, the total electrostatic energy of a molecule of *N* atoms is written as the sum of the energy of all atoms in the molecule and the inter-atomic electrostatic energy. Further details of the method are given in Refs. The QEq algorithm is one option that can be used to polarize the boundary groups in the PBRC and PBRCD methods. Two sets of parameters are available in QMMM: those of Rappé and Goddard(*119*) and those of Bakowise and Thiel.(*26*)

To reduce computational cost of the QEq method, the Coulomb interactions are evaluated approximately by using an empirical function. The function that is used in the Rappé-Goddard is as follows:

$$J_{\mathrm{AB}} = \frac{J_0}{\varepsilon R_{AB}}\ , \qquad (4.D.2.1)$$

where ε is the dielectric constant taken to be to 2, J_0 is defined as 14.4 eV·Å, and R_{AB} in Å is the distance between atoms A and B. We adopted this simple coulomb term empirical function together with (as the default) the parameters suggested by Rappé and Goddard. This option denoted QEq-SCT.

Bakowies and Thiel have proposed a modified QEq method, where the Klopman-Ohno function was employed to compute the Coulomb interactions:

$$J_{AB} = \frac{1}{\sqrt{R_{AB}^2 + \left(\frac{1}{2\eta_A} + \frac{1}{2\eta_B}\right)^2}}, \qquad (4.D.2.2)$$

where K^0 is 1 eVÅ . This option that uses Eq. 4.D.2.2 is denoted QEq-BT.

The default QEq-SCT and QEq-BT parameters are listed in Table 4.D.2.1.

Table 4.D.2.1. The QEq-SCT and QEq-BT parameters implemented in the QMMM program

Element	QEq-SCT		QEq-BT	
	χ^0(ev)	J^0(ev)	χ^0(ev)	J^0(ev)
Li	3.006	4.772	n/a	n/a
C	5.343	10.126	5.07305	10.06444
N	6.899	11.760	7.73699	12.96908
O	8.741	13.364	8.27885	14.93241
F	10.874	14.948	n/a	n/a
Na	2.843	4.592	n/a	n/a
Si	4.168	6.974	n/a	n/a
P	5.463	8.000	n/a	n/a
S	6.928	8.972	n/a	n/a
Cl	8.564	9.892	n/a	n/a
K	2.421	3.84	n/a	n/a
Br	7.790	8.850	n/a	n/a
Rb	2.331	3.692	n/a	n/a
I	6.822	7.524	n/a	n/a
Cs	2.183	3.422	n/a	n/a
H	4.528	13.890	4.42211	13.84036

4.D.3. Treatments Based on the EEM Model

An alternative to the QEq method for polarizing the boundary group in the PBRC and PBRCD schemes (or for polarizing more general polarizable groups in the PRC and PRCD calculations) is the EEM method. The EEM model, upon which the later QEq method was based, is described in Refs.(*120*) In our implementation, the default EEM parameters are taken from the 4th column of Table 1 in the paper of Bultinck and co-workers(*121*) (see Table 4.D.3.1), and the QM/MM calculations employing the EEM model are denoted EEM. The default parameters are listed in the following table.

Table 4.D.3.1. The EEM parameters implemented in the QMMM program

Element	EEM	
	χ^*(ev)	η^*(ev)
Li	n/a	n/a
C	2.30	9.1
N	7.20	13.2
O	5.10	11.1
H	1.00	13.8

4.E. Flexible-Boundary Treatment

The flexible-boundary treatment(*86, 87*) is based on the principle of electronic chemical potential equalization to account for partial charge transfers between the PS and SS. The QM subsystem is viewed as an open system with a fluctuating number of electrons and is described by a statistical mixture of ensemble that consists of states of integral number of electrons.(*122, 123*) The MM subsystem serves a reservoir that exchanges electrons with the QM subsystem. The electronic chemical potential of the MM subsystem varies when charges flow in or out until equilibrium is established for electronic chemical potentials between the PS and SS. Iterative treatments are therefore required to achieve self-consistence.

In the simplest situation, the PS consists of only two states, i.e., a reduced state (X) and an oxidized state (X^+). An example is the neutral sodium atom Na and the sodium cation Na^+. The oxidized state has a charge $q(X^+)$, and its molar fraction is x^+. The reduced state has a charge $q(X)$, and its molar fraction is $x = 1 - x^+$. Let us denote the charges on PS and SS as $q(PS)$ and $q(SS)$, respectively. Apparently, one has

$$q(PS) = (x_+ q(X^+) + (1 - x_+) q(X)) \tag{4.E.1}$$

We assume the existence of equilibrium of CPS ionization

$$X^+ + e^- \leftrightarrow X \tag{4.E.2}$$

This leads to

$$\mu(X^+) + \mu(e^-) = \mu(X) \tag{4.E.3}$$

Here, $\mu(X^+)$ is the chemical potential of X^+, $\mu(e^-)$ is the chemical potential of the free electrons, and $\mu(X)$ is the chemical potentials of X. Now assume that the free electrons are in equilibrium of the electrons in reservoir (SS),

$$\mu(e^-) = \mu(SS) = \mu \tag{4.E.4}$$

The electronic chemical potential of the SS, $\mu(SS)$, is related to the chemical potential for charge transfer, or electronegativity, χ, by:

$$\mu(SS) = -\chi \tag{4.E.5}$$

Eq. (4.E.4) is the central equation in our flexible-boundary treatment, which must be satisfied when the equilibrium is established for electron exchange between the PS and SS.

As in the polarizable-boundary treatments, the redistributed charge q_0 are fixed and do not vary during the polarization calculations.

The different zeroes of chemical potentials for electron (or charge) between QM calculations and the QEq-SCT calculations require calibration before Eq. (4.E.4) can be applied. The calibration can therefore be done as follows: First, one computes the QEq-SCT electronegativity χ_{cali} for the PS of a charge corresponding to $x_+ = 0.5$; for example, if $q(X) = 0$ and $q(X^+) = 1$ e, χ_{cali} will be calculated with the PS charge set to 0.5 e. An energy term E_{cali} is determined by comparing χ_{cali} and the ionization energy $I(X)$ for the PS in the gas phase:

$$-I(X) + E_{cali} = -\chi_{cali} \tag{4.E.6}$$

The electronic temperature is another parameter that can be adjusted. In our implementation, the user can manually specify the value of the electronic temperature, or let the program determine the value automatically; the latter option is the default option. The program determines the electronic temperature based on the following consideration:

$$d\mu/dq(PS) = -k_B T\,(x^+(1-x^+))^{-1}(q(X^+) - q(X))^{-1} \tag{4.E.7}$$

$$d^2\mu/dq(PS)^2 = (4k_B T/(q(X) - q(X^+))^2)\,((1-2x^+)/(x^+)^2(1-x^+)^2) \tag{4.E.8}$$

at $x^+ = 0.5$, where the molar fraction of the reduced and oxidized states are equal,

$$d\mu/dq(PS) = 4k_B T/(q(X) - q(X^+)) \tag{4.E.9}$$

$$d^2\mu/dq(PS)^2 = 0 \tag{4.E.10}$$

This tells us that the $d\mu/dq$ curves are linear at $x = 0.5$.

The QEq method (or other electronegativity equalization classic models) yields a linear line for μ as a function of charge q on the CPS. Therefore, for calibration purpose, we let the slope computed by Equation (4.E.9) equal the chemical potential slope calculated by QEq model:

$$d\mu/dq\,\big|_{x=0.5} = d\mu(QEq)/dq \tag{4.E.11}$$

The $d\mu(QEq)/dq$ can be computed numerically by

$$d\mu(QEq)/dq = (\mu(X)_{QEq} - \mu(X^+)_{QEq})]/(q(X) - q(X^+)) \tag{4.E.12}$$

where $\mu(X)_{QEq}$ and $\mu(X^+)_{QEq}$ are the chemical potentials calculated by the QEq model for the reduced and oxidized states.

We have implemented an iterative procedure for self-consistent. The loop continues until self-consistency is achieved, i.e., the changes in the charge distribution of the SS are smaller than preset thresholds.

4.F. Screened Charge Scheme

Point charges may be used for MM atoms in the QM/MM method, but one can also use more advanced treatments of the electrostatics. for example, one may use the screened charge scheme, which improves the point charge scheme by taking the penetration effects of MM atoms into consideration. The charge density of an atom in the MM subsystem is represented by two components: (i) a smeared charge $-n_{\text{screen}}$ distributed like electrons in the Slater-type orbital $j_{(n)} = Ar^{n-1}\exp(-zr)$, which models the extension of the MM electron density and (ii) the rest of the charge, which is located at the nucleus. The comparison of a point charge model and a screened charge model is shown in Figure 2.

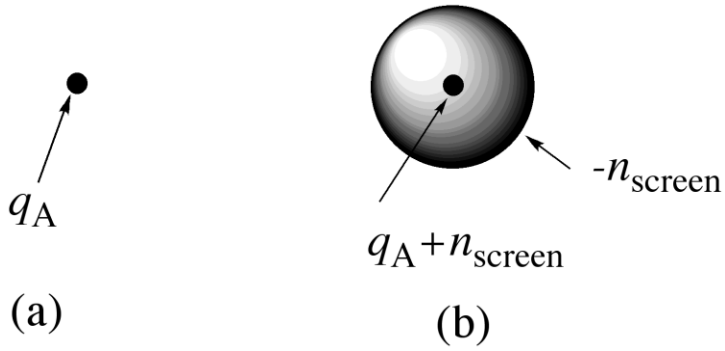


Figure 2. Comparison between (a) a point charge model and (b) a screened charge model of an MM atom A. The total smeared charge in model (b) is $-n_{\text{screen}}$, representing n_{screen} electrons.

Based on the new model, we can calculate the effective charge of the MM atom A.

$$q_A^* = q_A + n_{\text{screen}} f(zr)\exp(-2zr) \tag{4.F.1}$$

where the scaling factor f is

$$\begin{aligned} f(zr) &= 1 + zr && n = 1 \\ &= 1 + \frac{3}{2}zr + (zr)^2 + \frac{1}{3}(zr)^3 && n = 2 \\ &= 1 + \frac{5}{3}zr + \frac{4}{3}(zr)^2 + \frac{2}{3}(zr)^3 + \frac{2}{9}(zr)^4 + \frac{2}{45}(zr)^5 && n = 3 \\ &= 1 + \frac{7}{4}zr + \frac{3}{2}(zr)^2 + \frac{5}{6}(zr)^3 + \frac{1}{3}(zr)^4 + \frac{1}{10}(zr)^5 + \frac{1}{45}(zr)^6 + \frac{1}{315}(zr)^7 && n = 4 \end{aligned} \tag{4.F.2}$$

As compared to a point charge q_A, the expression for q^*_A has an additional term to include the penetration effects. This term goes to zero when r approaches infinity and goes to $q_A + n_{\text{screen}}$

when r approach 0. The parameters of ζ for common atoms (H, C, N, O, F, Si, P, S, Cl and Br) have been optimized against a test suite of 40 molecules with two different basis sets and three different geometries. They are listed in the Table 4.F.1. We recommend using optimized parameters for H, C, N, O, F, Si, P, S, Cl and Br, and MSB parameters for the elements whose parameters are not optimized.

Table 4.F.1. ζ values used in the Slater-type orbital

Atom	H	B	C	N	O	F	Si
optimized parameters	1.32	−	0.92	0.92	1.20	1.16	0.73
MSB parameters[a]	1.32	0.72	0.87	1.01	1.12	1.24	0.74
Atom	P	S	Cl	Ge	As	Se	Br
optimized parameters	0.68	0.90	0.98	−	−	−	0.91
MSB parameters[a]	0.81	0.88	0.95	0.83	0.88	0.95	1.01

[a] modified Strand-Bonham (MSB) parameters (optimized parameter for H and half of the Strand-Bonham parameters for B through Br)

The number of screened electrons is shown in equation 4.F.3

$$n_{\text{screen}} = \begin{cases} 1-q & \text{for H} \\ 1 & \text{for elements in Table 4.F.1 except H} \\ 0 \text{ (point charge)} & \text{for other elements} \end{cases} \quad (4.F.3)$$

where q is the partial atomic charge on H.

In the current implementation, the screened charges are only used to evaluate the QM–MM electrostatic interactions, while the MM–MM electrostatic interactions are still evaluated by point charges. Since the van der Waals terms used in the QM–MM interactions have been parametrized based on MM point charges, new van der Waals force field parameters would be needed if the screened charge scheme were used to handle the electrostatic interactions. Such new force field parameters have not been developed and therefore are not in the current program.

4.G. Smeared Redistributed Charges

In our study, we found that when the redistributed charges are close to the QM/MM boundary, these point charges can overpolarize the QM region. QMMM can alleviate this problem by smearing the redistributed charges. For the option of charge smearing, the redistributed charge q_{MM} is placed in a normalized Slater-type orbital

$$j = \exp(-r/r_0)/(\pi r_0^3)^{1/2} \tag{4.G.1}$$

where r is the distance from ithe center of the charge density, and r_0 is the smearing width. Then the charge density of the MM charge q_{MM} is expressed as

$$r_{\mathrm{MM}}(r) = q_{\mathrm{MM}} \exp(-2r/r_0)/(\pi r_0^3) \tag{4.G.2}$$

QMMM calculates the electrostatic potential generated by the smeared charge; this yields the effective charge as

$$q^*_{\mathrm{MM}} = q_{\mathrm{MM}} - q_{\mathrm{MM}}(1+\frac{r}{r_0})\exp(-2r/r_0) \tag{4.G.3}$$

The difference between the screened charges and the smeared charges is that we delocalize the outermost electron density in the screened charge scheme, while we delocalize the total charge distrbution in the smeared charge scheme.

4.H. Link Atoms

The position of the link atom is another important issue in the link-atom schemes, and it has been investigated extensively.(*25, 49, 56, 124*) Following the argument in Section 4.B that the active orbital is represented by a link atom HL, a natural location for HL is on the Q1–M1 bond with the Q1–HL distance depending on the Q1–M1 distance. Hence, we adopt the scaled-bond-distance method proposed by Maseras, Morokuma, and co-workers:(*11, 14*) The Q1–HL distance, *R*(Q1–HL), is related to the Q1–M1 distance, *R*(Q1–M1), by a scaling factor

$$R(\mathrm{Q1–HL}) = C_{HL}\, R(\mathrm{Q1–M1}) \tag{4.H.1}$$

During a QM/MM geometry optimization or a molecular dynamics or reaction path calculation, the equilibrium Q1–HL and Q1–M1 distances are constrained to satisfy equation (4.H.1).

The scaling factor, C_{HL}, depends on the nature of the bonds being cut and constructed. We set the scaling factor by

$$C_{HL} = R_0(\text{Q1–H}) / R_0(\text{Q1–M1}), \qquad (4.H.2)$$

where R_0(Q1–H) and R_0(Q1–M1) are the MM bond distance parameters for the Q1–H and Q1–M1 stretches, respectively.

Another fixed-bond-distance method to define the position of the link atom *L* is also implemented. This procedure, described next, is the same as a procedure that has been applied in the QM/MM scheme in *Amber* 10. (*163*) In this method, the Q1–L bond length is fixed during the QM/MM optimization. The link atom is placed along the bond vector joining Q1 and M1, and the position of the link atom is defined as:

$$\mathbf{r}_{\mathrm{L}} = \mathbf{r}_{\mathrm{Q1}} + D_{\mathrm{HL}} \frac{\mathbf{r}_{\mathrm{M1}} - \mathbf{r}_{\mathrm{Q1}}}{\left|\mathbf{r}_{\mathrm{M1}} - \mathbf{r}_{\mathrm{Q1}}\right|} \qquad (4.H.3)$$

where $\mathbf{r}_{\mathrm{L}}$, $\mathbf{r}_{\mathrm{Q1}}$, and $\mathbf{r}_{\mathrm{M1}}$ are the positions of the link atom, the Q1 atom, and the M1 atom, respectively; and D_{HL} is the fixed bond length of Q1–L, which is assigned as the standard Q1–L bond length in whatever force field is used for the MM calculations.

We can also vary the type of the link atom; in particular, H, F, tuned F, or other atoms can be used. If a tuned F atom is used, the parameters used for link atom should be provided in the input file. In tuned methods, the link atom is a tuned F atom, which is an atom that has an adjustable pseudopotential centered at its nucleus. The pseudopotential is given by

$$U(r) = C\exp[-(r/r_{\mathrm{T}})^2] \qquad (4.H.4)$$

where *C* is the tuning parameter, and r_{T} is $1\,a_0$. The pseudopotential is tuned to make the sum of the partial charges of the uncapped portion of the QM subsystem equal a target value. The partial charges are computed by Mulliken analysis with a 6-31G* basis set when the M1 atom is from the second period (Li through F) and with an STO-3G basis set otherwise, or by CM5 charges. (*159, 160, 162*) It is suggested that when the tuned F link atom is used, the fixed-bond-distance method should be used to define the link atom for QM/MM geometry optimization.

4.I. QM/MM Energy

The QM/MM energy is defined by

$$E(\text{QM/MM;ES}) = E(\text{MM;ES}) - E(\text{MM;CPS*}) + E(\text{QM;CPS**}), \qquad (4.I.1)$$

where ES and CPS denote the entire system and capped primary system, respectively, the CPS is the primary system capped by the link atom HL, the asterisk (*) denotes that the CPS is

embedded in the electrostatic field of the secondary subsystem (SS), and the double asterisks (**) denote such an embedding in an appropriately modified electrostatic field of the SS; the SS is defined as

$$SS = ES - PS, \tag{4.I.2}$$

where PS denotes the primary system. The PS is the QM subsystem, and the SS is the MM subsystem.

In equation (4.I.1), the MM energy for the ES, E(MM;ES), is a sum of the valence (stretch, bend, and torsion) energy E(Val;ES), the van der Waals energy E(vdW;ES), and the Coulombic energy E(Coul;ES):

$$E(\text{MM;ES}) = E(\text{Val;ES}) + E(\text{vdW;ES}) + E(\text{Coul;ES}). \tag{4.I.3}$$

The term E(MM;CPS*) is the MM energy for the CPS that is embedded in the background charge distribution of the SS. It includes both the MM energy for CPS itself, E(MM;CPS), and the Coulombic interaction energy between the CPS and the SS, E(Coul;CPS|SS):

$$E(\text{MM;CPS*}) = E(\text{MM;CPS}) + E(\text{Coul;CPS|SS}), \tag{4.I.4}$$

The term E(MM;CPS) also consists of three contributions, i.e., the valence energy E(Val;CPS), the van der Waals energy E(vdW;CPS), and the Coulombic energy E(Coul;CPS):

$$E(\text{MM;CPS}) = E(\text{Val;CPS}) + E(\text{vdW;CPS}) + E(\text{Coul;CPS}). \tag{4.I.5}$$

Special scaling factors are often used in MM force field for calculations of Coulombic interactions for pairs of atoms that are connected by a valence potential. For example, Coulombic interactions between neightboring or geminal atoms are neglected, and Coulombic interactions between vicinal atoms maybe neglected(*108*) or scaled by 0.5.(*109, 125-129*) This feature is retained in the calculations for E(Coul;ES) and E(Coul;CPS|SS) in equations (4.I.3) and (4.I.4).

The term E(QM;CPS**) is the QM energy for the CPS with a background charge distribution of the SS that has been modified by an appropriate boundary treatment. In particular, the M1 charge has been redistributed in the RC and RCD schemes, and the M1 and M2 charges have been modified to restore the contribution of q_0 to the M1–M2 bond dipole if the RCD scheme is adopted, or if the RCD2 scheme is selected, an additional pair of point charges q_- and q_+ are added to preserve the M1–M2 bond dipole. The only modification required for the electronic structure program to calculate E(QM;CPS**) is that it can carry out a calculation in the presence of a background point charge distribution; many electronic structure codes already

have this capability, and the required integral types are the ones already required for the nuclear attraction term.

One notices that both *E*(MM;ES) and *E*(MM;CPS*) include the MM energies for the PS, and thus those MM terms that involve only the PS atoms are completely cancelled in computations employing equation (4.I.1). On the other hand, those terms involving only SS atoms survive, and they provide the MM descriptions of the SS system. Also surviving in *E*(MM;CPS*) are the MM terms for interactions between the PS atoms and the HL atom, which can be considered as corrections to the QM calculations for the CPS.(*25*) (Note that the PS-HL Coulombic terms vanish because $q_{HL} = 0$.) The final group of MM energy terms that survive are interactions between CPS and SS; they are more complicated as discussed next.

First, we consider at the valence interactions *E*(Val;CPS|SS). The surviving terms are the Q1–M1 stretch, the Q2–Q1–M1 and Q1–M1–M2 bends, and the Q3–Q2–Q1–M1, Q2–Q1–M1–M2, and Q1–M1–M2–M3 torsions. The second kind of MM interactions that we consider is the non-bonded van der Waals interactions. We retain the contributions in *E*(vdW;PS|SS) that describe the interactions between the PS and SS, but the HL atom is not seen by the SS. The final type of CPS|SS interaction is Coulombic. We note that

$$E(\text{Coul;ES}) = E(\text{Coul;SS}) + E(\text{Coul;PS}) + E(\text{Coul;PS|SS}), \qquad (4.\text{I}.6)$$

Using equations (4.I.4) to (4.I.6) and the fact that the MM charge on the HL atom is zero, one finds that the *E*(Coul;PS) and *E*(Coul;PS|SS) terms cancel exactly. This is what we might have expected, because now the electrostatic interactions between PS and SS are handled at the QM/MM level, i.e., by the *E*(QM;CPS**) computations. Since the *E*(Coul;PS) and *E*(Coul;PS|SS) terms cancel exactly, the calculations can be simplified by setting all the MM charges on the PS atoms to zero.

The final expression for the QM/MM energy is therefore given as follows:

$$\begin{aligned} E(\text{QM/MM;ES}) = {} & (E(\text{Val;ES}) - E(\text{Val;CPS})) \\ & + (E(\text{vdW;ES}) - E(\text{vdW;CPS})) \\ & + E(\text{Coul;SS}) + E(\text{QM;CPS**}). \end{aligned} \qquad (4.\text{I}.7)$$

One special note here is related to the treatment of polar bonds for which the charge on the link atom cannot be neglected. We use the modified (balanced) M1 charge, rather than the

original M1 charge, to calculate *E*(Coul;SS) in 4.I.7.(*160*) In the special case in which the link atom bears zero charge, the modified M1 charge is the same as the original M1 charge.

4.J. Mechanical Embedding

The RC and RCD schemes discussed so far are electric embedding schemes. Another commonly used QM/MM scheme is the so-called mechanical embedding (ME) scheme,(*25*) which is the same as the original integrated molecular-orbital molecular-mechanics (IMOMM) scheme.(*11, 12, 124*) In the ME scheme, the CPS calculations are performed in gas phase, i.e., without the background charge distribution for the SS. The QM/MM energy is defined by

$$E(\text{QM/MM;ES}) = E(\text{MM;ES}) - E(\text{MM;CPS}) + E(\text{QM;CPS}). \qquad (4.\text{J}.1)$$

The electrostatic interactions between the PS (or CPS) and the SS are included in the *E*(MM;ES) term, i.e., they are handled at the MM level and require MM charge parameters for the ES. Thus, in contrast to the electronic embedding schemes, where one does not require the MM charge parameters for the PS atoms, the ME treatment relies on the availability of MM charge parameters for the PS atoms. This creates a problem for studying reactions, and the problem is especially serious for processes accompanied by charge transfer, such as a proton transfer reaction or an electron transfer reaction. Unlike the van der Waals interactions, the electrostatic interaction is long-range, and the use of inappropriate charge parameters can cause a serious error. For this reason, we do not recommend the ME scheme for most applications although we have implemented the ME scheme in the QMMM program.

4.K. Other Electrostatic-embedding Schemes Implemented in QMMM

In addition to the RC and RCD methods, QMMM also contains some other electronic embedding schemes for charge manipulation in the link atom approaches. The first one is the straight electronic embedding (SEE) where no special treatment for the background MM charges is performed; in particular, there is no redistribution, scaling, or zeroing of MM partial charges.

Four additional methods are included in QMMM; these methods, as employed here, differ from the RC and RCD schemes only in the treatment of the electronic embedding in the *E*(QM;CPS**) term of Equation (4.I.4). Three of these other methods are eliminated charge (*2*) schemes where selected MM point charges are eliminated. If only the M1 charge is zeroed, it is called the Z1 scheme. If both M1 and M2 charges are zeroed, it is called the Z2 scheme; and Z3

denotes the treatment where all M1, M2, and M3 charges are zeroed. It should be noted that the Z1, Z2, and Z3 schemes may not preserve the overall charge for the system under study, e.g., a neutral system may become partially negatively charged. It is also interesting to note that Z3 is the default scheme (i.e., the *scalecharge=500* keyword) in ONIOM calculations by *Gaussian*. The Z1 and Z2 schemes are equivalent to using *scalecharge=555* and *scalecharge=550* keywords, respecitively, in ONIOM calculations by *Gaussian*. (The SEE scheme can be considered as Z0 since no MM charges are zeroed; however, the SEE scheme is not available in *Gaussian* since the M1 charge is always set to zero.)

The fourth scheme is the shifted charge ("Shift") scheme,(*70*) where the M1 charge is evenly shifted onto all M2 atoms, and a pair of point charges (q_- and q_+) is added in the vicinity of M2 to preserve M1–M2 bond dipole. The distance between this pair of point charges is set to 20% of the M1–M2 bond distance by default, i.e.,

$$q_-^{\text{Shift}} = -q_+^{\text{Shift}} = q_0 / Cq_\pm \qquad (4.K.1)$$

$$Cq_\pm = R(q_- - q_+) / R(\text{M1}-\text{M2}) \qquad (4.K.2)$$

and

$$q_{\text{M2},k}^{\text{Shift}} = q_{\text{M2},k} + q_0 \qquad (4.K.3)$$

where $Cq_\pm = 0.2$. Here, $q_{\text{M2},k}^{\text{Shift}}$ is the modified M2 charge. In comparison with the RCD2 method, there is no redistributed point charge q_0, and

$$q_-^{\text{Shift}} = 2q_-^{\text{RCD2}} \qquad (4.K.4)$$

if the same values are used for $Cq_\pm$.

In all these schemes, q_{HL} is zero.

QMMM also contains several different balanced schemes. A balanced scheme is one in which the redistributed charges are adjusted to conserve the total charge of the system. These methods include: balanced straight electrostatic embedding (SEE), balanced RC2, Amber-1, balanced RC3, Amber-2, and balanced Shift. Amber-1 and Amber-2 are the options called *adjust_q* = 1 and *adjust_q* = 2 in *AMBER 10*. The distinction between these methods is in the position of the redistributed charges and whether the dipoles of M1–M2 bonds are corrected. In balanced SEE, the charge on the M1 atom is set to *q*, and it is not moved. In balanced RC2, we distribute *q* evenly to all M2 atoms. In balanced RC3, we distribute *q* evenly to all M2 and M3 atoms. In

Amber-1, we move *q* to the nearest M2 atom. In Amber-2, we distribute *q* evenly to all MM atoms, except the M1 atom. In balanced Shift, the redistributed charges are placed at M2 atoms, and dipoles are added around M2 atoms to compensate the movement of the charges. A summary of these charge schemes is shown in Table 4.K.1. The boundary charge schemes can be used with charge smearing schemes or screened charge schemes.

TABLE 4.K.1: Balanced charge schemes

	Position of the redistributed charges	Correction of bond dipole
balanced SEE	M1 atom	No
balanced RC	midpoints of M1–M2 bonds	No
balanced RC2	M2 atoms	No
Amber-1	Nearest M2 atom	No
balanced RC3	M2, M3 atoms	No
Amber-2	all MM atoms (except M1 atoms)	No
balanced RCD	midpoints between M1 and M2 atoms	Yes
balanced Shift	M2 atoms	Yes

We note that since we made the schemes as simple as possible to promote clarity and portability, our implementations for these schemes might not exactly be the same as other groups' implementations of their schemes. For example, the parameters selected and the treatments for locating the link atom position could be different. It is therefore not possible to make direct comparisons of our results with other groups' works based on the published literature.

For the sake of clarifying the differences between the methods, it is useful at this point to consider the limit of the Z*n* schemes as $n \rightarrow \infty$; we call this Z∞. In particular, we point out that the Z∞ scheme is not the same as mechanical embedding for two reasons. First, in the presence of a solvent or other non-bonded environment (e.g., a protein or a supramolecular cage), the Z∞ method does not zero out all charges but only those connected by a sequence of bonds to Q1. Second, the mechanical embedding scheme differs from the Z∞ scheme in the middle terms of equations (5) and (12). Thus, in the absence of non-bonded moieties, QM/MM electrostatic interactions would cancel out in Z∞, but not in ME.

4.L. Energy Derivatives

Two schemes have been implemented to define the location of the link atom, namely the scaled-bond-distance method and the fixed-bond-distance method. We will discuss them separately.

4.L.1. Scaled-bond-distance method

Energy derivatives (gradients and Hessians) can be obtained through chain rule as suggested by Morokuma et al. (*14*) Let us suppose there are two non-redundant sets of coordinates, S_1 and S_2, both describing the system of interest, and that the transform matrix $\boldsymbol{J}$ between these two sets of coordinates can be written as:

$$\boldsymbol{J} = \partial S_1 / \partial S_2 \qquad (4.L.1)$$

The gradient can be transformed by

$$\nabla S_2 = \boldsymbol{J}\, \nabla S_1 \qquad (4.L.2)$$

and the Hessian can be transformed by

$$\Delta S_2 = \boldsymbol{J}^{\mathrm{T}}\, \Delta S_1\, \boldsymbol{J} \qquad (4.L.3)$$

where $\boldsymbol{J}^{\mathrm{T}}$ is the transpose of $\boldsymbol{J}$.

First we consider gradient calculations. Suppose that we have an HL atom, whose Cartesian coordinate component X_{HL} depends on the Cartesian coordinates component of Q1 (X_{Q1}) and M1 (X_{M1}), by

$$X_{\mathrm{HL}} = (1 - C_{\mathrm{HL}})\, X_{\mathrm{Q1}} + C_{\mathrm{HL}}\, X_{\mathrm{M1}} \qquad (4.L.4)$$

where C_{HL} is the scaling factor described in equation (4.H.2). One obtains the following from Equation (4.L.4):

$$\partial E/\partial X_{\mathrm{Q1}} = \partial' E/\partial X_{\mathrm{Q1}} + \partial' E/\partial X_{\mathrm{HL}}\, \partial X_{\mathrm{HL}}/\partial X_{\mathrm{Q1}}$$
$$= \partial' E/\partial X_{\mathrm{Q1}} + (1 - C_{\mathrm{HL}})\, \partial' E/\partial X_{\mathrm{HL}} \qquad (4.L.5)$$

where we have introduced the convention that $\partial/\partial X_{\mathrm{Q1}}$ denotes a partial derivative of a function considered as a function of all the real coordinates (X_{Q1}, Y_{Q1}, …, X_{M1}, Y_{M1}, …), and $\partial'/\partial X_{\mathrm{Q1}}$ denotes a partial derivative of a function considered as a function of all the real coordinates plus the coordinates of the link atom (X_{Q1}, Y_{Q1}, …, X_{M1}, Y_{M1}, …, X_{HL}, Y_{HL}, Z_{HL}). Similarly,

$$\partial E/\partial X_{\mathrm{M1}} = \partial' E/\partial X_{\mathrm{M1}} + \partial' E/\partial X_{\mathrm{HL}}\, \partial X_{\mathrm{HL}}/\partial X_{\mathrm{M1}}$$
$$= \partial' E/\partial X_{\mathrm{M1}} + C_{\mathrm{HL}}\, \partial' E/\partial X_{\mathrm{HL}} \qquad (4.L.6)$$

For all schemes implemented in QMMM, the gradients on the Q1 atoms are determined by Equation (4.L.5). For the ME, SEE, Z1, Z2, and Z3 schemes, the gradients on the M1 atoms are

evaluated by Equation (4.L.6), and there is no special treatment for the gradients on the M2 atoms. For the RC, RCD, RCD2, and Shift scheme, however, we need to consider the redistributed point charge q_0 and the additional charge pair (q_- and q_+), and the gradients on the M1 and M2 atoms are more complicated, as will be addressed below.

The location for redistributed point charges q_0 and the pair of additional point charges (q_- and q_+) depends on the position of M1 and M2. Let us suppose that the Cartesian coordinate component of q_0 depends on the Cartesian coordinate components of M1 and M2 by

$$Xq_0 = (1 - Cq_0)\, X_{M1} + Cq_0\, X_{M2} \qquad (4.L.7)$$

where $Cq_0 = 0.5$ by default. For q_- and q_+, one has

$$Xq_- = (1 - Cq_-)\, X_{M1} + Cq_-\, X_{M2} \qquad (4.L.8)$$

$$Xq_+ = (1 - Cq_+)\, X_{M1} + Cq_+\, X_{M2} \qquad (4.L.9)$$

$$Cq_- = 1 - Cq_\pm / 2 \qquad (4.L.10)$$

$$Cq_+ = 1 + Cq_\pm / 2 \qquad (4.L.11)$$

The $Cq_\pm$ is defined by equation (4.C.9) and is set to 0.2 by default in the QMMM program. It is straightforward to work out the expressions for gradients for q_0, q_-, and q_+ in a completely similar way to Equations (4.L.5) and (4.L.6) for the HL; these expressions are not given here.

Taking into account the contributions from q_0, q_-, and q_+, the gradients on the M1 atoms is given by

$$\partial E/\partial X_{M1} = \partial' E/\partial X_{M1} + C_{HL}\, \partial' E/\partial X_{HL} + \Sigma_{i=1,2,3}\, (1 - Cq_{0i})\, \partial' E/\partial Xq_{0i} \qquad (4.L.12)$$

for the RC and RCD schemes,

$$\partial E/\partial X_{M1} = \partial' E/\partial X_{M1} + C_{HL}\, \partial' E/\partial X_{HL} + \Sigma_{i=1,2,3}\, (1 - Cq_{0i})\, \partial E'/\partial Xq_{0i} + \Sigma_{i=1,2,3}\, [(1 - Cq_{-i})\, \partial' E/\partial Xq_{-i} + (1 - Cq_{+i})\, \partial' E/\partial Xq_{+i}] \qquad (4.L.13)$$

for the RCD2 scheme, and

$$\partial E/\partial X_{M1} = \partial' E/\partial X_{M1} + C_{HL}\, \partial' E/\partial X_{HL} + \Sigma_{i=1,2,3}\, [(1 - Cq_{-i})\, \partial' E/\partial Xq_{-i} + (1 - Cq_{+i})\, \partial' E/\partial Xq_{+i}] \qquad (4.L.14)$$

for the Shift scheme.

The gradients on the M2 atoms is given by

$$\partial E/\partial X_{M2} = \partial' E/\partial X_{M2} + \Sigma_{i=1,2,3}\, Cq_{0i}\, \partial' E/\partial Xq_{0i} \qquad (4.L.15)$$

for the RC and RCD schemes,

$$\partial E/\partial X_{M2} = \partial' E/\partial X_{M2} + \Sigma_{i=1,2,3}\, Cq_{0i}\, \partial' E/\partial Xq_{0i} + \Sigma_{i=1,2,3}\, [Cq_{-i}\, \partial' E/\partial Xq_{-i} + Cq_{+i}\, \partial' E/\partial Xq_{+i}] \qquad (4.L.16)$$

for the RCD2 scheme, and

$$\partial E/\partial X_{M2} = \partial' E/\partial X_{M2} + \Sigma_{i=1,2,3}\, [Cq_{-i}\, \partial' E/\partial Xq_{-i} + Cq_{+i}\, \partial' E/\partial Xq_{+i}] \qquad (4.L.17)$$

for the Shift scheme.

Next, we look at the Hessian calculations, which are even more complicated. The central problem is how to partition the contributions for Hessian elements involving the link atoms (HL) and auxiliary point charges (q_0, q_-, and q_+) that are not present in the real system into the Hessian elements for real atoms. Here we use the term "real system" instead of entire system to distinguish the atoms in the ES from the link atoms and auxiliary point charges, which can be viewed as "artificial" atoms and whose coordinates depends on the real- atom coordinates. Let us use X_{p1} and X_{p2} to denote two of the Cartesian coordinate components for the link atoms and auxiliary point charges. We further assume that X_{p1} depends on the real-atom Cartesian coordinate components X_{1a} and X_{1b} with a scale factor g_1, and that X_{p2} depends on the real-atom Cartesian coordinate components X_{2a} and X_{2b} with a scale factor g_2.

$$X_{p1} = (1 - g_1)\, X_{1a} + g_1\, X_{1b} \qquad (4.L.18)$$

$$X_{p2} = (1 - g_2)\, X_{2a} + g_2\, X_{2b} \qquad (4.L.19)$$

Please note that we make no assumption on the relation between X_{1a}, X_{2a}, X_{1b}, and X_{2b}. Therefore, these four coordinate components can be different from each other ($X_{1a} \neq X_{2a}$, $X_{1a} \neq X_{2b}$, $X_{1b} \neq X_{2a}$, and $X_{1b} \neq X_{2b}$), or some of them can be referred to the same quantity (e.g., $X_{1a} = X_{2a}$). Finally, we assume that neither X_{p1} nor X_{p2} depends on the real-atom coordinate X_0. We will also use simplified notations to label Hessian elements: $\partial^2 H/\partial X \partial Y$ (or $\partial'^2 H/\partial X \partial Y$) is denoted XY.

The first step we take is to decompose each of the Hessian elements involving X_{p1} (i.e., X_0X_{p1}, $X_{1a}X_{p1}$, $X_{1b}X_{p1}$, $X_{2a}X_{p1}$, $X_{2b}X_{p1}$, $X_{p1}X_{p1}$, or $X_{p1}X_{p2}$) into certain contributions, and we add these contributions to specific Hessian elements for the real-atom coordinate components (X_0, X_{1a}, …, X_{2b}). This is done in accord with Table 4.L.1. One can see that only those Hessian elements involving X_{1a} and/or X_{1b} are updated, and the Hessian elements involving only X_0, X_{1a} and X_{1b} are not affected.

Secondly, we repeat the operation for Hessian elements involving X_{p2}, leaving out those elements involving both X_{p1} and X_{p2} that have been treated in the first step. That is, we will treat X_0X_{p2}, $X_{1a}X_{p2}$, $X_{1b}X_{p2}$, $X_{2a}X_{p2}$, $X_{2b}X_{p2}$, and $X_{p2}X_{p2}$ this time. By repeating the operation and going over all of the Cartesian coordinate components for the link atoms and auxiliary charges, the Hessian matrix for the real system is obtained, and vibrational analysis can be performed.

Table 4.L.1. Decomposition of the Hessian elements for X_0X_{p1}, $X_{1a}X_{p1}$, $X_{1b}X_{p1}$, $X_{2a}X_{p1}$, $X_{2b}X_{p1}$, $X_{p1}X_{p1}$, and $X_{p1}X_{p2}$ and their contributions to the Hessian elements for real atom coordinates.[a]

Add to …	Involving only X_{p1}	Involving both X_{p1} and X_{p2}
X_0X_0		
X_0X_{1a}	$(1-g_1)X_0X_{p1}$	
X_0X_{1b}	$g_1X_0X_{p1}$	
X_0X_{2a}		
X_0X_{2b}		
$X_{1a}X_{1a}$	$2(1-g_1)X_{1a}X_{p1} + (1-g_1)(1-g_1)X_{p1}X_{p1}$	
$X_{1a}X_{1b}$	$g_1X_{1a}X_{p1} + (1-g_1)X_{1b}X_{p1} + g_1(1-g_1)X_{p1}X_{p1}$	
$X_{1a}X_{2a}$	$(1-g_1)X_{2a}X_{p1}$	$(1-g_1)(1-g_2)X_{p1}X_{p2}$
$X_{1a}X_{2b}$	$(1-g_1)X_{2b}X_{p1}$	$(1-g_1)g_2X_{p1}X_{p2}$
$X_{1b}X_{1b}$	$2g_1X_{1b}X_{p1} + g_1g_1X_{p1}X_{p1}$	
$X_{1b}X_{2a}$	$g_1X_{2a}X_{p1}$	$g_1(1-g_2)X_{p1}X_{p2}$
$X_{1b}X_{2b}$	$g_1X_{2b}X_{p1}$	$g_1g_2X_{p1}X_{p2}$
$X_{2a}X_{2a}$		
$X_{2a}X_{2b}$		
$X_{2b}X_{2b}$		

[a] XY stands for $\partial^2/\partial X\partial Y$ or $\partial'^2/\partial X\partial Y$ (depending on if both X and Y are real coordinates or not), and $\partial^2/\partial X\partial Y = \partial^2/\partial Y\partial X$ (not shown). Only elements involving X_{p1} are shown. The elements involving X_{p2} can be derived by simultaneous interchange of g_1 and g_2, X_{p1} and X_{p2}, X_{1a} and X_{2a}, and X_{1b} and X_{2b}.

4.L.2. Fixed-bond-distance method

We derive only the first derivative using the fixed-bond-distance method. Just as for the scaled-bond-distance method, the energy derivatives with respect to the link atom are partitioned into the energy derivatives with respect to the M1 atom and the Q1 atom. The energy expression

for the boundary charge terms $\partial X_{HL}/\partial X_{Q1}$ and $\partial X_{HL}/\partial X_{M1}$ are show below. Following the same derivations as in Amber 10 (*163*), we have:

$$\partial X_{HL}/\partial X_{Q1} = (1 - d_{Q1\text{-}L}/|\mathbf{r}_{M1}\text{-}\mathbf{r}_{Q1}|)\, \boldsymbol{i} + d_{Q1\text{-}L}\,(X_{M1} - X_{Q1})\,/|\mathbf{r}_{M1}\text{-}\mathbf{r}_{Q1}|^3\,(\mathbf{r}_{M1}\text{-}\mathbf{r}_{Q1})$$

$$\partial X_{HL}/\partial X_{M1} = d_{Q1\text{-}L}/|\mathbf{r}_{M1}\text{-}\mathbf{r}_{Q1}|\, \boldsymbol{i} - d_{Q1\text{-}L}\,(X_{M1} - X_{Q1})\,/|\mathbf{r}_{M1}\text{-}\mathbf{r}_{Q1}|^3\,(\mathbf{r}_{M1}\text{-}\mathbf{r}_{Q1}) \qquad (4.L.20)$$

where $\boldsymbol{i}$ is the Cartesian unit vector along the *x* axis.

4.M. Dynamics

The time evolution of the system can be simulated with molecular dynamics simulation. Forces are obtained through the QM, MM, or QM/MM calculations. The velocity Verlet algorithm(*130*) is used to integrate the equations of motion. In *NVT* simulations, temperature is controlled with a Berendsen or a Nose-Hoover thermostat(*131*). The Nose-Hoover thermostat uses the chain coupling method with the chain length two. Both thermostats use a coupling parameter to indicate the strength of the bath coupling. The Nose- (for example, if we simulate a protein with 5000 atoms, then thermostat implementation uses the chain coupling method where the chain length is two; the coupling parameter τ, which governs an exponential decay of the system towards the desired temperature, is $\tau = (Q / N_{df} k_B T_0)^{1/2}$, where Q is the "mass" of the thermostat, N_{df} is the number of degrees of freedom of the system being simulated (e.g., if we simulate a proteon with 5000 stoms, then N_{df} is 15000), k_B is the Boltzmann constant, and T_0 is the target temperature. The smaller is τ, the stronger is the coupling, and the faster is the target temperature reached. The temperature can be set to a constant, or it can be changed according to a ramp with a constant slope for annealing simulations. Initial temperatures can be assigned via a random (Gaussian) distribution of speed for individual atoms. Water molecules can be constrained to be rigid molecules; the bond distances can be set to the values of the commonly used TIP3P(*132*) or SPC(*133*) water models. Constraints on bond distance are implemented with the RATTLE algorithm(*134*). In addition, other bonds with hydrogen can be constrained to the initial distance. We offer the flexibility that one can impose constraints on bonds with both QM and MM non-water hydrogen or only on bonds with MM non-water hydrogen (so that the QM hydrogen are not affected by the constraints). For adaptive-partitioning dynamics, the use of constraints only on MM hydrogen but not on QM hydrogen is not recommended, because it may cause numerical instability.

Periodic conditions with cubic boxes can be used in dynamics simulations. The box lengths in all three dimensions can be input directly by user or can be assigned by the program based on the atomic coordinates of the initial configuration. Because we cannot add infinite number of background point charges in the embedded-QM calculations, the embedded-QM calculations are carried on without periodic conditions – the system is translated such that the PS is always at the center of the primary cell, and the embedded-QM calculations are done with all MM point charges of the primary cell (unless cut-off has been used to limited the number of MM point charges in the embedded-QM calculations). In our implementation in QMMM, the subtractive definition of QM/MM energy is adopted. Thus, we view the difference between the QM and MM energies for the capped primary system (CPS) as a correction to the MM energy of the entire system. This means that the QM and MM calculations on the CPS must be consistent with each other in terms of periodic boundary conditions, but they do not have to the same as the MM calculations of the entire system in this regard. As a result, the CPS calculations do not use periodic boundary conditions (because the embedded QM calculations are based on a cluster model). The MM calculations for the entire system can be with either minimum image or Ewald sum.

4.N. Adaptive Partitioning

In some simulations, it is desired to have dynamical partitioning of QM and MM subsystem, i.e., on-the-fly reclassification of atoms/group into the QM and MM subsystems. Exchange of particles between the QM and MM subsystems may cause sudden changes in the energy and forces, leading to numerical instability and prevent correct sampling of configuration space in the desired ensemble. It is therefore critical to conserve energy and momentum in QM/MM dynamics simulations with adaptive partitioning.

A number of algorithms have been developed to handle the on-the-fly reclassification of atoms/group into the QM and MM subsystems.(*18, 34, 85, 135*) All methods rely on setting up a narrow buffer zone (also called switching shell) between the QM subsystem (also called active zone) and the MM subsystems (also called environmental zone). As illustrated in **Figure 3**, the active zone is usually defined as a sphere of the inner radius R_{min} centered at a given primary atom (or a given spatial location defined by the cartersian coordinates), and the buffer zone is defined as a shell within the inner and outer radii R_{min} and R_{max}.

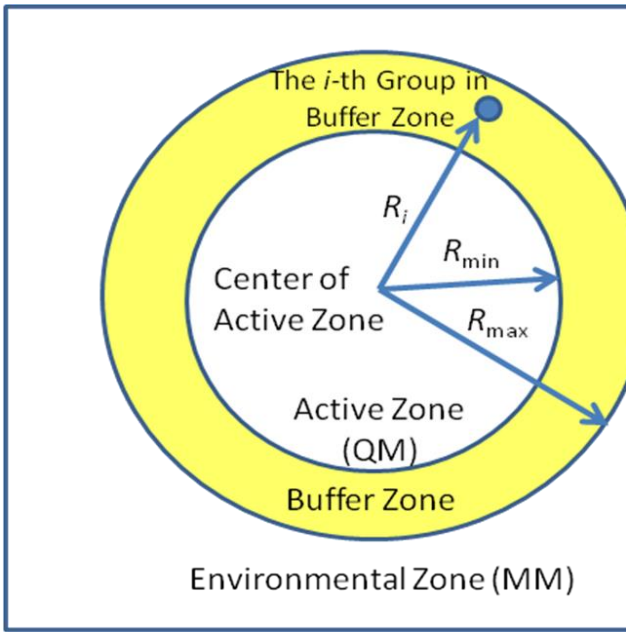


Figure 3. Illustration of the active zone, buffer zone, and environmental zone in the hot-spot, ONIOM-XS, and adaptive-partitioning (AP) schemes for on-the-fly reclassifications of atoms/groups into QM and MM subsystems in QM/MM dynamics simulations.

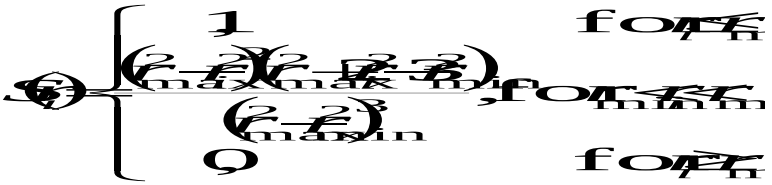
Various smoothing functions are applied to remove the discontinuity in the potential energy and/or forces when atoms/groups enter or leave the buffer zone. The smoothing functions depend on the distance *R* between the active-zone center and the center of mass or a delegate atom of the given atom/group in the buffer zone. We have implemented the hot-spot method,(*34*) the ONIOM-XS method,(*18*) and the sorted-AP and permuted-AP methods.(*85*) It seems that both the hot-spot and ONIOM-XS methods were originally designed for mechanical embedding QM/MM calculations; in the QMMM program, however, both methods can be invoked in the electronic embedding setup.

In the hot-spot method, (*34*) the gradient (and force) on the *i*-th group is calculated by:

(4.N.1)

(4.N.2)

where the superscript A+B denotes the calculations with the groups in the active zone and buffer zone together at the MM or QM level. The smoothing function $S(r_i)$ is as follows:



(4.N.3)

where r_{min} is the radius of PS (the same as $R_{\min}$ in **Figure 3**), r_{max} is the radius of the buffer zone (the same as $R_{\max}$ in **Figure 3**), and r_i is the distance of group *i* from the center of PS. The hot-spot method does not conserve momentum, however, because Newton's Third Law is not obeyed. Moreover, the energy is not defined or evaluated.

In ONIOM-XS**,**(*18*) the potential energy is defined by

$$V = PV^{A+B} + (1 - P)V^{A}$$

where V^{A+B} is the potential computed for the groups in the active zone and in the buffer zone together, V^{A} is the potential for the groups in the active zone only, and *P* is the smoothing function as follows:

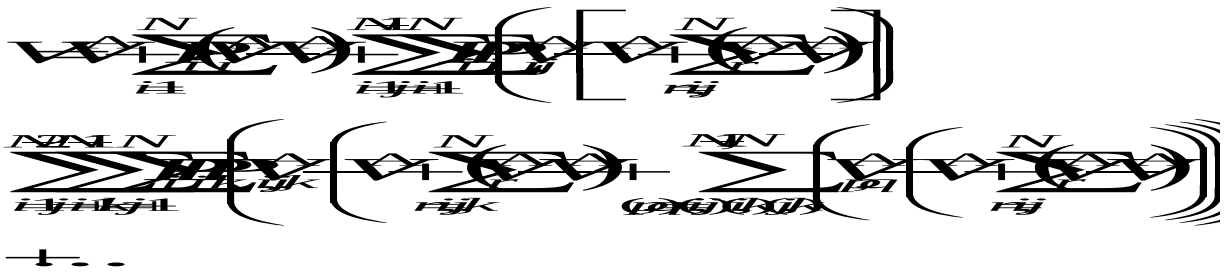
$$P = \frac{1}{N}\sum_{i=1}^{N} P_i(\alpha_i), \tag{4.N.4}$$

$$P_i(\alpha_i) = -6\alpha_i^5 + 15\alpha_i^4 - 10\alpha_i^3 + 1 \tag{4.N.5}$$

$$\alpha_i = (r_i - r_{\min})/(r_{\max} - r_{\min}) \tag{4.N.6}$$

Here, *N* is the number of groups in the buffer zone. Two QM calculations have to be performed in a given time step, and therefore the computational cost is twice as much as in the hot-spot method. The ONIOM-XS scheme does conserve momentum, but it does not conserve energy in microcanonical (*NVE*) simulations if two or more groups are present in the buffer zone.

In the permuted-AP scheme, the energy is expressed in many-body-expansion manner

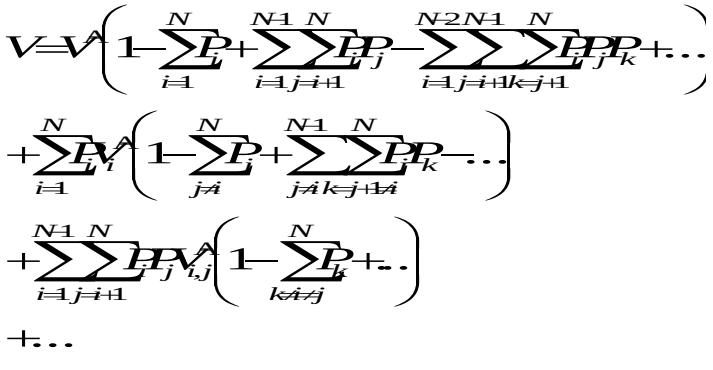


(4.N.7)

where V^{A} is the energy determined with the groups in the active zone at the QM level, V_i^{A} with all active-zone groups and the *i*-th buffer-zone group at the QM level, $V_{i,j}^{\mathrm{A}}$ with all active-zone groups, the *i*-th buffer-zone group, and the *j*-th buffer-zone group at the QM level, … $V_{1,2\ldots N}^{\mathrm{A}}$ with all active-zone groups and all buffer-zone groups at the QM level, and P_i is the smoothing function given by Equations (4.N.5) and (4.N.6). Equation (4.N.7) is Equation (10) in Ref.(*85*), which can be further reorganized to

(4.N.8)

or



(4.N.9)

In total, 2^N embedded-QM calculations are to be performed. All derivatives of the potential energy with respect to the coordinates vary smoothly up to the same order for which the smoothing functions P_i vary continuously.

As the energy contributions of the terms in the series in Equation (4.N.7) decreases rapidly, it may be advisable to truncate the series. The truncation significantly reduce the number of embedded-QM calculations, but it also introduce small (but controllable) discontinuities in the energy and derivatives. Our test calculations showed that the discontinuities are insignificant if the series is truncated at the 5th order, i.e. if only up to 5 groups in the buffer zone are included in the embedded-QM calculations and any terms with $P_iP_j\ldots P_k$ at the 5th or higher order are omitted. The omission of any terms with $P_iP_j\ldots P_k$ at the 5th or higher order is necessary because the sum of the smoothing functions must be 1. The truncation at the 5th order is the default option for the QMMM computer program, but users can override this option by specifying the order (up to 13) at which the expansion will be truncated. The actual number of embedded-QM calculations is determined by the sum of the binomial coefficient for the number of buffer-zone groups up to the specified order. The QMMM program is currently limited to a maximum number of groups in the buffer that can be handled, which is 64, and to a maximum number of permutations that can be treated, which is 10,000. (See also Section 4.Q.7. Limitations of the Program.) The program will not continue if there are too many groups in the buffer zone or if the number of calculations due to the permutation up to the specified order is larger than the preset number. However, the current value for the maximum number of buffer-zone groups and the maximum number of permutation should be sufficient for most applications.

The QMMM program also has a "temporarily reducing order" option to offer some flexibility in the order truncation. This option, which is invoked by the keyword **REDUCEORDER,** is mainly for expert users or for debugging purpose. This option allows the user to temporarily reduce the order of truncation if the actual number of embedded-QM calculations is larger than the maximum number of embedded-QM calculations for a time step. The QMMM program will print a warning when the order is temporarily reduced. The procedure of temporarily reducing order for truncation is shown in **Figure 4**.

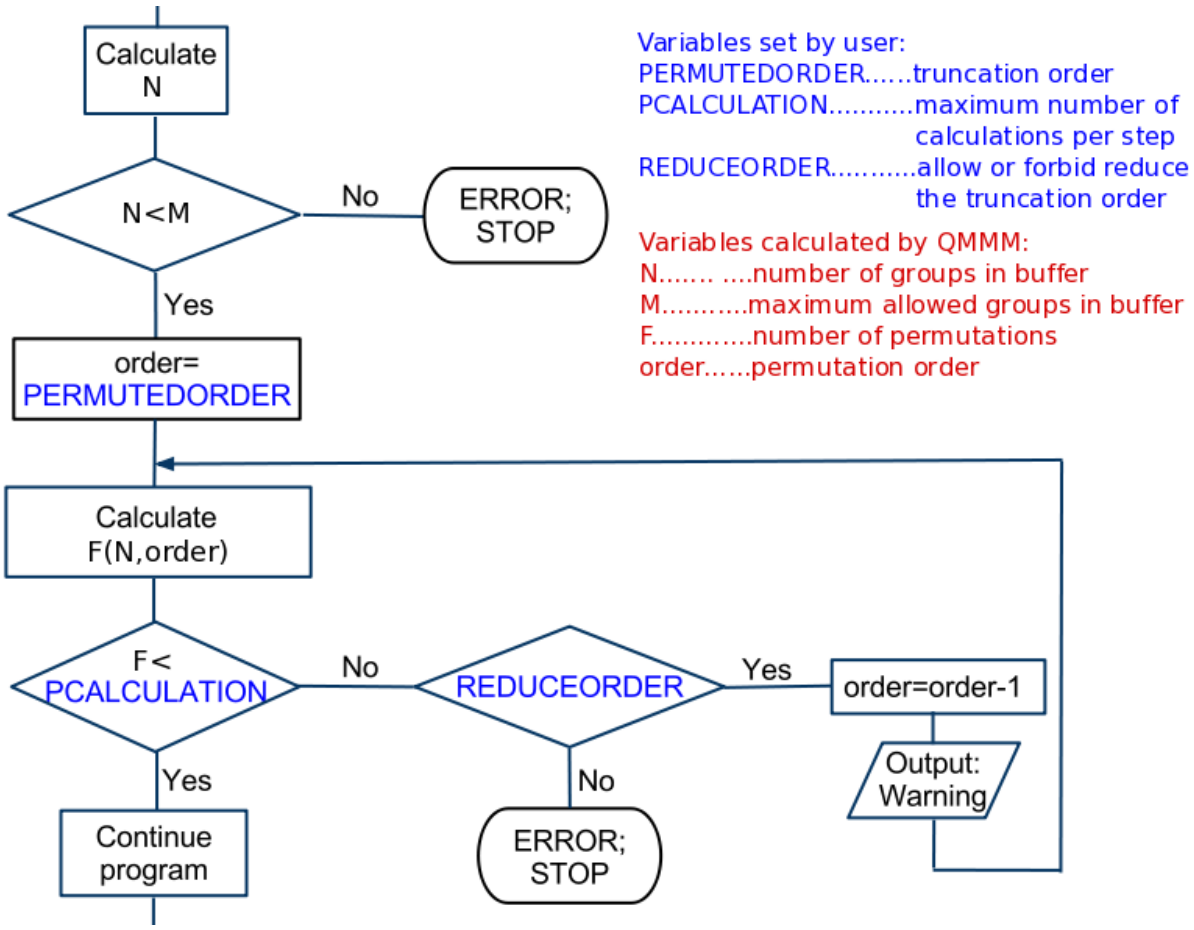


Figure 4. Procedure of temporarily reducing order of truncation in permuted AP.

In sorted AP,(*85*) the groups in the buffer zone are sorted from the closest to the PS to the farthest. The embedded-QM calculations begins with the active zone only, and the buffer-zone groups are added one at a time according to the increasing distance of the group to the active zone, leading to in total *N*+1 QM calculations. The potential energy in sorted AP is given by

(4.N.10)

with

$$\Phi_i = (1 - \chi_i)^{-3},$$

and

(4.N.11)

The chosen smoothing function ensures that the energy and gradient stay constant when two groups in the buffer change the rank.

In our implementation, a group can be a single atom, a molecule, a fragment of a molecule, or a combination of all of them. To produce a list of groups for use in QMMM , we have developed a program called `createGroups`. This program can create a group list in the file `group.log` based on the covalent bonding patterns. By default, all atoms connected via covalent bonds are put into one group. Another option is to define each amino acid residue as one group. The third option is to further divide the backbone and the side chain of the amino acid into two groups. Although it is often to set the boundary between amino acids through the backbone C–N bond, our program sets by default the boundary through the C_α–N bond, following the recommendation by Ref.(*136*) for avoiding destroy of the double-bond character of the C–N bond. Users can always override this setup by manually modifying the `group.log` file.

For the AP methods, there is a problem associated with the zero of energy for each group. The absolute energies given by the QM calculations are several orders of magnitude larger than the MM counterparts; properly setting the zeros of energy to be subtracted is therefore critical to the calculations of energies and forces in the simulations. The zero of QM (or MM) energy for a group that is a whole molecule is straightforward to obtain, which we set to the QM (or MM) energy of the isolated molecule at its geometry optimized at the given QM level of theory (or MM force field). For a group that is part of a molecule, the situation is more complicate, and the zero of energy depends on how this group is linked to other groups of the molecule. This can be illustrated by the example in **Figure 5**, which is a complex formed by a butanol molecule and two water molecules.

As shown in **Figure 5 (a)**, each water molecule is a group, whose zero of energy is the energy for an isolated water molecule at its optimized geometry. The butanol molecule is divided into three groups: Group 1 is $–CH_2OH$, Group 2 is $–CH_2CH_2–$, and Group 3 is $CH_3–$. Suppose that the active-zone center is the O atom of Group 1, which is sometimes called the primary atom. In **Figure 5 (b)**, Groups 1 and 4 are in the active zone. In **Figure 5 (c)**, Groups 1 and 2 are in the active zone. Consequently, the zero of energy of Group 1 is the energy of the capped-PS, CH3OH, at its optimized geometry:

$$E_0(\text{Group 1}) = E(CH_3OH) \qquad (4.N.12)$$

Notice that when Groups 1 and 2 are in the active zone, they merge to produce a "super-group" – $CH_2CH_2CH_2OH$. As a result, the zero of energy for Group 2 is defined as the energy difference between the capped super-group and the capped Group 1:

$$E_0(\text{Group 2}) = E(CH_3CH_2CH_2OH) - E(CH_3OH) \quad (4.N.13)$$

Similarly, the zero of energy for Group 3 is

$$E_0(\text{Group 3}) = E(CH_3CH_2CH_2CH_2OH) - E(CH_3CH_2CH_2OH) \quad (4.N.14)$$

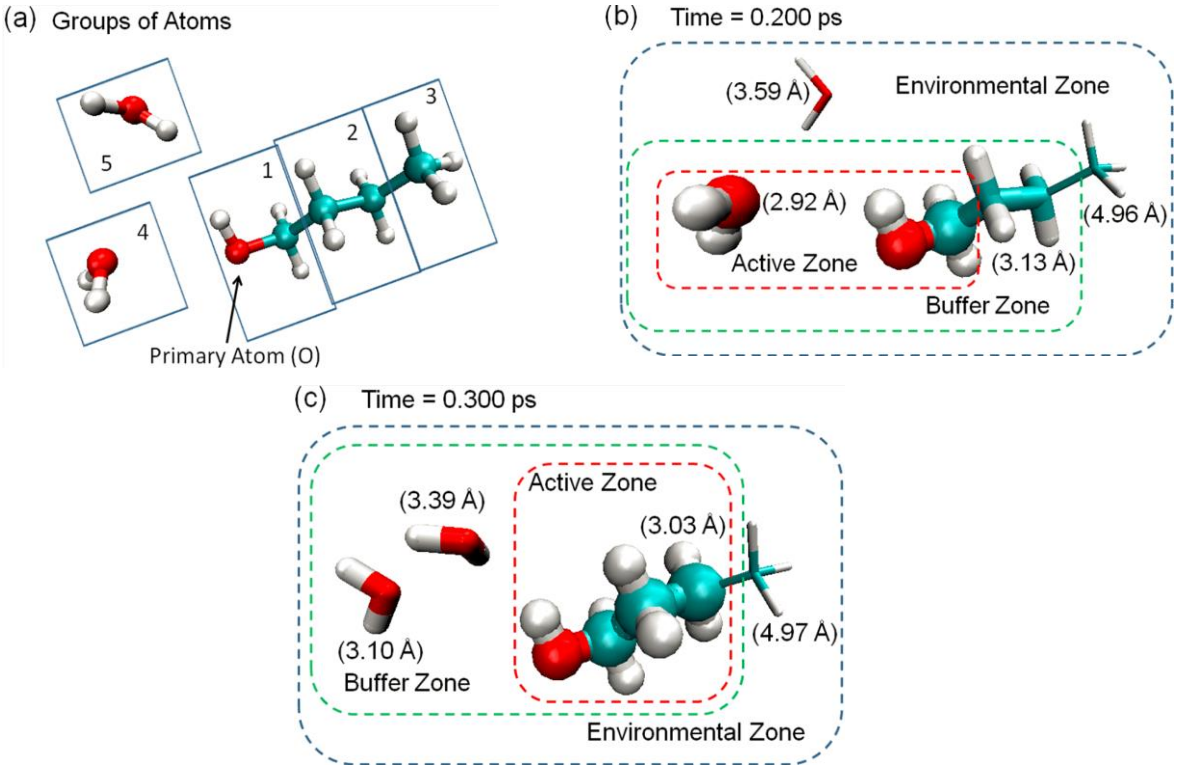


Figure 5. A butanol molecule in complex with two water molecules. (a) The setup of groups. (b) A snapshot showing Groups 1 and 4 in the active zone. (c) A snapshot showing Groups 1 and 2 in the active zone. The distance between a group and the primary atom is given in the parentheses except for Group 1, for which the distances are 0.66 Å in both snapshots.

Apparently, the zero of energy of a fragmental group must be obtained in accord to how the group is merged with other groups in the active zone and buffer zone. For example, if the C atom in Group 3 is set to be the primary atom of the QM subsystem, the zero of energy for Groups 1 to 3 will be given by

$$E_0(\text{Group 1}) = E(CH_3CH_2CH_2CH_2OH) - E(CH_3CH_2CH_3) \quad (4.N.15)$$

$$E_0(\text{Group 2}) = E(CH_3CH_2CH_3) - E(CH_4) \quad (4.N.16)$$

$$E_0(\text{Group 3}) = E(CH_4) \quad (4.N.17)$$

User must supply the definition of groups and their zeros of energy in the input file `group.log`. The QMMM program will automatically relocate the boundary and find the correct zero of energy based on the topology of the system. Currently, the program can only handle the group with one zero of energy.

When determining the zero of energy for a fragmental group, it was not necessary to include groups that were present in the active zone and/or buffer zone but did not covalently connect to the fragmental group. Inclusion of those groups changed the zero of energy slightly, but seemed to have negligible effects on the energy and momentum conservations in the MD simulations.

The total zero of energy for the whole system, $E_0(\text{sys})$, is the sum of the zeros of energy for all groups, according to Equation (4.N.7) for the permuted-AP method and Equation (4.N.10) for the sorted-AP method. A group in the buffer zone has dual (QM and MM) characteristics, so its contribution to the total zero of energy varies when its distance to the active-zone center changes. As a result, the total zero of energy for the whole system can change significantly (a few to a few hundreds of hartree, depending on the system) and rapidly (in a few tens of femtoseconds) during simulations, presenting a challenge in maintaining numerical precision and stability in long-time simulations, especially *NVE* simulations. (Note that the gradients due to the smoothing functions depend on the difference between the QM and MM energies.) Such drastic variations in the zero of energy were not present in previous dual-level MM simulations,(*85, 135*) since the zeros of MM energy of a group such as a water molecule are usually rather small. The numerical stability also relies critically on the availability of highly accurate gradients. Therefore, for QM calculation, tight SCF convergence is desired. In particular, for density functional theory (DFT) calculations, fine grids for numerical integration are strongly recommended.

4.O. Units

The programs called by QMMM are using different units. For example, energies in QM calculation are usually given in hartree energy, while in MM simulations kcal/mol is more common. In the QMMM program, we convert the quantities given in the QMMM input as well as in the output produced by the invoked programs into the following units: distance in Angstrom, mass in atomic mass unit, time in femtosecond, energy in hartree, and charge in electron charge,

and all other units are derived from these units. The QMMM output will be using the above-mentioned units, if not otherwise explicitly indicated.

4.P. Special Notes on Implementation

4.P.1. Calling GAMESS

Currently, only mechanical embedding scheme is supported for QM/MM calculations with GAMESS as the QM package. It is a pity that GAMESS does not support gradient calculations with the background point charges, and actually, GAMESS discourages users to do calculations with background point charges. Therefore, QM/MM calculation with electronic embedding is not implemented in the current version of QMMM if GAMESS is selected to be the QM package.

The basis set specification in GAMESS is quite complicated and we have adopted a convention to keep the QMMM input in consistent with the GAMESS input format as much as possible. The value of BASIS keyword (in the QMKEY list) is ignored, and the basis sets are actually specified in the OPTION list. An example of doing a calculation using the 6-31G basis set is as follow:

```
QMKEY
        Basis 6-31g
    Options
            ! $basis gbasis=n31 ngauss=6 $end
        End
End
```

Here, the *6-31g* following *Basis* is a comment and will be ignored by the QMMM program. However, we suggest user to keep this comment, because it helps people to recognize the basis set. The line

```
! $basis gbasis=n31 ngauss=6 $end
```

listed as the options are the actual specification of the basis set in the GAMESS format, except that "*!*" given at the very beginning of this line; the "*!*" is an indicator of GAMESS keywords.

The QM method specification in GAMESS is even more complicated. We have tried hard to simplify the method specification so that it is in consistent with QMMM input format as much as possible. Below, we list the QM methods in GAMESS supported by QMMM and the corresponding values to be given for the METHOD keyword (in the QMKEY list): To use, user can just specify the value for the METHOD keyword and thus avoid the $CONTROL, $DFT, or $BASIS groups in GAMESS.

1) The SCF type calculations (those specified in the SCFTYP keyword in the $CONTRL group of GAMESS):

RHF UHF ROHF GVB MCSCF

2) MP2 calculation (in GAMESS, one needs to give the value "2" of the MPLEVL keyword in the $CONTRL group):
MP2

3) The CI calculations (those specified in the CITYP keyword in the $CONTRL group of GAMESS):

CIS ALDET ORMAS FSOCI GENCI
GUGA

4) The coupled-cluster (CC) calculations (those specified in the CCTYP keyword in the $CONTRL group of GAMESS):

LCCD CCD CCSD CCSD(T) CR-CC
R-CC CR-CCL CCSD(TQ) EOM-CCSD CR-CC(Q)

CR-EOM

5) The DFT calculations (those specified in the DFTTYP keyword in the $DFT group of GAMESS):

SLATER	BECKE	GILL	PBE	VWN	LYP
OP	SVWN	SLYP	SOP	GLYP	GVWN
OP	PBEVWN	PBELYP	PBEOP	B3LYP	BHHLYP
BVWN	B3LYP	BOP	XALPHA	DEPRISTO	
CAMA	BALF	PWLOC	BPWLOC	GAMB	XVWN
XPWLOC	SPWLOC	WIGNER	WS	WIGEXP	

6) The semi-empirical calculation (those specified in the GBASIS keyword in the $BASIS group of GAMESS):

MNDO AM1 PM3

4.P.2. Calling ***Gaussian***

In principle, if energy calculations are possible, all energy derivatives can be calculated numerically. In this sense, any one of the existing electronic structure packages that can perform the QM calculation in the presence of a distribution of background point charge can be used by QMMM program to do QM calculations. However, the program will be much more efficient if analytic gradients and/or Hessians are available, especially for the background point charges. This feature is however not available in most of the electronic structure packages including the popular *Gaussian* package.

To solve this problem, the QMMM program uses a “trick” when carrying out QM calculations by calling *Gaussian*. The QMMM program prepares an ONIOM calculation input, where all the background point charges are treated as He atoms. The AMBER force field is adopted, the atom types for all PS atoms are set to CT with MM charges set to 0, and atom types for HL and background point charges are set to HC. Moreover, in this ONIOM input file, the MM non-bonded van der Waals interactions are suppressed by adding the keywords “*amber=softfirst*” and

"*NonBon 0 1*". Since the MM charges on PS atoms are 0, the PS atoms are not involved in MM coulomb interactions. The coulomb self-energy among the background point charges is computed in ONIOM twice: first in the MM calculations for the ES and again in the electrostatic-embedded MM calculations for the PS; this leads to an exact cancellation. Next, except for selected stretches (the Q1–HL bonds), all bonded interactions are suppressed by removing the bond-to parameters in the connectivity list. Since there is no covalent bond is cut, the two MM covalent calculations in ONIOM calculations canceled exactly. Because both the bonded and nonbonded interactions cancel exactly, the final energy, gradient, and Hessian are for the term *E*(QM;CPS**) in Equation (4.E.1).

There are two additional MM calculations in the ONIOM calculation described above; but the increased computational cost is negligible compared with QM calculations. If a future version of the *Gaussian* electronic structure package would give analytic gradients and/or Hessians for the background point charges, we will be able to avoid using the "trick". In this regard, we note that ORCA has analytic gradients for the background point charges, and thus it is straightforward to invoke ORCA for embedded QM calculations.

The *E*(QM;CPS**) computed via the ONIOM trick includes the coulomb self-energy among the background point charges. This energy differs from the electrostatic energy for the SS and should be subtracted to obtain the real *E*(QM;CPS**). The QMMM program thus carries out an additional MM calculation by use of TINKER to evaluate these interactions so that they can be subtracted.

Concerning the *external* option, through which the *Gaussian* optimizer can be invoked to do geometry optimization, we found that different versions of *Gaussian* (*g03.b01, g03.c01, g03.d01, g03.e01, g09.a02, g09.e01, g16.a03, g16.b01,* and *g16.c01*) require slightly different procedures to invoke the *external* option. More specifically, the input and output files required for the *g03.d01, g03.e01, g09.a02, g09.e01, g16.a03, g16.b01,* and *g16.c01* versions are in the scratch directory, while those of the *g03.c01* and *g03.b01* versions are in the directory where the *Gaussian* input files are located. Thus, the program includes different shuttle scripts for calling different versions of *Gaussian*. The user can select the approriate script for his or her calculations. See also Section 8.A. Installation Instructions for more details.

A special note about the use of *Gaussian 09* and *Gaussian 16* is that we have made *QMMM* compatible with *Gaussian09.a02, Gaussian09.e01, Gaussian16.a03, Gaussian16.b01* and

Gaussian16.c01 for most QM/MM calculations. However, we found that the default settings for external optimization in *Gaussian 09* and *Gaussian16* are different from those in *Gaussian 03*, and one may get slightly different results when using *Gaussian 03*, *Gaussian 09*, and *Gaussian 16*. Therefore, for the test runs in the package, *Gaussian03.e01* is used for test2001-test2060, *Gaussian09.e01* is used for test2061-test2067 and test2070, *Gaussian16.b01* is used for test2068 and test2069.

4.P.3. Calling ORCA

The current version of QMMM may also call the electronic structure package ORCA to calculate energies, gradients, and Hessians, respectively. Currently, QM/MM single-point energy is available for HF, DFT, and MP2 as the QM. The QM/MM single-point gradients are available in the HF and DFT cases, for which ORCA provides analytic gradients on both the QM atoms and the background point charges.

The QM/MM single-point Hessians are available for HF and DFT as the QM methods with mechanical embedding. Currently, ORCA provides numerical Hessian on the QM atoms, but does not provide Hessian for the background point charges. Therefore, QM/MM Hessian with electric embedding is not calculated.

To be consistent with the ORCA input format, when a DFT method is selected as the QM method, the METHOD keyword (in the QMKEY list) must be set to *dft*, and the functional is specified in the OPTION list as follow:

```
QMKEY
        Method    dft
    Options
            >! b3lyp
        End
End
```

or

```
QMKEY
        Method    dft
    Options
```

>%method functional B3lyp

> end

End

End

The format of the input functional is the same as ORCA, except that each line begins with ">", which is an indicator of ORCA keywords. The reason of using this indicator is that ORCA also uses END as keyword as QMMM does. The indicator ">" in the above example makes these two END keywords distinguishable.

We also note that in ORCA calculations, the self-energy of the background point charges is not calculated while some other electronic-structure packages like *Gaussian* do calculate it. Therefore, a different (actually simpler) treatment is employed to work out the QM/MM energy for ORCA than for *Gaussian*.

4.P.4. Calling TINKER

The current version of QMMM calls the TINKER executables `analyze`, `testgrad`, and `testhess` to calculate energies, gradients, and Hessians, respectively. Furthermore, the `analyze` routine is called to get MM parameters such as MM point charges, and another executable `Newton` is invoked for pre-optimization at the MM level if required.

In principle, modifications to the TINKER program are not needed. However, we suggest that users make a small modification to the output formats for the gradient in the `testgrad` subroutine and for the Hessian elements in the `testhess` subroutine. The current output format used by TINKER in the `testgrad` subroutine is (*3F16.8*) if the *digits = 8* keyword is specified, (*3F14.6*) if *digits = 6*, and (*3F12.4*) if *digits = 4* is specified. Our recommendation is to change the formats to (*3(E16.8,1X)*), (*3(E14.6,1X)*), and (*3(E12.4,1X)*), respectively. The use of scientific format helps to handle special cases where the gradient and Hessian elements are very large, and the insertion of a space between the numbers makes the output more readable. Similarly, we recommend that users change the output format used by TINKER in the `testhess` subroutine from *(4F16.8)* to *(4(E16.8,1X))* if the *digits = 8* keyword is specified, from *(5F14.6)* to *(5(E14.6,1X))* if the *digits = 6* keyword is specified, and from *(6F12.4)* to *(6(E12.4,1X))* if the *digits = 4* keyword is specified. [We have provide users in the `script` directory two scripts

(`modtinker` and `pswitch`) to make such a modification. Users can simply copy these two scripts into the same directory where TINKER source files are placed, and run the `modtinker` script.]

We have tested the current version of QMMM with five versions of TINKER: version 3.5,[1] version 4.1, version 4.2, version 5.1, and version 6.3, and the test runs in the current version of QMMM are made to call modified TINKER 6.3 for all QM/MM calculations with *Gaussian* as the QM package (test2001-test2068) and to call TINKER 4.2 for other calculations. Modified TINKER 6.3 is included in the distribution package in qmmm2018/tinker_QMMM/. It is possible (and very straightforward) to make QMMM calls other versions of TINKER without modifying the QMMM code, provided the input and output formats used by the other version of TINKER are the same as TINKER 6.3. At the TINKER web site (access date: March 13, 2015), there is a statement that "Please note that as with prior new releases, version 7 is neither backward nor forward compatible with earlier versions of TINKER. In particular, older versions of parameter files should not be used with TINKER 7 executables and vice versa." Although we found only small changes in versions 4.1, 4.2, 5.1, and 6.3, it is possibible that there may be difficulties when using other versions of TINKER. In the distribution folder, we include the modified TINKER 5.1 and TINKER 6.3 for use by users of QMMM. These modified versions of TINKER include several modified subroutines of TINKER for the screened charge and smeared charge models.

There are two things we want to point out concerning the differences between different versions of TINKER: (1) The default Coulomb constant has been updated from 332.0538 in TINKER 4.2 to 332.0637 in TINKER 6.3, and this will make a small difference (usually $< 10^{-5}$ hartree) between results from TINKER 4.2 and TINKER 6.3. (2) It is also found that part of the MM3 force field parameter file has been changed from TINKER 4.2 to TINKER 5.1 and TINKER 6.3. Although this may not cause problems in most cases (this can cause problems for test2033 and test2034), it is suggested to use the parameter files in TINKER 6.3 if one want to use TINKER 6.3 as the MM package. If one want to use a different version of TINKER 6.3 as the MM package, it is suggested to verify whether there have indeed some changes to the input and output formats by running new TINKER calculations with old input files and comparing the outputs. The only thing that one needs to do is to change one line in the shuttle script for running TINKER, in particular

[1] For TINKER 3.5, we have so far only tested the mechanical embedding scheme using MM3 force field.

the line that specifies the directory where the TINKER executables is located. (See also section 7.B.2. Program Shuttle Scripts.)

4.P.5. Atom Index

Another important note concerns the list of the atoms for each of the single-level calculations. Each atom is labeled by an atom index, but a different set of indices is used at different stages of the calculation.

The atom index for the real system is from 1 to N, where N is the total number of atoms.

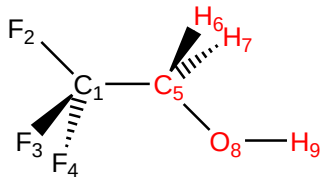
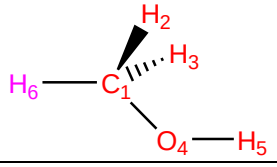
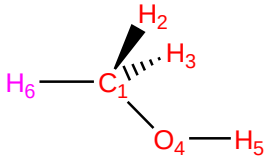
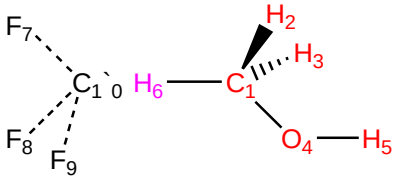
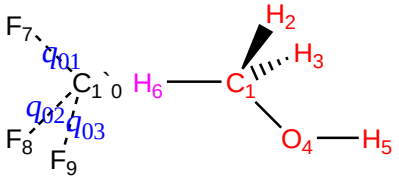
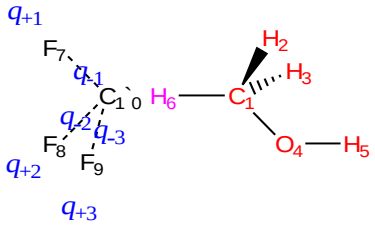
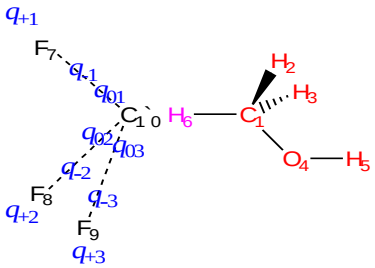
The CPS system in the ME scheme and the "embedded-CPS" system in all electronic embedding schemes, both generated by QMMM, have a different scheme for labeling the atoms from the real system. For the CPS system in the ME scheme, the index for QM atoms ranges from 1 to N_{QM}, where N_{QM} is the number of QM atoms. Next come the HL atoms, whose indices are from N_{QM} + 1 to N_{QM} + N_{HL}, where N_{HL} is the number of HL atoms. [Currently, we allow each M1 atom to bond to one Q1 atom, and thus the number of HL atoms is equal to the number of M1 atoms (N_{M1}).]

For the "embedded-CPS" system in all electronic embedding schemes, which contains not only the CPS atoms but also all the background point charges including auxiliary charges q_0, q_-, and q_+, the CPS atoms are listed first, in the same order as that we used for the CPS system in the ME scheme. After the CPS atoms come all the point charges for MM atoms excluding M1 atoms, with the index ranging from N_{QM} + N_{HL}+ 1 to N. Finally, depending which electronic embedding scheme is chosen, either the charges for M1 atoms or auxiliary point charges will be added to the index list as follows.

For the SEE, Z1, Z2, and Z3 schemes, the charges for M1 atoms are added to the list. For the RC and RCD schemes, the redistributed charges q_0 are added. For the Shift scheme, the auxiliary point charge pairs (q_- and q_+) are added, i.e., q_{-1}, q_{+1}, q_{-2}, $q_{+2, \ldots,}$ q_{-M}, and q_{+M}, where M is the number of pairs. For the RCD scheme, both the the redistributed charges q_0 and auxiliary point charge pairs (q_- and q_+) are added: q_{01}, q_{-1}, q_{+1}, q_{02}, q_{-2}, $q_{+2, \ldots,}$ $q_{0\mathrm{M}}$, q_{-M}, and q_{+M}.

The scheme for labeling atoms and point charges is illustrated by the example CF_3-CH_2OH as in Table 4.P.5.1.

Table 4.P.5.1. List of atoms and background point charges (including auxiliary point charges) for CF_3-CH_2OH in QM/MM calculations.

Entire (real) system • QM atoms: C5, H6, H7, O9, H9 • MM atoms: C1, F2, F3, F4 In particular, C1 is an M1 atom, F2, F3, and F4 are M2 atoms.	
Capped primary system • QM atoms: C1, H2, H3, O4, H5 • Cap atom: H6	
Large system (CPS + background charges)	
1. The ME scheme. No background charges at all.	
2. Scaled/eliminated charge schemes, e.g., the SEE, Z1, Z2, and Z3 schemes. No auxiliary charge.	
3. The RC and RCD schemes. Indices for auxiliary charges: $q_{01}(11)$, $q_{02}(12)$, $q_{03}(13)$	
4. The Shift scheme. Indices for auxiliary charges: $q_{-1}(11)$, $q_{+1}(12)$, $q_{-2}(13)$, $q_{+2}(14)$, $q_{-3}(15)$, $q_{+3}(16)$	
5. The RCD2 scheme Indices for auxiliary charges: $q_{01}(11)$, $q_{-1}(12)$, $q_{+1}(13)$, $q_{02}(14)$, $q_{-2}(15)$, $q_{+2}(16)$, $q_{03}(17)$, $q_{-3}(18)$, $q_{+3}(19)$	

A comparison of the indices for the real atoms in the entire system and in the large (i.e., CPS + background charges) system is given in Table 4.P.5.2.

Table 4.P.5.2. Indices for the real atoms in the entire system and in the large (i.e., CPS + background charges) system for CF_3-CH_2OH in QM/MM calculations.

Entire (real) system	Index in the large system				
	ME	SEE, Z1, Z2, and Z3 [a]	RC and RCD	Shift	RCD2
C5 (QM)	1	1	1	1	1
H6 (QM)	2	2	2	2	2
H7 (QM)	3	3	3	3	3
H8 (QM)	4	4	4	4	4
H9 (QM)	5	5	5	5	5
	6 (HL)	6 (HL)	6 (HL)	6 (HL)	6 (HL)
F2 (MM)		7	7	7	7
F3 (MM)		8	8	8	8
F4 (MM)		9	9	9	9
C1 (MM)		10	10	10	10
			11 (q_{01})	11 (q_{-1})	11 (q_{01})
			12 (q_{02})	12 (q_{+1})	12 (q_{-1})
			13 (q_{03})	13 (q_{-1})	13 (q_{+1})
				14 (q_{+2})	14 (q_{02})
				15 (q_{-3})	15 (q_{-2})
				16 (q_{+3})	16 (q_{+2})
					17 (q_{03})
					18 (q_{-3})
					19 (q_{+3})

[a] And all scaled/eliminated charge schemes.

A PERL script `QMMMatomList.pl` is provided in the subdirectory `scripts`; this script can help users to find the index of each atom in the old list for the entire system and in the new list for the large system.

4.Q. Special Notes on Applications

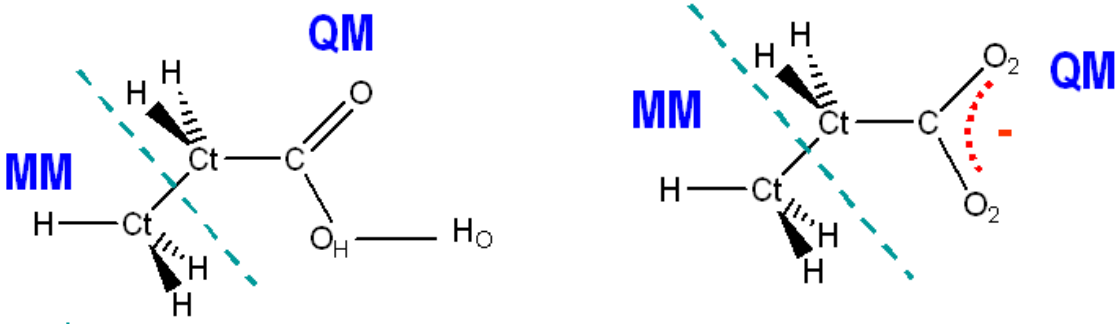
4.Q.1. MM Parameters for the PS

The QM/MM schemes implemented in QMMM program are designed to be applicable with any MM method that employs atom-centered partial charges. Some QM/MM methods, such as the GHO method and the pseudobond method, require new parameters for boundary atoms, integral scaling factors in the QM calculations, or specially parameterized ECPs. Such parameters usually require reconsideration if one switches MM scheme (e.g., from CHARMM(*108*) to OPLS-AA(*109, 125-129*)), QM scheme (e.g., from semi-empirical molecular orbital methods to density functional theory or post-Hartree-Fock *ab initio* methods), or QM basis set. A central objective in our method development is to avoid introducing any new parameters. Thus, for example, no MM parameters are changed, no integrals are scaled, and the link atom is an ordinary hydrogen atom with a standard basis set.

The key issue discussed in this section is how to select MM parameters for the atoms in the PS. As discussed in Section 4.I, we do not need the partial charges of PS atoms, but we do need parameters for stretching vibrations of the Q1 atoms, parameters for bending vibrations of Q1 and Q2 atoms, torsion parameters for Q1, Q2, and Q3 atoms, and van der Waals parameters for all Q atoms. This presents a problem since reaction is allowed to occur in the PS, and therefore the atom types of the Q atoms are not uniquely defined. An example is the deprotonation of RCH_2COOH to form RCH_2COO^-, for which the R group is the SS, and the CH_2COOH subunit is the PS. The COOH group becomes a COO^- group upon deprotonation; therefore, the atom types for the Q2 carbon atom and the Q3 oxygen atoms are different at different points along the reaction path. Which set of MM parameters should we use for Q2 and Q3 atoms in M1−Q1−Q2 bends, M1−Q1−Q2−Q3 and M2−M1−Q1−Q2 torsions, and in van der Waals interactions of Q2 and Q3 when carrying out molecular dynamics calculations or following the reaction path, those for the protonated form or those for the deprotonated form? Switching between these two sets of parameters during a dynamics calculation or along the reaction path is not convenient. Moreover, even if the switching between parameters could be done, one does not know at which point along the reaction path it should be done. There is no unambiguous answer; the decision that we make in Ref.[82] is to use the MM parameters for the protonated form, even for calculations on the

deprotonated reagent. Although our treatment is not a perfect solution, it is very practical, and it appears to be reasonable as discussed next for the protonation of RCH_2COO^-.

First consider the valence interactions, in particular, those for the Q1, Q2, and Q3 atoms, since, as seen above, certain valence interactions involving these atoms do not cancel. In principle, this can be avoided if one uses a larger QM subsystem, such that the atoms types for the Q1, Q2, and Q3 atoms do not change. However, a larger QM subsystem is not always feasible, e.g., in the RCH_2COOH case where R is a naphthyl group; in particular, we show the example for R = CH_3 .



Protonated form Deprotonated form

Generally speaking, the Q1–M1 stretch is the most important interaction among those surviving valence interactions due to its large force constant; the Q2–Q1–M1 and Q1–M1–M2 bends are less significant, and the Q3–Q2–Q1–M1, Q2–Q1–M1–M2, and Q1–M1–M2–M3 torsions are the least critical. Fortunately, the Q1 atom type does not change in this case (and in most applications), thus the Q1–M1 stretch, the Q1–M1–M2 bend, and the Q1–M1–M2–M3 torsion are unambiguous. The parameters for the other bends and torsions often remain the same or change just slightly. The OPLS-AA(*109, 125-129*) force field (in the TINKER(*116*) implementation that we use in this work) uses the same parameters for CH_3CH_2COOH and $CH_3CH_2COO^-$ for the Q2–Q1–M1 bend and for the Q2–Q1–M1–M2 torsion. There are two kinds of Q3–Q2–Q1–M1 torsion in CH_3CH_2COOH: (a) the O–C–C–C torsion where the O bonds to the H atom and (b) the O=C–C–C torsion with a double bond between the O and C atoms. There is only one kind of Q3–Q2–Q1–M1 torsion in $CH_3CH_2COO^-$, the O–C–C–C torsion. The (a) torsion in CH_3CH_2COOH also uses the same parameters as the O–C–C–C

torsion in $CH_3CH_2COO^-$, and only the (b) torsion uses a different one. Due to the very small force constants (the torsional barrier height is less than 0.9 kcal/mol) for all Q3–Q2–Q1–M1 torsions, using a single set of valence parameters along the reaction path does not seem to produce unacceptably large uncertainty in comparison with the errors produced by other approximations that are introduced into the QM/MM framework.

Next, we examine the non-bonded interactions. For the van der Waals interactions, any PS atoms that change atom types are ambiguous, and in principle, this problem cannot be avoided even if a larger QM subsystem is adopted. Fortunately, in practice it is not a serious problem, since the van der Waals interactions are significant only at short distances, and the use of only one set of van der Waals parameters is often adequate.

Turning to the electrostatic interactions, this is not a problem at all. In our RC and RCD schemes, as well as all other electronic embedding schemes implemented in QMMM, the electrostatic contributions to *E*(MM;ES) and *E*(MM;CPS*) that involve PS charges cancel exactly, and they do not need to be evaluated.

4.Q.2. MM Point Charges for the SS

In the electronic embedding schemes, the CPS is polarized by the background point charges, i.e., the appropriately modified MM point charges of the SS. The values of the MM point charge parameters of the SS plays a critical role in perturbing the electronic structure of the CPS. Usually one takes whole set of MM point charges, van der Waals parameters, and valence interaction parameters from the same force field, which is convenient. This is often a good choice, and sometimes it is the only choice.

However, such a choice for MM point charges does not seem to be a good choice when one studies a system in the gas phase. The investigation of model systems in the gas phase is not uncommon, e.g., when one performs validation tests for QM/MM methods. The use of gas-phase models is understandable, since it is not practical to employ an extensive training/testing set of examples corresponding to liquid solution. The complication is that the point charges in many MM force fields, such as CHARMM(*108*) and OPLS-AA,(*109, 125-129*) are designed for simulations in condensed phases, and strictly speaking, they are not suitable for validation tests in the gas phase. One quite simple but rather important case is the alkyl groups. In CHARMM and OPLS-AA, the MM charges for atoms in an alkyl group are quite large, e.g., the C atom in a

CH_2 group is assigned a charge of –0.18 e (CHARMM) and –0.12 e (OPLS-AA), respectively. (We notice that a recent re-parameterization(*137*) of the OPLS-AA force field suggests reducing the original OPLS-AA charges for alkanes by 25% of their magnitude for improved simulations in liquids.) The gas-phase system is expected to be less polar. Unfortunately, we do not know what the accurate partial charges are in the gas phase (in fact the concept of partial atomic charge is an approximate one so there is no unique answer). The electrostatic potential (ESP)(*138, 139*) fitted charges seem to be candidates for reasonable charges, but the ESP fitting procedure can be problematic for systems with buried atoms,(*59, 140, 141*) although it is sometimes stable for very small compounds. The ESP charges computed from gas-phase molecules at least have the advantage that they are not parameterized for the liquid phase. The very small gas-phase ESP charges on the alkyl groups(*82*) do imply that the alkanes are very nonpolar in gas phase, a point that is further supported by our recent tests on proton affinities employing different QM/MM schemes and different sets of MM point charges.(*82*) In principle, CM*x* charges(*142-145*) are even more accurate than ESP charges, but in any given case, the results do depend on the CM*x* parameterization, which may or may not lead to an improvement as compared to ESP charges.

In conclusion, we remind users to be careful in choosing MM point charges for gas-phase studies, where validations of the selected MM point charge values are highly recommended.

For a very large system, the polarization effects on the CPS by the background point charges very far away from the CPS are expected to be small due to the screening. Exclusion of those charges is unlikely to have significant effects on the electronic-structure of the CPS and will probably cause only small changes in the relative energies such as the energy of reaction. Thus, in order to reduce the computational cost, one could include in the embedded-QM calculations for the CPS only a subset of the MM background point charges that are within a preset distance called QM/MM cutoff (see the QMMMCUTOFF keyword). When using the QM/MM cutoff, to be on the safe ground, it is strongly recommended that users check the convergence of their calculations with respect to the QM/MM cutoff before drawing conclusions.

4.Q.3. Location of the QM/MM Boundary

The guideline for locating the QM/MM boundary is to make the CPS as small as possible and as large as necessary. From the point of view of computational costs, it is always good to use a smaller CPS. The bottom line is that the electronic structure of the CPS must provide a good

model for the quantum process in the ES. How good the model is depends on the problem under investigation and on the computational method being used. Here we give some general guidelines.

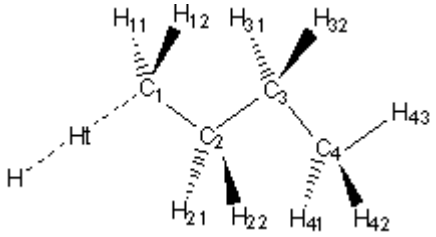
In general, with the extension of the size of the CPS, the QM/MM calculations should show some kind of convergence, e.g., for reaction barrier heights.

In order to minimize the boundary effect on the electronic structure of the CPS, the QM/MM boundary should not go through a polar bond or a multiple bond, and cutting a conjugated ring is not recommended. The ideal choice of the QM/MM boundary is that the boundary goes through a non-polar C–C bond where the C atoms have sp^3 hybridization.

Moreover, the QM/MM boundary should not be too close to the atoms that undergoing bond breaking and forming. A boundary close to a reaction center is not recommended, not only because the electronic structure of the CPS might not be a good model for the quantum process in the ES, but also because it introduces the ambiguity in selecting MM parameters for the PS atoms, as discussed in section 4.Q.1. MM Parameters for the PS. As a rule of thumb, reaction should occur no closer to the boundary than Q4, if possible and affordable.

As an example, we study the H atom transfer reaction

$$H + CH_3CH_2CH_2CH_3 \rightarrow H_2 + CH_2CH_2CH_2CH_3$$



The QM level is set to MPWB1K/6-31+G(d,p), the MM force field is MM3,[94-96] and the mechanical embedding scheme is chosen. We test three locations for the QM/MM boundary, i.e., the boundary going through the C1–C2 bond, the C2–C3 bond, or the C3–C4 bond, giving rise to the CPS as H + CH_4, H + CH_3CH_3, and H + $CH_3CH_2CH_3$, respectively. We also test two kinds of choice for the atom type for the C1 atom: the "sp^3 C in alkanes" and the "C radical", as defined in the MM3[94-96] force field. We compare in Table 4.Q.3.1 the QM/MM optimized saddle-point-geometries with the full QM calculations.

As can be seen from Table 4.Q.3.1, the extension of PS size leads to better agreement with full QM calculations for the critical bond distances involving bond breaking and forming. The QM/MM boundary going through the C2–C3 bond provides a quite reasonable compromise

between computational accuracy and cost. In addition, the different atom-types show largest variations for the smallest CPS, and the C radical (C*) atom-type does a better job, as expected. However, when the QM/MM boundary moves one bond further away and the CPS extends to H + C_2H_6, the difference due to atom types becomes negligibly small.

Test runs 2033 to 2035 give the calculations for entries 2, 4, and 5 in Table 4.Q.3.1.

Table 4.Q.3.1. QM/MM optimized saddle-point-geometries compared with full QM calculations for reaction H + $CH_3CH_2CH_2CH_3$ → H_2 + $CH_2CH_2CH_2CH_3$.[a]

	Full QM	QM/MM								
Entry	1	2	3		4	5		6	7	
		CPS = H + CH_4			CPS = H + C_2H_6			CPS = H + C_3H_8		
R(A–B)		C	C*	Δ_{CC*}	C	C*	Δ_{CC*}	C	C*	Δ_{CC*}
H–Ht	0.905	0.872	0.871	0.001	0.909	0.909	0.000	0.905	0.905	0.000
Ht–C1	1.373	1.420	1.375	0.045	1.366	1.367	-0.001	1.374	1.374	0.000
C1–C2	1.496	**1.503**	**1.491**	**0.012**	1.499	1.498	0.001	1.497	1.497	0.000
C2–C3	1.523	1.536	1.536	0.000	**1.516**	**1.517**	**-0.001**	1.526	1.526	0.000
C3–C4	1.516	1.534	1.534	0.000	1.534	1.535	-0.001	**1.512**	**1.512**	**0.000**

[a] Distance is Å, ΔR_{CC*}(A–B) = R_C(A–B) – R_{C*}(A–B), C_{HL} = 0.734 in calculation for entry 3 and C_{HL} = 0.729 for entries 2, 4, 5, 6, and 7. The C–C bond where QM/MM boundary passes is indicated in bold. See also test runs 2033 to 2035.

4.Q.4. Saddle-Point Optimizations

Optimization is often much more difficult for saddle points than for minima. (See Section 5 for discussion of optimization algorithms.) Here we make some comments and suggestions that might be helpful for users when searching for saddle points employing QMMM.

First, we note that it is often not necessary to optimize saddle points for a very large system, e.g. a protein containing hundreds of residues solvated in a water box. There are at least two reasons for this. (1) The geometry for the whole system can be very floppy, and, for example, a H-bond that breaks or forms 10 Å far away from the active center can cause a big change in the total energy, but such a jump is usually irrelevant to active-site dynamics. (2) We need a new

theoretical description, e.g., the ensemble-averaged variational transition state theory,[43] for dynamical processes in a big liquid-phase or enzymatic system. Therefore, a practical case will be to optimize the saddle point for a subset of atoms while fixing the other (usually distant) atoms. In enzyme dynamics, the number of atoms in the subset might be 25 – 100.

Second, a good guess for the saddle point geometry is often necessary for most optimization algorithms, especially when the potential surface is rather flat. It is often helpful to perform a constrained optimization or a relaxed surface scan by fixing distinguished reaction coordinates, e.g., the distances for the bonds that are forming and/or breaking, so as to have such a good guess. In QMMM, this can be done invoking the *Gaussian* optimizer through the *Gaussian* external option (page 95) by specifying the appropriate keywords in the *multiopt section (page 107). For example, a constrained optimization by fixing the distance between atoms 2 and 3 can be carried out with this choice:

```
GAUEXTOPTIONS
    OPT=(modredund)
    ! 2 3 F
END
```

Another example is a relaxed potential energy scan by scanning the distance (two steps with step size of 0.005 Å) between atoms 2 and 3, which can be accomplished by:

```
GAUEXTOPTIONS
    OPT=(modredund)
    ! 2 3 S   2 0.005
END
```

Next, after having a good guess for the geometry, one can start the optimization for the saddle point. Here, we emphasize the importance of the initial Hessian for the eigenvector following (page 92) algorithm; an initial Hessian of good quality is a great help, and we recommend that this Hessian be calculated at the same QM/MM level. This can be done by specifying the keyword HESSIAN in the *multiopt section (page 107) to be *qmmm* if the QMMM internal optimizer is invoked, and by specifying the *Gaussian* keyword *CalcFC* if the *Gaussian*

optimizer through the *Gaussian* external option is invoked. An example for the latter is as follows:

```
GAUEXTOPTIONS
    OPT=(ts, CalcFC, noeigentest)
END
```

However, as discussed in Section 5.D (page 95), although the *Gaussian* manual says *Gau_External* can pass the energy, gradient, and Hessian, our tests show that in *Gaussian03* Revision B.05, *Gau_External* can only pass energy and gradient, and *Gaussian* will use numerical differentiation to calculate the Hessian needed for optimization. Thus, the use of the *CalcFC* keyword will make the optimization very expensive because the initial Hessian is going to be computed numerically, even if the analytic Hessian at the corresponding QM level is available! A solution to this is to calculate the analytic Hessian beforehand and then input the Hessian through the *Gaussian* keyword *FCCards* (see the QMMM keyword gauextoptions on page 109 for the details).

Finally, after the stationary point is reached, we suggest that users perform a vibrational analysis at the resulting geometry to check if the correct saddle point is obtained. This is recommended, especially when one uses the *Gaussian* Berny optimizer and turns off the *eigentest* option. Sometimes, one can have imaginary-frequency torsional modes; they might or might not be the desired modes. To get rid of undesired imaginary-frequency torsional modes is, however, not easy in general. One might need to tighten the convergence threshold or to start from another geometry guess, or even ignore those modes of very low (e.g., less than 30 cm^{-1}) imaginary frequencies if they are irrelevant to the reaction and are very far away from the active center.

The last point that we wish to point out is that special attention is needed if density functional theory (DFT) is chosen for the QM level. In such cases, it is often necessary to tighten the convergence threshold for SCF (*scf=tight*) and for geometry optimizations (*opt=tight*) and to use the ultrafine grid (*int=grid=ultrafine*) for integration, the *Gaussian* keywords being given in parentheses.

4.Q.5. QM/MM Cutoff

The use of the QM/MM cutoff (via the QMMMCUTOFF keyword in the QM/MM section) allows the embedded-CPS calculation includes only a subset of the MM background point charges that are within a distance from a user-defined center. The center is not necessarily an atom, although one can specify an atom to be the center. If one specifies an atom as the center, the center may change coordinates, e.g., in a geometry optimization. Alternatively, one can specify the center by providing its Cartesian coordinates, and those coordinates will be fixed in all calculations.

The use of the QM/MM cutoff reduces computational effort, and it is justified by that the background point charges far away from the CPS have insignificant effects on the CPS electronic structures due to screening. Excluding those "far-away" background point charges in the embedded-QM calculations for the CPS will probably change negligibly the relative energies (e.g., the energy difference between the reactant and product), although the absolute energies will change. The use of the QM/MM may cause sudden change in energy in a geometry optimization if SS atoms enter or leave the cutoff sphere. It is necessarily to check the convergence of the interested molecular properties such as reaction barrier heights and spin densities with respect to the QM/MM cutoff before making any definite conclusions.

The QM/MM cutoff must be used with caution in molecular dynamics, especially in adaptive-partitioning schemes. There is a possibility that the cutoff leads to numerical instability in conservation of energy and momentum.

4.Q.6. Using Previous *Gaussian* Checkpoint File

The use of previous *Gaussian* checkpoint files (machine-dependent!) is convenient in many situations. This is easy to do, as exemplified here by testrun2050:

> Copy the checkpoint file to the directory where the QMMM input files (`test2050.dat`, `test2050.ml`, `test2050.prm`, and `test2050.crd`) locate, and rename the checkpoint file to `guess.chk`.

4.Q.7. Limitations of the Program

A. Maximum Number of Atoms in Energy, Gradient, Hessian, and Optimization Calculations

Currently, the maximum number of atoms than can be handled is 9900 for energy and gradient calculations and 800 for Hessian calculations. For geometry optimization using internal optimizers, the limit is 500, the same as in Hessian calculations. For geometry optimization using *Gaussian* optimizer via the *Gaussian* external option, the limit is 9900 for full optimization (the same as in energy and gradient calculations) and 1000 for partial optimization. Be aware that *Gaussian* may request huge amount of memory (e.g., 10 GW) for geometry optimization for large-size systems.

The current setting should be enough for many applications. However, one can modify the parameters in the `module.F` file such that even larger system can be treated. Be aware that this may not work in 32-bit machines, for which large-memory request is not possible.

To modify the maximum number of atoms for energy and gradient calculations as well as for geometry full-optimization employing the *Gaussian* optimizer, one must update the following lines (make sure that the new values you put in are the same in all lines):

In `module input`,

```
integer, parameter :: maxatm = 9900
```

In `module param`,

```
integer, parameter :: maxatm = 9900
```

In `module gradient`,

```
integer, parameter, private :: maxatm  = 9900
```

In `module ehard`,

```
integer, parameter, private :: maxeeatm  = 9900
```

To modify the maximum number of atoms for geometry partial-optimization employing *Gaussian* optimizer, one must update the following lines:

In `module input`,

```
integer, parameter :: maxpartatm = 1000
integer, parameter :: maxpart1stlay = 2000
integer, parameter :: maxpart2ndlay = 3000
```

Here, `maxpartatm` specifies the maximum number of moving atoms to be included in the partial optimization, whose Cartesian coordinates will be passed to *Gaussian*. The maximum numbers of 1st-layer frozen atoms, i.e., the fixed-coordinate atoms that directly bond to the moving atoms, is specified by `maxpart1stlay`. The maximum number of 2nd-layer frozen atoms, i.e., the fixed-coordinate atoms that directly bond to the 1st-layer frozen atoms, is specified by `maxpart2ndlay`. The program will automatically determine the 1st- and 2nd-layer frozen atoms for users after the users specify the moving atoms; if the numbers of those atoms exceed the values specified here, users should specify new values.

To modify the maximum number of atoms for Hessian calculations as well as for geometry optimization employing internal optimizer, one must update the following lines (please make sure that the new values you put in are the same in both lines):

In `module optimize`,

```
integer, parameter, private :: maxatm = 500
```

In `module hessian`,

```
integer, parameter :: hessmaxatm = 500
```

B. Maximum Number of Covalent Bonds that can be Cut

The maximum number of covalent bonds that can be cut is set to 30. This should be enough for most applications. To modify this number, one should change the value in the following lines manually in the `module.F` file (please make sure that the new values you put in are consistent in all lines):

In `module input`,

```
integer, parameter :: maxcapatm = 30
integer, parameter :: max1stlay =  30
integer, parameter :: max2ndlay =  90
integer, parameter :: maxredist =  90
integer, parameter :: max3rdlay = 270
```

In `module gradient`,

```
integer, parameter, private :: maxgrad  = 3*maxatm+540
```

Here, the relation between the maximum numbers is as following:

number of capping atoms = number of bonds being cut

number of M1 atoms = number of capping atoms

number of M2 atoms = number of M1 atoms × 3

number of redistributed charges = number of M2 atoms

number of M3 atoms = number of M2 atoms × 3

number of gradient components = number of real atoms × 3 + number of M1 atoms × 18

C. Maximum Number of Atoms in Group-Based Treatments

In the polarizable-boundary and flexible-boundary treatments, users divide SS atoms that are to be treated by polarization and/or charge transfer into groups; within each of those groups, charge transfer occurs between the atoms based on the principle of electronic chemical potential equalization. The maximum number of atoms in any one of those polarizable/charge-transferable groups is set to 9999. This should be enough for most applications. To modify this number, one should change the value in the line below manually in the `module.F` file.

In `module input`,

```
integer, parameter :: maxidnum    = 9999
```

D. Maximum Number of Groups in Polarizable-Boundary Treatments

The maximum number of polarizable groups (within each of which charge transfer is allowed) is set to 100. This should be enough for most applications. To modify this number, one should change the value in the line below manually in the `module.F` file.

In `module input`,

```
integer, parameter :: maxpolgnum   = 100
```

E. Maximum Number of Groups in Flexible-Boundary Treatments

The maximum number of polarizable groups (within each of which charge transfer is allowed) is set to 100. The SS atoms that exchange charges with the PS belong to the first group by design. This should be enough for most applications. To modify this number, one should manually change the value in the line below in the `module.F` file.

In `module input`,

```
integer, parameter :: maxgnum  = 100
```

F. Maximum Numbers of Lines for Optional Keyword Input for QM and MM Packages

The maximum numbers of lines for optional keyword input for QM and MM packages are set to 999. This should be enough for most applications. To modify those numbers, one should manually change the value in the lines below in the `module.F` file.

In `module input`,

```
integer, parameter :: maxoptl = 999
```

G. Maximum Numbers in Adaptive-Partitioning Simulations

The maximum number of groups in the buffer zone is 64 in permuted-AP simulations and is 100 in the sorted-AP, hot-spot, and ONIOM-XS simulations. The highest order of energy expansion number in the permuted-AP is 13, and the maximum number of permutations is 10,000.

H. Other Limitations

The width of the `.crd` and `.dat` input files must be 80 columns or less.

The title in the `.dat` file should be 5 lines or less.

The names of atom types should not be longer than 4 characters.

The length of a keyword must be shorter than 20 characters, preferably shorter than 10 characters. There are other limitations, but they are in general not important to users.

Chapter Five

5

5. Optimization Procedures

This chapter describes the optimization procedures available in this version of the QMMM code.

5.A. Optimization Algorithms

Currently there are two kinds of geometry optimization algorithms available in QMMM: (1) four Newton-Raphson-type algorithms and (2) the eigenvector following (EF) algorithm.

The differences among the Newton-Raphson-type algorithms are in the treatment of the Hessian in those steps where it is not recalculated with an HHOOK call. Each of the three algorithms takes a Newton-Raphson (NR) step with Brent line minimization(*146*) at every iteration of the optimization. The NR step is calculated by solving the following linear equation for $\boldsymbol{x}$:

$$\boldsymbol{H}\,\boldsymbol{x} = -\boldsymbol{g} \qquad (5.A.1)$$

where $\boldsymbol{H}$ is the Hessian matrix, $\boldsymbol{x}$ is a vector consisting of the Cartesian components of the step, and $\boldsymbol{g}$ is the gradient vector. The Hessian and gradient are both in unscaled Cartesian coordinates. Brent line minimization(*146*) then scales the geometry step $\boldsymbol{x}$ to a magnitude that minimizes the energy by the greatest amount. This line minimization attempts to prevent taking steps that are too large or too small.

The first Newton-Raphson algorithm (to be called just plain NR from now on) does not alter the Hessian in steps where the Hessian is not recalculated with an HHOOK. Equation (5.A.1) is simply solved using the most recently calculated Hessian. If a Hessian is not recalculated at every step, this is a quasi-Newton algorithm. The remaining two algorithms are always quasi-Newton methods; in the steps between Hessian recalculations, eq. (5.A.1) is not solved using a true Hessian from a previous iteration, but rather using an approximate inverse Hessian created from a variable metric update. The second algorithm uses Broyden-Fletcher-Goldfarb-Shanno (BFGS) Hessian updates, and the third uses Davidon-Fletcher-Powell (DFP) Hessian

updates.(*147*) The forth algorithm is the limited memory BFGS (LBFGS). Both of these updating algorithms take advantage of information contained in the change in the gradient from one iteration to the next in order to build up corrections to the inverse Hessian matrix, which should in principal tend to converge to the true inverse Hessian, at least for the important components. In fact the two types of inverse Hessian updates differ from one another by only one term.

One might notice that both updating schemes operate on the inverse Hessian. Due to this, in the implementation of the BFGS and DFP algorithms the step size is obtained with another equation instead of eq. (5.A.1):

$$\boldsymbol{x} = -\boldsymbol{H}^{-1}\boldsymbol{g} \qquad (5.A.2)$$

This manner of solving for the Newton-Raphson step is somewhat more memory intensive than just solving eq. (5.A.1) in that it involves explicitly finding the inverse of the Hessian; the NR algorithm does not actually compute $\boldsymbol{H}^{-1}$ but rather solves eq. (5.A.1). Since the BFGS and DFP algorithms require the inverse Hessian, utilization of eq. (5.A.2) is most reasonable for them.

EF is an optimization routine based on Simons' P-RFO algorithm as implemented by Baker.(*148*) Step scaling to keep the step size within the trust radius is taken from Culot et al. (*149*) The trust radius is automatically updated dynamically by the method of Fletcher.[127] The EF step is calculated by the following linear equation for $\boldsymbol{x}$:

$$\boldsymbol{x} = (s\boldsymbol{I} - \boldsymbol{H})^{-1}\boldsymbol{g} \qquad (5.A.3)$$

where s is a shift factor which ensures that the step length is within or on a hypersphere, and $\boldsymbol{I}$ is the unit vector. If the Hessian has one and only one negative eigenvalue, the shift factor is set equal to zero. If this step is longer than the trust radius, a P-RFO step is attempted. If this is also too long, then the best step on the hypersphere is made via the QA formula. Both P-RFO and QA steps are obtained with eq. (5.A.3), but these methods use different formulas for s**.**

Using the step calculated, a new geometry is obtained, at which a new energy and gradient are evaluated. If it is a TS search, two criteria are used in determining whether the step is accepted. The ratio between the actual and predicted energy change should ideally be 1. If it deviates substantially from this value, the second order Taylor expansion is no longer accurate. If the ratio is outside the interval defined by the **RMIN** and **RMAX** limits, the step is rejected, the trust radius reduced by a factor of two, and a new step is determined. The second criterion is

that the eigenvector along which the energy is being maximized should not change substantially between iterations. The minimum overlap of the TS eigenvector with that of the previous iteration should be larger than **OMIN**; otherwise the step is rejected.

In the EF routine, there are three Hessian update options which are specified by the keyword **IUPD**. The BFGS and DFP updating schemes are included in the EF routine. The third option is to reuse the Hessian without updating, i.e., to freeze the Hessian. The BFGS update is generally regarded as the best update to use for optimizing to a minimum energy structure, but it tends to preserve positive definiteness, i.e., if the Hessian before the update is positive definite (all the eigenvalues are positive), then the updated Hessian will also have this property. For this reason, BFGS is not recommended for a transition state search. The DFP update has no particular bias towards positive definiteness. Thus the DFP updating scheme is recommended for transition state optimization.

Historical note: As discussed elsewhere,(*150, 151*) a more appropriate name for the Newton-Raphson method would be "Newton-Raphson-Simpson method."

5.B. Hessians Obtained with OPTHHK

As the above equations make clear, each optimization algorithm calculates the step size based upon the Hessian or some update of the Hessian inverse. For smaller molecules, using some of the less expensive methods within QMMM, the use of the true Hessian is reasonable. Yet many of the methods contained in this program are expensive, and, due to this, calculating a Hessian every few steps in an optimization may not be feasible. Therefore, OPTHHK may employ 4 different types of Hessian strategies depending on the user's specifications (see the **HESSIAN** keyword in the multiopt section). Option 1 is to use the Hessian calculated at the QM/MM level. Option 2 is that one uses a Hessian calculated at a lower level of electronic-structure theory, e.g., semi-empirical level of theory. Option 3 is that one uses a Hessian calculated at an MM level. The last option is to use a unit matrix scaled to the approximate magnitude of the components of the true Hessian. While the last three types of Hessians do not contain information on the exact second derivatives of the potential energy surface, it has been shown that any type of Hessian results in faster geometry convergence than simply following the gradient.

In particular, we have found that optimizations of minima using lower-level Hessians recalculated every several steps converge in as few if not fewer iterations than optimizations using a "true" Hessian. And one must remember that not only does the optimization with the lower level Hessian converge in a comparable number of iterations, but it is also costs considerably less than an optimization with the higher level Hessian. Optimizations for saddle-points are more complicated, and the use of a lower-level Hessian might not work (see also Section 4.J.4. Saddle-Point Optimizations for more information).

5.C. Comments on Optimizing in Cartesian Coordinates

Cartesian coordinates are currently the only way to specify geometry in QMMM, and the geometry optimizations are performed in Cartesian coordinates.

One issue that must be addressed is that Cartesian coordinates have η more degrees of freedom than are needed to fully describe the system ($\eta = 6$ for non-linear systems and $\eta = 5$ for linear systems). If all Cartesian coordinates are allowed to vary, the optimization becomes unstable because the changes in geometry correspond not only to movement of the atoms relative to one another but also to translations of the entire molecule across space and rotations of the whole molecule. Thus either 6 or 5 Cartesian coordinates are fixed during the optimization; the default constant coordinates are *x*, *y*, and *z* for the first atom, *y* and *z* for the second atom, and if the molecule is non-linear *z* for the third atom. These may be changed (see the **CONSTANT** keyword in the **MULTIGEN** section), but the remainder of this discussion will be carried out using the default. If different coordinates are held constant, the treatment described below should change only superficially in terms of axes and coordinates.

It is not sufficient though to hold only the designated six coordinates constant. The molecule must be oriented in a certain way in order not to lose any generality in the optimization. The *y* and *z* coordinates of the second atom must be the same as the *y* and *z* coordinates of the first atom. And for non-linear molecules, the *z* coordinate of the third atom must be the same the *z* coordinate for the first and second atoms. However, the user is not required to enter a geometry that adheres to these requirements. The optimization routine automatically reorients the molecule to adhere to these requirements (or requirements applicable to the constant coordinates specified by the user). This is achieved by first translating the molecule so that the first atom is at the origin. Then the molecule is rotated about the *z*-axis so that the *y*-coordinate

of the second atom is 0. Subsequently a similar rotation is made about the *y*-axis in order to set the *z*-coordinate of the second atom to 0. Finally, (for non-linear molecules) the molecule is rotated about the *x*-axis in order to force the *z*-coordinate of the third atom to 0.

Since this reoriented geometry may be undesirable to the user, at the end of the optimization these rotations and translations are reversed to place the first three atoms back in their original plane. (See noreorient in the **MULTIGEN** section to switch this off.) Note that this reorientation requires only that the first atom's coordinates all remain the same. The coordinates of the other two atoms most likely will have changed during the optimization. Yet these three atoms should still define the same plane as before the optimization. In our (limited) experience, this reorientation at the end usually results in a negligible energy change of 10^{-12} hartrees. In certain cases, however a change as large as 10^{-9} hartrees has been observed, so the user may want to pay attention to the energy before and after the reorientation.

The freezing of the coordinates has very little effect on the algorithms actually employed during the optimization as described in Section 4. The portions of the Hessian and the gradient that are specific to these frozen coordinates are ignored during the calculation of the geometry steps, thus allowing the step for each of the frozen coordinates to be zero.

5.D. Optimization with *Gaussian*'s Optimizer

If one uses QMMM in conjunction with *Gaussian*, one can use the algorithm keyword **GAUEXT** to specify that geometry optimization is to be carried out by using the optimizers in *Gaussian*. This option is available for both QM/MM and pure MM calculations.

The overall control for this procedure is:

QMMM ↔ *Gaussian*↔ *Gau_External* ↔ QMMM

If one uses **GAUEXT** as the optimization algorithm keyword, the primary QMMM calculation will call a *Gaussian* optimization with the *external* keyword. This *Gaussian* calculation calls an external PERL scripts *Gau_External*, which will provide the QMMM energy, gradient and Hessian needed for optimization. *Gau_External* will call a secondary QMMM calculation and pass the secondary QMMM results to *Gaussian*. When *Gaussian* finishes the optimization, it will return the optimized geometry to the primary QMMM calculation. Note that in the *Gaussian* manual, it says *Gau_External* can pass the energy, gradient, and Hessian, but our tests show that in *Gaussian*

Revision G03/B.05, *Gau_External* can only pass the energy and gradient, and *Gaussian* will use numerical differentiation to calculate the Hessian needed for optimization.

When the keyword **GAUEXT** is presented, the reorientation of the molecule is suppressed, even if the keyword reorient is specified.).

A note of caution: We found that different versions of *Gaussian* (*g03.b01*, *g03.c01*, *g03.d01*, *g03.e01*, *g09.a02*, *g09.e01*, *g16.a03*, *g16.b01*, and *g16.c01*) require slightly different procedures to invoke the *external* option. More specifically, the input and output files required for *g03.d01*, *g03.e01*, *g09.a02*, *g09.e01*, *g16.a03*, *g16.b01*, and *g16.c01* versions are in the scratch directory, while those of *g03.c01* and *g03.b01* are in the directory where the input files locate. Thus, we prepare different shuttle scripts for calling different versions of *Gaussian*. Depending on which *Gaussian* version the user is using, user can select corresponding scripts for his/her calculations. See also Section 8.A. Installation Instructions for more details.

5.E. Partial Optimization

Full optimizations for very large-size systems such as protein are challenging. Only first-order algorithms, which do not require Hessians, are usually practical. This makes tight convergence difficult. Moreover, the first-order algorithms are not suitable for transition-state searches. In practice, one often carries out a partial optimization where a subset of the atoms such as the active site of an enzyme is optimized while keeping the surroundings fixed. The selected atoms that are allowed to move are called the active atoms, while the other atoms fixed to their present coordinates are called the frozen atoms. Of particular interest are the frozen atoms directly bonded to the active atoms, which are called the 1st layer frozen atoms, and the frozen atoms directly bonded to the 1st layer frozen atoms, which are called the 2nd layer frozen atoms. The active atoms are not necessarily to be the QM atoms; they can consist of both the QM and MM atoms.

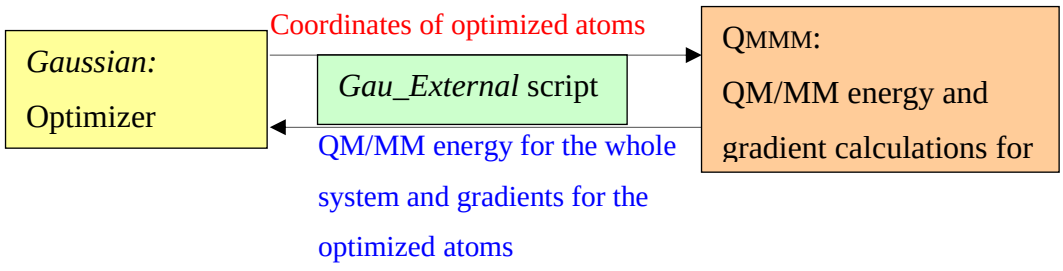
There are two ways of doing partial optimization in QMMM. The first is to use the *Gaussian* optimizer via the external option. The other is to use the internal optimizer with the limited-memory BFGS (LBFGS) algorithm. Please note that those two options employed different keywords to specify atoms: In the *Gaussian* optimizer option, one specifies which atoms to optimize explicitly via the PARTATM keyword, or implicitly by setting the center and radius of a sphere via the PARTRAD and PARTCENTXYZ (or PARTCENTID) keywords. In

the LBFGS option, one specifies the atoms whose positions to be fixed during the optimization via the FROZEATM keyword.

In the partial optimizations using the *Gaussian* optimizer via the *Gaussian* external option, the optimized geometry differs between if only the active atoms and if both the active and frozen atoms are passed to the *Gaussian* optimizer. The reason seems to have something to do with the use of internal coordinates in *Gaussian* for geometry optimization. To be on the safe ground, one would like to pass all frozen atoms passes to *Gaussian*, but that is not always convenient. Fortunately, we found that, when the Newton-Raphson algorithm (`opt=newton`) is invoked, it is often sufficient to pass to *Gaussian* the 1st and 2nd layer of frozen atoms togther with the active atoms. The optimized geometry obtained by such a treatment differs often negligibly from the optimized geometry obtained by passing all frozen atoms to *Gaussian*. However, using the Rational Function Optimization algorithm (`opt=RFO`), which is default in *Gaussian*, does not show the same advantage; the optimized geometry differs noticeably between only the 1st and 2nd layer frozen atoms and all the frozen atoms are passed to *Gaussian*. For this reason, users should select the Newton-Raphson algorithm for partial optimization when the *Gaussian* optimizer is invoked for partial optimizations. See test runs 2037, 2038 and 2039 for examples.

In a partial optimization using the *Gaussian* optimizer through the external option, QMMM only passes the coordinates and gradient of those optimized atoms, which include both the active atoms and the 1st layer (or the 1st and 2nd layers) atoms, to the *Gaussian* optimizer. The *Gaussian* optimizer does not "see" the other atoms, since they are not passed to *Gaussian* optimizer. However, the QM/MM energies and gradients that are required by the *Gaussian* optimizer to determine the coordinate displacements for the optimized atoms are still calculated by QMMM in the presence of the other atoms. That is, although the *Gaussian* optimizer does not see the atoms, it can still "feel" the existence of the atoms.

The procedure for partial optimization is described by the following diagram:



First, QMMM writes a *Gaussian* input file for geometry optimization using the *Gaussian* external option; the *Gaussian* input file lists only the moving atoms specified by the user. Next, QMMM calls *Gaussian* to do the optimization.

When the *Gaussian* optimizer is in work, it writes the coordinates for the moving atoms in a file, and then executes the *Gau_External* script which in turn invokes QMMM to calculate energy and gradients.

QMMM reads in the coordinates of the moving atoms from the file prepared by *Gaussian*, and updates the coordinates of the whole system. With the updated coordinates, QMMM calculates the QM/MM energy and gradients for the whole system. After the energy and gradient calculations are done, QMMM writes the energy (for the whole system) and the gradients for the moving atoms to a new file, and returns the control to the *Gaussian*.

The *Gaussian* program then reads in the energy and gradients given by QMMM. Based on the newly obtained information, the *Gaussian* optimizer is able to determine the coordinate displacements for the moving atoms. Gaussian will write the new set of coordinates to a file, and executes the *Gau_External* script again which will in turn calls QMMM for energy and gradient calculations. The loop is terminated when the geometry converges or the maximum number of steps is exceeded.

Finally, after the *Gaussian* optimization is done, QMMM reads the *Gaussian* output file of optimization, produces the final geometry for the whole system, and ends the partial optimization procedure.

Users have two ways to specify the active atoms. The first way is to list the atomic index for the active atoms. The second way is to specify a sphere via its center and radius, and the QMMM program would make the atoms within the sphere active atoms. The center can be either an atomic index or a point specified by its Cartesian coordinate.

The partial optimization is also possible with the internal first-order limited-memory BFGS algorithm (LBFGS). This option is, however, not yet extensively tested. This will be improved in the future.

Chapter Six

6

6. Input Description

The input file `ml.inp` is divided into six sections namely, the ***MULTIGEN** section, the ***EXTOPT** section, the ***MULTIOPT** section, the ***QMMM** section, the ***TEST** section, and the ***TESTMM** section. The ***MULTIGEN** section must be first in the input file, and the ***MULTIOPT** section must be the second in the input file. Each section *must* be preceded by the asterisk as shown above. The description for each of these sections is given below. There are three types of keywords: switches, variables, and lists. The syntax for each type of keyword is as follows:

Switch

……

Variable *Value*

……

List

.

.

.

End

List keywords must be terminated by **END**, and they may contain other keywords within their bodies (when this is the case, it shall be indicated in the description of the keyword). All keywords are case insensitive. The value for a variable should be on the same line as a variable keyword, though, and the contents of a list keyword should be on the lines between the list keyword and its terminating **END** but not on those two lines. Also, all list keywords specifically designated for a title are constrained to a maximum content of *5* lines, while all list keywords specifically designated for molecular mechanics programs or for electronic structure program options are constrained to a maximum content of *999* lines.

In the sections below, each keyword is in bold, and directly following the keyword is both its type and its default value.

Note that keywords are not case sensitive.

6.A. *MULTIGEN Section

The ***MULTIGEN** section contains keywords that are needed for any calculation. The keywords are:

CALMODEL VARIABLE *qm*

The CALMODEL select the QM, MM, or QM/MM model.

mm: MM

qm: QM

qm/mm: QM/MM

CHARGE VARIABLE *0*

The CHARGE keyword is used to specify the charge of the entire system.

DEBUG/NODEBUG SWITCH *NODEBUG*

The DEBUG keyword is used to specify additional information to be printed for debugging.

ENERGY/NOENERGY SWITCH *ENERGY*

The ENERGY keyword is used to specify a single-point energy calculation of the system defined by the GEOM keyword.

GEOM LIST *no default*

The GEOM keyword specifies the geometry of the entire system. It is required if **MMVALEN** is not set, i.e., geometry is not given in a separate file. The following is an example of the input format, although there are no strict requirements on spacing or number formats so long as an atom and its *3* coordinates are on the same line.

```
GEOM
      O      0.4515E+00   -0.3543E+00   0.0000E+00
      H      0.4853E+00    0.6115E+00   0.0000E+00
      H     -0.4788E+00   -0.6161E+00   0.0000E+00
END
```

GEOMTYPE VARIABLE *cartesian*

The GEOMTYPE keyword is used to specify the format of the geometry. Currently Cartesian coordinates are the only valid type of geometry specification.

GEOMUNIT VARIABLE *ang*

The GEOMUNIT keyword is used to specify the units of the geometry.

ang: Angstroms

au: atomic units (bohrs)

GRADIENT/NOGRADIENT SWITCH *NOGRADIENT*

The GRADIENT keyword is used to specify a single-point gradient calculation of the system defined by the GEOM keyword. The energy is also calculated.

HESSIAN/NOHESSIAN SWITCH *NOHESSIAN*

The HESSIAN keyword is used to specify a single-point Hessian calculation of the system defined by the GEOM keyword. The energy and gradient are also calculated, and the program will calculate the harmonic vibrational frequencies and the normal mode eigenvectors in the mass-scaled coordinates. The eigenvalues are printed (including the six zero eigenvalues corresponding to translations and rotations), and the eigenvectors are printed both in mass-scaled Cartesians and in unscaled Cartesians.

MMVALEN/NOMMVALEN SWITCH *NOMMVALEN*

The MMVALEN keyword is used to specify whether the geometric data (including MM connectivities) should be read from the coordinate file (`.crd`). This is required for MM and QM/MM calculations; in such a case, the GEOM section is not needed.

MULTIPLICITY VARIABLE *1*

The MULTIPLICITY keyword is used to specify the multiplicity of the entire system, i.e., *1* for singlet, *2* for doublet, etc.

NATOMS VARIABLE *no default*

The NATOMS keyword is used to specify the total number of atoms in the system. When requesting an IMO calculation, NATOMS should include the cap atom (number of atoms in the entire system plus the cap atom). When requesting an QM/MM calculation, NATOMS does *not* include the cap atom. This keyword is required for all calculations. The maximum allowed value is 9900.

PRNATMPRM/NOPRNATMPRM

NOPRNATMPRM

The PRNATMPRM keyword is used to specify whether the MM parameters will be printed; it is valid for pure MM and QM/MM calculations.

PRSUM/NOPRSUM SWITCH *PRSUM*

The PRSUM keyword is used to specify whether a summary file is printed.

TITLE LIST *no default*

The TITLE keyword allows the user to give up to a five-line description of the calculation.

6.B. *EXTOPT Section

The *EXTOPT section contains keywords that are needed for a pre-optimization with an external electronic structure or molecular mechanics program. This section is useful for creating good starting point geometries. The initial guess geometry for the external optimization should be supplied by the GEOM keyword in the MULTIGEN section in the case of a pure QM calculation, or be supplied in the coordinate file (`t41.crd`) in the cases of pure MM or QM/MM calculations. The keywords in the EXTOPT section are:

BASIS VARIABLE *3-21g*

Keyword that indicates the basis set to be used for the pre-optimization at the QM level. Note: When GAMESS is selected for QM calculations, this keyword is ignored, and the basis set is actually specified in the OPTION list. (See also Section 4.P.1. Calling gamess).

FORCEFIELD LIST *no default*

This keyword is used to give the name of the molecular mechanics force field used for the external pre-optimization. This is required when the MM model is selected. The name of the molecular mechanics force field should be consistent with the parameter file (see Section 7.B.3. Molecular mechanics force field Parameter File).

METHOD VARIABLE *hf*

This keyword specifies the electronic structure theory level at which to carry out the pre-optimization. Note: When DFT methods are desired and ORCA is selected for QM calculations, this keyword should be set to DFT, and the functional, e.g., B3LYP, should be specified in the OPTION list. (See also Section 4.P.3. Calling orca). If GAMESS is selected for QM calculations, please see Section 4.P.1. Calling gamess for special instructions.

MMCONVERG VARIABLE *0.01*

This keyword passes a number as the convergence threshold to the external MM program for geometry pre-optimization.

OPTIONS LIST *all options off*

This keyword is used to give the options the user desires for the external pre-optimization. There may be a maximum of *999* lines of options.

If electronic structure program *Gaussian* is called, there must at the very least be one line specifying the optimization algorithm. For example to request a transition-state optimization in *Gaussian* one must give the following OPTIONS:

Options
  Opt=TS
End

To specify the memory requirements and the number of processors to be used when calling *Gaussian*, simply add Link1 commands to this list. Link1 commands (% commands) must come first in the list of options as shown below:

Options

%nproc=2

%mem=800mb

scf=(tight,maxcycle=1000) Opt=(ts,noeigentest)

End

If electronic-structure program ORCA is called, the lines containing keywords should begin with ">", which is an indicator of ORCA keywords. The reason of using this indicator is that ORCA also uses END as keyword as QMMM does. The indicator ">" in the above example makes these two END keywords distinguishable. For example, for the DFT calculations with ORCA, the functional should be written as follow:

Options

>%method functional b3lyp

>end

End

If electronic-structure program GAMESS is called, the lines containing keywords should begin with "!", which is an indicator of GAMESS keywords. In particular, the basis set must be specified through the OPTION list instead of through the BASIS keyword. For example, for calculations using the 6-31G basis set, the basis set could be input as follows:

Options

! $basis gbasis=n31 ngauss=6 $end

End

If molecular mechanics program TINKER is called, we recommend users to specify keyword *digits 8* in order to improve precision for the output.

PROGRAM VARIABLE *g03*

The PROGRAM keyword specifies the electronic structure or molecular mechanics package to be used for the geometry pre-optimization. Currently only GAMESS (gamess), *Gaussian* (*g03*), ORCA (*orca*), and *TINKER* (*tinker*) are valid input values.

VERSION VARIABLE *4.2*

This keyword is used to give the version of the molecular mechanics package used for the external pre-optimization. This is required when the MM model is selected. So far, five versions of TINKER, i.e., version 3.5, version 4.1, version 4.2, version 5.1, and version 6.3 have been tested with the current version of QMMM. The name of the molecular mechanics force field should be consistent with the parameter file (see also Section 4.P.4. Calling TINKER).

6.C. *MULTIOPT Section

The *MULTIOPT section contains keywords to specify a geometry optimization of the system via QMMM multi-level algorithms or via the algorithms in *Gaussian* with the *external* keyword. If an external pre-optimization has been carried out, the pre-optimized geometry will be used as a starting point for this optimization. Otherwise, the initial geometry is obtained from the MULTIGEN section in the cases of QM calculations or from the coordinate file (`t41.crd`) in the cases of MM or QM/MM calculations. If MULTIOPT is used, the geometry is subsequently redefined for all remaining calculations as the resulting optimized geometry. Six of the keywords in this section, in particular DDMAX, DDMAXTS, IUPD, OMIN, RMAX and RMIN are used only with the EF algorithm. Eleven keywords CAPPA, FROZEATM, INTPOLAT, MAXANG, MAXDX, MAXSLOPE, MAXSTEP, MINENR, MINSTEP, RMSDX, and RMSGRAD are used only with the LBFGS algorithm, with three of them, MAXDX, RMSDX, and RMSGRAD, set the convergence criteria. The valid options are:

ALGORITHM VARIABLE *nr*

This keyword specifies the optimization algorithm to be used. The algorithms currently available are Newton-Raphson with Brent line minimization (*nr*), NR with Broyden-Fletcher-Goldfarb-Shanno updates on the Hessian (*bfgs*), NR with Davidon-Fletcher-Powell updates to the Hessian (*dfp*), EF with three Hessian update scheme, algorithms in *Gaussian* (*gauext*), and the limited-memory BFGS quasi-newton nonlinear optimization (*lbfgs*).

CAPPA VARIABLE *0.9*

The CAPPA variable is used to set the maximum of the normal. If the ratio of the current gradient projection on the line to the projection at the start of the line search falls below this value, the line-search exits successfully. It is valid in the limited-memory BFGS algorithm.

CONSTANT LIST *see below*

The CONSTANT keyword indicates which coordinates will be frozen during the optimization. The default is that after reorientation the first atom will be frozen at the origin (i.e. its *x*, *y*, and *z*

coordinates remain fixed). In addition, the second atom will be fixed on the *x*-axis and the *y* and *z* coordinates of the second atom will be held constant. If the molecule is non-linear, the third atom will be in the *xz*-plane and its *z*-coordinate will remain fixed. The user may change which coordinates will be fixed, but it is strongly recommended that these coordinates be chosen in such a way that one atom has 3 coordinates fixed, a second atom has 2 coordinates fixed, and (if the system is non-linear) a third atom has 1 coordinate fixed. Additionally the coordinate held constant for this third atom should be one of the two coordinates held constant for the second atom. As long as these guidelines are adhered to, the system will be neither over- nor under-constrained during an optimization. An example of the use of the CONSTANT keyword is given below. The first integer on each line is the number of an atom, and the letters following specify which coordinates will be frozen for that atom.

```
CONSTANT
   3          x
   7          x    y    z
   2          x    z
END
```

DDMAX VARIABLE *0.5*

DDMAXTS VARIABLE *0.3*

The DDMAX and DDMAXTS variables are used to set the maximum of the trust radius (in angstroms); the DDMAX variable is used for minima (equilibrium structures), and the DDMAXTS variable is used for saddle points (transition states).

DEBUG VARIABLE *0*

This keyword allows user to control how much debugging information will be provided in the output file. (Note: the size of the output file can be very large if debug information is requested for dynamics simulations!) Possible values are 0 and positive integers, as described below:

0: Standard output without debugging information

1: All debug information including those in Debug options 3 to 5

2: All debug information without those in Debug options 3 to 5

3: Writes velocities and gradients in files *fort.61* and *fort.62* when the trajectory is recorded.

4: Writes the analysis of velocity distribution in dynamics simulations in the main output file

5: Writes information for adaptive-partitioning QM/MM dynamics in the main output file.

FROZEATM LIST *no default*

This keyword list the atomic ID of the frozen atoms in the optimization. It is valid in the internal limited-memory BFGS algorithm. (Please do not confuse it with the PARTATM keyword, which is exclusively for when invoking the *Gaussian* external optimizer. See Section 5.E. Partial Optimization for more discussion.)

```
FROZEATM
  1   2   3   4   5
  6   7
END
```

GAUEXTOPTIONS LIST *all options off*

This keyword is used to give the options the user desires for the *Gaussian* optimizer. There may be a maximum of *999* lines of options, and there must be at least one line specifying the optimization algorithm. Lines containing keywords that should be input after molecule specification should begin with "!". For example to request a transition state optimization in *Gaussian* with the distance between atoms 2 and 3 fixed, one can give the following OPTIONS:

```
GAUEXTOPTIONS
  OPT=(modredund)
  ! 2 3 F
END
```

Special attention should be given to saddle point optimizations. We found that when one uses the *Gaussian external* option, the *Gaussian* optimizer only asks for a gradient and then evaluates the Hessian numerically, even if the analytic Hessian is available in direct *Gaussian* calculations (i.e., not through the *external* option). This makes the saddle point optimization expensive if one wishes to make use of the Hessian for the initial geometry. In such a case, we suggest users obtain the gradient and Hessian for the initial geometry from a single point calculation, where the Hessian is calculated analytically. Then the user can supply this energetic information to the

saddle point optimization process (through *Gaussian external* option) by specifying the *Gaussian* keyword *FCCards*, e.g.,

GAUEXTOPTIONS

opt=(ts, FCCards, noeigentest, maxcycle=100)

! -1.18017367278035E+02

! 0.01115327 -0.00385643 -0.00492079 0.00000021 -0.00000014 -0.00000020

! -0.00000018 -0.00000032 0.00000014 0.00000034 -0.00000013 -0.00000010

! -0.01115479 0.00385488 0.00492290 -0.00000330 0.00000809 -0.00000760

…

END

The first line that begins with "!" gives the energy in the (D24.16) format, the next lines give the gradient, and final lines give Hessian, where gradient and Hessian in the (6F12.8) format. A PERL script called `qmmmhess2g03` is provided in the `script` directory, which can be used to convert the Hessian calculated by QMMM (as given in the `ml.sum` file) into the *Gaussian* format that can be cut and pasted into the input file with the *FCCards* keyword as shown above. This is particularly useful for saddle-point optimization. See also Section 4.J.4. Saddle-Point Optimizations for more information.

One can require *Gaussian* to do more than just optimization by taking advantage of this option. For example, one can perform a vibrational normal mode analysis by specifying the keyword *freq*.

GCOMP VARIABLE *1.0e-3*

This keyword gives the convergence criterion for the optimization. In particular, once the component of the gradient with the maximum magnitude falls below this value in atomic units with the BFGS, NR, DFP, and EF algorithms, the structure is considered optimized. For the LBFGS algorithm, there are three additional convergence criteria: RMSGRAD, MAXDX, and RMSDX.

HBAS VARIABLE *3-21G*

The HBAS keyword indicates the basis set to be used for the lower level Hessian calculations. Note: this keyword is only valid when the HESSIAN keyword value is *lowqm*.

HESSIAN VARIABLE *mm*

The HESSIAN keyword specifies the type of Hessian to be used in the optimization algorithm. In this version of QMMM, four options are available: a scaled unit matrix (*unitmat*), a Hessian at a low-level electronic-structure theory (*lowqm*), a Hessian at the MM level that is specified in the QM/MM section (*mm*), and a Hessian at the QM/MM level as the optimization method chosen (*qmmm*). See also Section 5.B. Hessians Obtained with opthhk for discussions.

HMETH VARIABLE *hf*

This keyword gives the method for the lower-level QM Hessian to be calculated. Note: this keyword is only valid when the HESSIAN keyword value is *lowqm*.

HPROG VARIABLE *g03*

This keyword indicates the program to be used to gather the low-level QM Hessian. Note: this keyword is only valid when the HESSIAN keyword value is *lowqm*.

HREC VARIABLE *10*

This keyword indicates the number of iterations between recalculation of the Hessian. If the HESSIAN keyword is set to *unitmat*, the HREC keyword is ignored.

HSCALE VARIABLE *1.0e-5*

The HSCALE keyword gives the value by which the unit matrix used for a Hessian is scaled. Note: this keyword is only significant when the unit matrix is chosen for the HESSIAN keyword or INITHESS is set to *off*.

INTPOLAT VARIABLE

5

The INTPOLAT variable is used to set the maximum number of interpolation cycles during the line search phase of an optimization. If the value is exceeded, the status is set to error and the search is repeated with a much smaller initial step size. It is valid in the limited-memory BFGS algorithm.

INITHESS VARIABLE *on*

This keyword tells whether a Hessian should be calculated before the first step of the optimization or a scaled unit matrix should be used initially as the Hessian. The choice is either *on* or *off*.

IUPD VARIABLE

0

IUPD = *n* selects the Hessian updating scheme in Eigenvector Following optimizations.

IUPD = 0 No updating

IUPD = 1 Powell updating scheme

IUPD = 2 BFGS updating scheme

LINMN VARIABLE *on*

This keyword tells whether a Brent line search should be performed during geometry optimization. The choice is either *on* or *off*.

MAXANG VARIABLE

180

The MAXANG variable is used to set the maximum of the permissible angle between the current optimization search direction and the negative of the gradient direction. If this value is exceeded, the optimization is failure. It is valid in the limited-memory BFGS algorithm.

MAXDX VARIABLE *0.4*

This keyword sets one of the convergence criteria: the maximum of the permissible displacement (in atomic units) in the Cartesian coordinates for the optimizations. It is valid in the limited-memory BFGS algorithm.

MAXSLOPE VARIABLE *1.0e4*

The MAXSLOPE variable is used to set the maximum of the ratio between the current and initial projected gradients. If this value is exceeded, the initial step size is reduced by a factor of 10. It is valid in the limited-memory BFGS algorithm.

MAXSTEP VARIABLE *5.0*

The MAXSTEP variable is used to set the maximum of the step size computed as the norm of the vector of changes in the optimized parameters. If this value is exceed, the step size is set to the value. It is valid in the limited-memory BFGS algorithm.

METHOD VARIABLE *qmmm*

This keyword specifies the theory level at which to carry out the optimization. The valid value for the variable is *qmmm*.

MINENR VARIABLE *-1.0e-6*

This keyword gives the enforced convergence criteria for the optimization. If the absolute energy in the atomic unit falls below the value, the optimization is enforced convergence. This is valid in the limited-memory BFGS algorithm, and it is most useful in code debugging.

MINSTEP VARIABLE *1.0e-9*

The MINSTEP variable is used to set the minimum of the step size computed as the norm of the vector of changes in the optimized parameters. If the step size falls below this value, the step size is set to the value. It is valid in the limited-memory BFGS algorithm.

MOLTYPE VARIABLE *nonlin*

This keyword indicates the type of molecule to be optimized. Currently the four valid values for this variable are *lin*, *nonlin*, *lints*, or *nonlints*, for a linear reactant/product, a non-linear reactant/product, a linear saddle point, or a non-linear saddle point, respectively. Currently this keyword has two effects. The first is that if the molecule is a saddle point, Brent line minimization is turned off. The second is that the program will freeze 5 coordinates during the optimization for a linear species, as opposed to 6 for a non-linear species.

NITER VARIABLE *50*

This keyword gives the maximum number of iterations in the optimization.

OMIN VARIABLE

0.8

During transition state optimizations, the EF algorithm calculates the dot product between the previously followed direction and the eigenvector of the Hessian corresponding to the imaginary-frequency mode. The new step will be along the direction defined by the eigenvector for which this dot product is maximum, if this value is greater than OMIN.

PARTIAL LIST

This keyword specifies the options for the partial optimization to be done using the *Gaussian* optimizer via the external option. See Section 5.E. Partial Optimization. In the partial optimization, only the Newton-Raphson algorithm is permitted, which is to be specified by users in the input. The following are its valid options:

EXTLAYERNUM VARIABLE *1*

This keyword is used to specify how many layers of frozen atoms that are surrounding the active atoms are passed to the *Gaussian* optimizer. The 1st layer frozen atoms are the frozen atoms that are directly bonded to the active atoms, and the 2nd layers of frozen atoms are those frozen atoms that are directly bonded to the 1st layer frozen atoms. This keyword has three options: 0, 1, and 2. Option 0 indicates no frozen atoms are passed to the *Gaussian* optimizer, i.e., only the active atoms are passed to the *Gaussian* optimizer.

Option 1 denotes both the active atoms and the 1st layer frozen atoms are passed to the *Gaussian* optimizer, and option 2 indicates the active atoms, the 1st layer frozen atoms, and the 2nd layer frozen atoms are passed to the *Gaussian* optimizer. If option 1 or 2 is used, one needs to add in GAUEXTOPTIONS the following *Gaussian* keyword:

opt=(modredund, newton)

PARTATM LIST *no default*

This keyword is used to specify the atoms to be optimized in the partial optimization. The list of the optimized atoms must be input by the user. Each line can list up to 5 atoms.

```
PARTATM
 1   2   3  4    5
 6   7  8  9  10
11
END
```

PARTCENTID VARIABLE *no default*

This keyword indicates the atomic index of the central atom of a sphere that defines active atoms. The atoms within the sphere, i.e., within a distance from the central atom, are active atoms in the partial optimization. See also the keywords PARTCENTXYZ and PARTRAD.

PARTCENTXYZ VARIABLE *no default*

This keyword indicates the Cartesian coordinates (in Å) for the center of a sphere that defines active atoms. The center is not necessarily the coordinates of an atom. The atoms within the sphere, i.e., within a distance from the center, are active atoms in the partial optimization. For example, one specifies the center to be at (X, Y, Z) = (3.00, 4.00, 5.00):

```
PARTCENTXYZ    3.0    4.0    5.0
```

See also the keywords PARTCENTID and PARTRAD.

PARTCHARGE VARIABLE *0*

The keyword is used to specify the total charge of the subset of the atoms to be optimized in the partial optimization. This charge is formally required by the *Gaussian* optimizer, and it should be consistent with the multiplicity of the subset of the atoms to be optimized (see the PARTMULT keyword). However, the charge and the multiplicity for the subset of optimized atoms do not affect the optimized geometry, because they do not affect the actual calculations of energy and gradients.

PARTINIT

If the keyword PARTINIT is specified, the QMMM program will print out the list of active atoms in partial optimization and stop. This is useful for users to check and decide which atoms are included in the partial optimization.

PARTMULT VARIABLE *1*

The keyword is used to specify the multiplicity of the subset of the atoms to be optimized in the partial optimization. This multiplicity is formally required by the *Gaussian* optimizer, and it should be consistent with the charge of the subset of the atoms to be optimized (see the PARTCHARGE keyword). However, the charge and the multiplicity for the subset of optimized atoms do not affect the optimized geometry, because they do not affect the actual calculations of energy and gradients.

PARTRAD VARIABLE *999.d0*

The PARTRAD keyword is used to specify the radius (in Å) for the sphere that defines active atoms. The atoms within the sphere, i.e., within a distance from the center, are active atoms in the partial optimization. See also the keywords PARTCENTID and PARTCENTXYZ.

REORIENT/NOREORIENT SWITCH *REORIENT*

As stated above, for an optimization, the molecule's orientation is changed in such a way that one atom will remain at the origin, another will remain on an axis, and a third will remain in a

plane. REORIENT returns these three atoms to their original plane once the optimization is done. If the user prefers to leave the molecule in the orientation it had during the optimization for any further energy, gradient, and Hessian calculations, this may be achieved by specifying NOREORIENT. When the *Gauextoptions* keywords are present, the reorientation is suppressed, even if the *reorient* switch is turn on.

RETRY VARIABLE *on*

This keyword tells whether the optimization routine should switch to HREC=1 if the optimization fails with regards to STPTOL.

RMIN VARIABLE

0.0

RMAX VARIABLE

4.0

For an Eigenvector Following step to be accepted, the value of the ratio of the calculated energy change to the predicted energy change must be bracketed by the values of RMIN and RMAX. Default values are RMIN = 0 and RMAX = 4.

RMSDX VARIABLE *0.01*

This keyword specifies the convergence criterion of the component change RMS of the coordinate for the optimization (in the atomic unit). It is valid in the limited-memory BFGS algorithm.

RMSGRAD VARIABLE *0.01*

This keyword specifies the convergence criterion of the gradient RMS for the optimization (in the atomic unit). It is valid in the limited-memory BFGS algorithm.

SCALE VARIABLE *1.0*

This keyword gives the maximum value (in bohrs) of the square root of the sum of the squares of the components of the calculated step in the geometry. Should the step exceed this value, every component of the step is scaled smaller to yield a sum of this size.

STPTOL VARIABLE *1.0e-5*

The STPTOL keyword specifies the failure criteria for an optimization. If the maximum component of the calculated step is smaller than this value (in bohr), the optimization will fail unless RETRY is on. The reason for this is that when a geometry step is very small, there is very little change in either the energy or the gradient. The overall effect of this situation is a useless geometry step. Since the gradient has not changed at all, the next geometry step will take one to geometry with the exact same result. Therefore, the net effect of such a small geometry step is a stalled optimization.

6.D. *QM/MM Section

This *QMMM section contains keywords that are specific to the QM/MM methods.

BORDERCHARGE VARIABLE *rcd*

This keyword specifies the way to treat the MM point charges close to the QM/MM border. Currently there are five ways implemented:

- Scale the charges on M1, M2, and M3 (*scale*); the SEE, Z1, Z2, and Z3 schemes require this option.
- Shift the M1 charges on to M2 and adding a pair of point charges in vicinity of M2 to preserve the M1−M2 bond dipole (*shift*).
- Redistribute the M1 charge onto the M1−M2 bond (*redist1* or *rc*); this is the RC scheme.
- Redistribute the M1 charge onto M1−M2 bond and also add a pair of point charges in vicinity of M2 to preserve the M1−M2 bond dipole (*redist2* or *rcd2*); this is the RCD2 scheme.
- Redistribute the M1 charge onto M1−M2 bond and preserve the M1−M2 bond dipole by modifying the M2 and redistributed charges (*redist3* or *rcd*); this is the RCD scheme.

The following schemes are balanced schemes.

- Redistribute the adjusted M1 charge onto the M1−M2 bond (*rcbal*); this is the balanced RC scheme.
- Keep the adjusted M1 charge on M1 atom (*seebal*); this is the balanced SEE scheme.
- Redistribute the adjusted M1 charge on the nearest M2 atom (*amber1*); this is the Amber-1 scheme.
- Redistribute the adjusted M1 charge evenly to all M2 atoms (*rcbal2*); this is the balanced RC2 scheme.
- Redistribute the adjusted M1 charge evenly to all M2 and M3 atoms (*rcbal3*); this is the balanced RC3 scheme.
- Redistribute the adjusted M1 charge evenly to all MM atoms, except M1 atoms (*amber2*); this is the Amber-2 scheme.

- Redistribute the adjusted M1 charge onto the M1–M2 bond and compensate the movement of the charges by modifying the M2 and redistributed charges (*rcdbal*); this is the balanced RCD scheme.
- Redistribute the adjusted M1 charge evenly to all M2 atoms and add dipoles around M2 atoms to compensate the movement of the charges (*shiftbal2*); this is the balanced shift scheme.

This keyword will be ignored if the ME scheme is selected via the EMBED keyword.

BORDERTYPE VARIABLE *capatm*

This keyword indicates the cap atom to be used for QM/MM border treatment. This is the only option that is implemented in the current version of the QMMM program.

CAPATOM LIST *see below*

This keyword is used to provide information for the cap atoms. Each line contains four parameters for one cap atom, and understading the fourth parameter requires that you have already read the description of the **CAPCONSTRAIN** keyword. The four parameters are given in this order: (1) the atom center for the MM host (M1 atom), (2) the atom type name for the cap atom, (3) the atom type for the cap atom, and (4) the scale factor C_{HL} if scaled-bond-distance scheme is used or the fixed distance between Q1 and the link atom D_{HL} (in equation 4.H.3) if the fixed-bond-distance scheme is used. In the example below using the scaled-bond-distance scheme, one uses HLL for the atom type name for the cap atom that replaces atom 4 with a scale factor of 0.71, and one uses the HA atom type for the cap atom that replaces atom 5 with a scale factor of 0.72. In the example below using the fixed-bond-distance scheme, one uses HLL for the atom type name for the cap atom that replaces atom 4 with a fixed distance of 1.30. Such information must be supplied by users; the atom type name and the atom type depend on the molecular mechanics force field that one uses. C_{HL} is set to 0.713 by default, but the use of this default value without examining its performance is dangerous.

For scaled-bond-distance scheme:

CAPATOM

4 HLL 199 0.71

5 HA 1 0.72

END

For fixed-bond-distance scheme:

CAPATOM

4 HLL 199 1.30

END

The MM charges for the cap atoms are set to zero by the QMMM program, regardless their actual values in the force field. Zeroing out the MM charges on the cap atoms does not affect the charges for the normal MM atoms.

CAPCONSTRAIN VARIABLE *scale*

This keyword indicates the how the location is determined for the cap atom. Currently the allowed values are *scale* and *fix*, which correspond to scaling the Q1–HLL distance with respect to Q1–M1 distance by the factor C_{HL} specified in the CAPATOM section, and to fixing the Q1–HLL distance at D_{HL} specified in the CAPATOM section. If tuned F link atom is used, it is suggested to use *fix* option. [See Section (4.H.2) for definition for C_{HL}.]

BALGROUP LIST *see below*

This keyword is used to provide information for conserving the total charge of QM/MM system in the balanced methods. In each group, the charge on the M1 atom is adjusted to make the total charge of the group to a provided value. Each block contains the total charge of the group, the index of the M1 atom, and indices of other atoms in the group. Notice that only one M1 atom is allowed to appear in a group. If balanced methods (e.g., balanced RC and balanced RCD) are used without specifying BALGROUP, the whole MM region is considered as a group, and the charge on M1 atom will be adjusted to conserve the total charge of the QM/MM system.

BALGROUP 0.0

1

2

3

END

CHARGELAYER VARIABLE *3*

This keyword indicates the how many MM tiers at the QM/MM boundary are involved where MM point charges are scaled. Currently the allowed maximum number of layers is 3. This keyword is needed only when the BORDERCHARGE keyword is set to *scale*.

CHARGEPOSIT VARIABLE *0.5*

This keyword indicates the location of the redistributed M1 point charge along the M1–M2 bond. The value is equal to the ratio of the M1–q_0 distance and the M1–M2 distance, i.e., Cq_0 defined in equation (4.C.1) on page 33, and it must be between 0.1 and 0.9. A value within the range of 0.4 to 0.8 is highly recommended. This keyword is needed only when the BORDERCHARGE keyword is set to *redist1* (or *rc*), *redist2* (or *rcd2*), or *redist3* (or *rcd*), and otherwise it is ignored.

CHARGESCALE LIST *0.0, 0.0, 0.0*

This keyword is used to give the scale factor for each MM layer where the MM point charges are going to be scaled. It is needed only when the BORDERCHARGE keyword is set to *scale*, and otherwise it is ignored. The default setup is to set the values to 0.0 for all the M1, M2, and M3 atoms; doing this corresponds to the Z3 scheme. The following example corresponds to the Z2 scheme.

```
CHARGESCALE
   0.0
                  0.0
   1.0
END
```

CHGCORR/NOCHGCORR SWITCH NOCHGCORR

The CHGCORR/NOCHGCORR keyword is used to specify whether to use the balanced M1 charge in the MM calculations of the SS system. If CHGCORR is turned on, balanced M1 charge is used to calculate E(Coul;SS) in equation 4.I.7.

DAMPCHG VARIABLE *0*

The DAMPCHG keyword is used to specify the screened charge (ODS) model and smeared charge model used for the MM charges and the redistributed charges used in QM-MM interactions. In the current implementation, the screened charges and smeared charges are only used to evaluate the QM-MM interactions, while the MM-MM interactions are always evaluated by point charges. Currently a value of 0, 1, 2 and 3 is allowed.

0: No screened or smeared charge scheme is used.

1: The redistributed charges are smeared, and the MM charges are screened using the outer-density screening (ODS) method. The MM atoms being screened are defined in the DAMPATOM keyword.

2: The redistributed charges are not smeared, but the MM charges are screened using the outer-density screening (ODS) method. The MM atoms being screened are defined in the DAMPATOM keyword.

3: The redistributed charges are smeared, but the MM charges are not screened.

DAMPATOM LIST *see below*

This keyword is used to provide information for the MM charges that are screened. Each line contains parameters for one screened MM charge. (1) If only one number is provided, the program reads the MM center, and assigns the zeta value and number of screened electrons from Table 4.F.1 and eq. 4.F.3. (2) If three numbers are provided, the first number is the atom number of the screened atom in the coordinate file, the second number is the ζ value, the third number is the number of valence electrons; the number of electrons in the screening region n_{screen} will be the third number minus *q*, where *q* is the partial atomic charge on the atom. (3) If four numbers are provided, the first number is the atom number of the screened atom in the coordinate file, the second number is the ζ value, the third number is the number of valence electrons, and the last number is the extra charge included in the screening region. The number of electrons in the screening region n_{screen} will be the third number minus the fourth number. If one wants to turn off the screening of a screened atom, use 0.0 and 0.0 for the third and fourth numbers (no electrons in the screening region).

DAMPATOM

1

3 1.32 1.0

4 1.12 1.0 0.0

END

If a MM charge is not specified in the list, only point charge scheme will be used for this MM charge.

DIPDIST VARIABLE *0.2*

This keyword indicates the distance between the members of the point charge pair (q_- and q_+) added to preserve the M1−M2 bond dipole. The value is equal to the $Cq_\pm$, i.e., the ratio of the distances $R(q_- - q_+)$ and the distance R(M1−M2) [see equation (4.C.9) on page 34 for the definition for $Cq_\pm$], and it should be between 0.1 and 0.9. A value within the range of 0.15 to 0.4 is highly recommended. This keyword is needed only when the BORDERCHARGE keyword is set to *shift* or *redist2* (or *rcd2*).

DIPPOSIT VARIABLE *1.0*

This keyword indicates the location of the center for the two point charges (q_- and q_+) added to preserve the M1−M2 bond dipole. The value is equal to the ratio of the distances between Q1−*q*c (*q*c is the center of position for q_- and q_+) and Q1−M1, and should be between 0.1 and 1.9. A value within the range of 0.5 to 1.2 is highly recommended. This keyword is needed only when the BORDERCHARGE keyword is set to *shift* or *redist2* (or *rcd2*).

EMBED VARIABLE *electric*

This keyword indicates the type of embedding. Currently the allowed values are *mechanical* and *electric* which correspond to mechanical and electronic embedding schemes, respectively.

FLEXBOUND LIST

This keyword specifies the options for the flexible-boundary treatment. See Section 4.E. Flexible-Boundary Treatment. Currently this treatment works only in the situations where the QM/MM boundary does not go through a covalent bond. The electronic-structure packages *Gaussian* and ORCA are currently supported for the polarized-embedding calculations. The following are its valid options:

CHARGE VARIABLE *0*

The CHARGE keyword is used to specify the charge of the second state for the PS, in addition to the first state of the PS, which is already specified by the QMKEY keyword. In the flexible-boundary treatment, one needs to specify the charges and multiplicities for the PS in both the reduced state and the oxidized state. The first state, which can be either the reduced state or the oxidized state, is often set to the state where the PS carries the normal formal charge, e.g., the Na^+ state where the Na center carries a formal charge of +1 e. The charge and multiplicity of the first state are specified by the QMKEY keyword. For the second state, e.g., the Na state where the Na center carries a formal charge of 0, the charge is specified here by the CHARGE keyword.

CALCHARGMETH VARIABLE *qeqrg*

This keyword specifies the classical polarization method for the determination of the partial atomic charges of the SS atoms in the flexible-boundary treatments. There are three literature methods implemented: the charge equalization method employing a shielded coulomb term proposed by Rappé and Goddard (QEQRG), a modified version of the charge equalization method by Bakowies and Thiel (QEQBT), and the electronegativity equalization method of Mortier and coworkers (EEM). By default, the parameters used for QEQRG are those of Rappé and Goddard, the parameters used for QEQBT are those of Bakowise and Thiel, and the parameters used for EEM are those of Mortier and workers, but the users can override the defaults with the PARAMETER keyword.

FBGROUPID VARIABLE *1*

The keyword specifies the flexible-boundary group ID, which charge would flux.

GROUP VARIABLE *no default*

The keyword specifies the partitioning of polarizable SS atoms into groups. The groups are identified by their ID, which are 1, 2, 3 … In each group, the atomic index are listed. Each line can list up to 5 atomic index. The example below shows two polarizable groups

for the SS, and that atoms 1, 2 and 3 are put into group 1, while atoms 4, 5, and 6 are put into group 2.

GROUP 1
1 2 3
END
GROUP 2
4 5 6
END

MAXCHARGTRANS VARIABLE *0.02*

The keyword specifies one of the convergence criteria for the flexible-boundary calculations, namely the allowed maximum amount of charge transferred between the PS and SS. The other keywords for convergence are RMSDQ and MAXDQ.

MAXCYCLE VARIABLE *30*

The keyword specifies the allowed maximum number of iterations for flexible-boundary calculations.

MAXDQ VARIABLE *5.0d-3*

This keyword specifies one of the convergence criteria for the flexible-boundary calculations, namely the allowed maximum change in the partial atomic charges on the SS atoms. The other keywords for convergence are MAXCHARGTRANS and RMSDQ.

MULT VARIABLE *1*

The MULT keyword is used to specify the multiplicity of the second state for the PS, in addition to the first state of the PS, which is already specified by the QMKEY keyword. In the flexible-boundary treatment, one needs to specify the charges and multiplicities for the PS in both the reduced state and the oxidized state. The first state, which can be either the reduced state or the oxidized state, is often set to the state where the PS carries the normal formal charge, e.g., the Na^+ state where the Na center carries a formal charge of

+1 e. The charge and multiplicity of the first state are specified by the QMKEY keyword. For the second state, e.g., the Na state where the Na center carries a formal charge of 0, the multiplicity is specified here by the MULT keyword.

PARAMETER LIST *no default*

The keyword allows users to specifies their own parameters used in the charge equalization procedure for the determination of atomic partial charges. The user-input parameters will override the default parameters implemented in QMMM. The parameters (in eV) are χ^0 and J^0 for the QEq method and χ^* and η^* for the EEM method. See QEq parameter 40 and EEM parameter 41. In the example below, one specifies parameters for the elements H and O.

PARAMETER
H 4.528 13.890
O 8.741 13.364
END

RMSDQ VARIABLE *2.0d-3*

This keyword specifies one of the convergence criteria for the flexible-boundary calculations, namely the allowed room mean square change in the partial atomic charges on the SS atoms. The other keywords for convergence are MAXCHARGTRANS and MAXDQ.

TEMPERATURE VARIABLE *no default*

This keyword specifies the electronic temperature. If not specified, it will be determined by the program automatically. See section 4.E. Flexible-Boundary Treatment

LAMBDA VARIABLE *1.0*

The LAMBDA keyword is used to specify the smearing width of the smeared redistributed charges used in RC, RCD, balanced RC, balanced RCD, and so on.

NGTO1 VARIABLE *3*

The NGTO1 keyword is used to specify the number of Gaussian functions to fit a Slater-type function for the screened charges.

NGTO2 VARIABLE *6*

The NGTO2 keyword is used to specify the number of Gaussian functions to fit a Slater-type function for the smeared redistributed charges.

PSEUDOF/NOPSEUDOF SWITCH NOPSEUDOF

The PSEUDOF keyword is used to specify whether the tuned pseudo F link atom is used in QM/MM calculations.

POLAR LIST

This keyword specifies the options for the polarized-embedding treatment for the RC and RCD schemes, i.e., to do the PRC and PRCD calculations, of which the standard PBRC and PBRCD methods are special cases. See Section 4.D. The PBRC and PBRCD Schemes. The electronic-structure packages *Gaussian* and ORCA are currently supported for the polarized-embedding calculations. The following are its valid options:

CYCLE VARIABLE *30*

The CYCLE keyword specifies the maximum number of the iterations for the charge equalization procedure in the embedded-QM calculations.

GROUP LIST *no default*

This keyword specifies the SS atoms in one group for the charge equalization procedure in the embedded-QM calculations. The list of the atoms must be input by user. Each line lists at most 5 atomic IDs. In the PBRC and PBRCD methods, there is only one group. If there is more than one polarized group, one simply repeats the keyword for each polarizable group.

GROUP
1 2 3 6 7
END

GROUP

4 5 16 17

END

In the above example, the SS atoms 1, 2, 3, 6, and 7 are put into the first polarizable group, and the SS atoms 4, 5, 16, and 17 are put into the second polarizable group.
All the unspecified SS atoms are put into the unpolarized group, whose charges are fixed during the charge equalization procedure.

GROUPNUM VARIABLE *0*

This keyword indicates the total number of the groups, excluding the unpolarized group where the SS atoms of fixing charges are registered.

MAXDQ VARIABLE *5.0d-3*

This keyword gives one of the convergence criteria for the charge equalization procedure, namely the allowed maximum change in the charges on the SS atoms. The other keyword for convergence is RMSDQ.

METHOD VARIABLE *none*

This keyword specifies the method for the determination of the background charges in the SS polarization treatments. There are three literature methods: the charge equalization method proposed by Rappé and Goddard (QEQRG), a modified version of the charge equalization method by Bakowies and Thiel (QEQBT), and the electronegativity equalization method of Mortier and coworkers (EEM) The four valid options are: NONE, QEQRG, QEQBT and EEM. By default, the parameters used for QEQRG are those of Rappé and Goddard, the parameters used for QEQBT are those of Bakowise and Thiel, and the parameters used for EEM are those of Mortier and workers, but the users can override the defaults with the PARAMETER keyword.

MPOT VARIABLE *uq0*

This keyword has two valid options: UM1 and UQ0. In the calculations using the UM1 keyword, the external electric field is calculated before one redistributes the charge on the

M1 atom (q_{M1}), while in the calculations using the UQ0 keyword, the field is calculated by the redistributed charge q_0 instead of q_{M1}). In either way, the redistributed charges q_0 does not change value during the mutual polarization treatment. We recommend the UQ0 option.

PARAMETER LIST *no default*

The keyword allows users to specifies their own parameters used in the charge equalization procedure. The user-input parameters will override the default parameters implemented in QMMM. The parameters (in eV) are χ^0 and J^0 for the QEq method and χ^* and η^* for the EEM method. See QEq parameter 40 and EEM parameter 41. In the example below, one specifies parameters for the elements H and O.

```
PARAMETER
  H      4.528   13.890
  O      8.741   13.364
END
```

RMSDQ VARIABLE *2.0d-3*

This keyword gives one of the convergence criteria for the charge equalization procedure, namely the allowed RMS change in the charges on the SS atoms. The other keyword for convergence is MAXDQ.

QMATOM LIST *no quantum atoms*

This keyword is used to specify the atom centers in the QM subsystem. The list of QM atoms must be input by the user.

```
QMATOM
  1
  2
END
```

QMKEY LIST

This keyword specifies the options for the QM calculations to be done. The following are its valid options:

BASIS VARIABLE *3-21G*

The BASIS keyword indicates the basis set for the QM calculation. Note: When GAMESS is selected for QM calculations, this keyword is ignored, and the basis set is actually specified in the OPTION list. (See also Section 4.P.1. Calling gamess).

CHARGE VARIABLE *no default*

The CHARGE keyword is used to specify the charge of the CPS. By default, the charge of the CPS is set to that of the ES. For flexible-boundary treatments, this specifies the charge of the first oxidation state. See also the keyword flexbound.

METHOD VARIABLE *hf*

The METHOD keyword specifies the method for the QM calculation. Note: When DFT methods are desired and ORCA is selected for QM calculations, this keyword should be set to DFT, and the functional, e.g., B3LYP, should be specified in the OPTION list. See also Section 4.P.3. Calling orca for details. If GAMESS is selected for QM calculations, please see Section 4.P.1. Calling gamess for special instructions.

MULTIPLICITY VARIABLE *no default*

The MULTIPLICITY keyword is used to specify the multiplicity of the CPS. By default, the multiplicity of the CPS is set to that of the ES. For flexible-boundary treatments, this specifies the multiplicity of the first oxidation state. See also the keyword flexbound.

OPTIONS LIST *no default*

This keyword indicates the options for the electronic-structure program call.

If electronic-structure program *Gaussian* is called, lines containing keywords that are going to be input after the molecule specification and before the keyword NONBON should

begin with "!", and lines containing keywords that are going to be input after the keyword NONBON should begin with "!2". Users who want to use previously obtained checkpoint files please see Section 4.L.6. Using Previous Gaussian Checkpoint File.

If the electronic-structure program ORCA is called, the lines containing keywords should begin with ">", which is an indicator of ORCA keywords. The reason of using this indicator is that ORCA also uses END as keyword as QMMM does. The indicator ">" in the above example makes these two END keywords distinguishable. For example, for the DFT calculations with ORCA, the functional should be written as follow:

OPTIONS

>%method functional b3lyp

>end

END

If electronic-structure program GAMESS is called, the lines containing keywords should begin with "!", which is an indicator of GAMESS keywords. In particular, the basis set must be specified through the OPTION list instead of through the BASIS keyword. For example, for calculations using the 6-31G basis set, the basis set could be input as follows:

OPTIONS

! $basis gbasis=n31 ngauss=6 $end

END

PROGRAM VARIABLE *g03*

This keyword indicates the electronic structure package to be used for the QM calculations. The only valid option is *g03*, *g09*, *g16*, *gamess*, and *orca*.

QMMMCUTOFF LIST

This keyword specifies the options for the use of the QM/MM cutoff. The keyword allows the embedded-QM calculations for the CPS include only a subset of the MM

background point charges that are within a distance from a user-defined center. (See Section 4.K.5. QM/MM Cutoff) The valid options are:

CUTOFFCENTID VARIABLE *no default*

This keyword indicates the ID of the central atom. The MM background point charges that are within a cutoff distance from the central atom are included in the embedded-QM calculations.

CUTOFFCENTXYZ VARIABLE *no default*

This keyword indicates the Cartesian coordinates (in Å) for the center, which is not necessarily the coordinates of an atom. The MM background point charges that are within a cutoff distance from the center are included in the embedded-QM calculations. For example, one specifies the center to be at (X, Y, Z) = (3.0, 4.0, 5.0):

CUTOFFCENTXYZ 3.0 4.0 5.0

CUTOFFRAD VARIABLE *999.d0*

The CUTOFFRAD keyword is used to specify the cutoff radius (in Å).

MMKEY LIST

This keyword specifies the options for the MM calculations to be done. The following are its valid options:

FORCEFIELD VARIABLE *no default*

The FORCEFIELD keyword indicates the molecular mechanics force field for the MM calculation. See Section 7.B.3. Molecular Mechanics Force Field Parameter File for details.

OPTIONS LIST *all options off*

This keyword indicates the options for the MM program call. We suggest that users specify the TINKER keyword DIGITS 8 in order to increase the precision for the result outputs.

For the list, lines containing keywords that are going to be input only for the entire system should begin with "!1", while keywords that are going to be input only for the large system (i.e. the capped small system and the background charges) should begin with "!2".

Using keywords begin with "!1" and "!2" are recommended only for debugging or for expert users, since the order of atom centers will change in the large system in comparison with that in the entire system. The order for atom centers in the capped small system is as follows:

i. QM atoms as specified in the QMATOM section,
ii. cap atoms as specified in the CAPATOM section,
iii. MM atoms (excluding M1 atoms) as listed in the coordinate files, and
iv. either M1 atoms in the scaled/eliminated charge schemes or auxiliary point charges in the RC, RCD, Shift, and RCD2 schemes.

See the discussion on the atom index on page 74.

For example, there are ten atoms (atoms 1 to 10), three of which are QM atoms (atoms 2, 3, and 4), and two HL atoms are used, whose MM host atoms are atoms 7 and 8, respectively. The order of atoms and auxiliary point charges for the large system will be:
If EMBED is set to *mechanical*: 2, 3, 4, HL7, HL8
If EMBED is set to *electric* and BORDERCHARGE is set to

scale: 2, 3, 4, HL7, HL8, 1, 5, 6, 9, 10, 7, 8

shift: 2, 3, 4, HL7, HL8, 1, 5, 6, 9, 10, $q_{-}(7)$, $q_{+}(7)$, $q_{-}(8)$, $q_{+}(8)$

redist1: 2, 3, 4, HL7, HL8, 1, 5, 6, 9, 10, $q_0(7)$, $q_0(8)$

redist2: 2, 3, 4, HL7, HL8, 1, 5, 6, 9, 10, $q_0(7)$, $q_{-}(7)$, $q_{+}(7)$, $q_0(8)$, $q_{-}(8)$, $q_{+}(8)$

redist3: 2, 3, 4, HL7, HL8, 1, 5, 6, 9, 10, $q_0(7)$, $q_0(8)$

One needs to check very carefully for those atoms of interest when using this option. In the above example, if one wishes to examine the effects due to the MM point charges on MM atom centers 5 and 6 of the entire system, one would need to do calculations with the corresponding charges being zeroed out:

OPTIONS
!1 charge –5 0.00
!1 charge –6 0.00
!2 charge –7 0.00
!2 charge –8 0.00
END

PROGRAM VARIABLE *tinker*

This keyword indicates the electronic structure package to be used for the MM calculations. The only valid option in the current version of the code is *tinker*, which corresponds to the TINKER program. See Sections 4.P.4. Calling TINKER and 7.C.3. The Molecular Mechanics Program Input and Output Files for calling TINKER.

VERSION VARIABLE *4.2*

This keyword is used to give the version of the molecular mechanics package used for the external pre-optimization. This is required when the MM model is selected. So far, four versions of TINKER, i.e., version 3.5, version 4.1, version 4.2, version 5.1, and version 6.3 have been tested with the current version of QMMM. The name of the molecular mechanics force field should be consistent with the parameter file (see also Section 4.P.4. Calling TINKER).

READMMCHG/NOREADMMCHG SWITCH NOREADMMCHG

The READMMCHG/NOREADMMCHG keyword is used to specify whether to read the MM charges from the .dat file. This keyword is only needed when MMFF94 force field is used.

6.E. *TEST Section

The *TEST section contains keywords that are specific to single-level QM calculations (usually for test). This function allows the user to gather energies, gradients, and Hessians at only one level of electronic structure theory. The user may also pre-optimize at the specified level using the algorithms enabled in QMMM. This may be helpful in testing starting point geometries and Hessian levels to be used in an optimization, before attempting to optimize with the more expensive methods. This section's keywords are as follows:

BASIS VARIABLE *cc-pvdz*

The BASIS keyword indicates the basis set to be used in the calculation. Note: When GAMESS is selected for QM calculations, this keyword is ignored, and the basis set is actually specified in the OPTION list. (See also Section 4.P.1. Calling gamess).

METHOD VARIABLE *mp2*

This keyword specifies the electronic structure method to be used. Note: When DFT methods are desired and ORCA is selected for QM calculations, this keyword should be set to DFT, and the functional, e.g., B3LYP, should be specified in the OPTION list. See also Section 4.P.3. Calling orca for details. If GAMESS is selected for QM calculations, please see Section 4.P.1. Calling gamess for special instructions.

OPTIONS LIST *no default*

The OPTIONS keyword may be used to specify any options necessary for the electronic structure program.

If electronic-structure program *Gaussian* is called, lines containing keywords that should be input after molecule specification should begin with "!".

If electronic-structure program ORCA is called, the lines containing keywords should begin with ">", which is an indicator of ORCA keywords. The reason of using this indicator is that ORCA also uses END as keyword as QMMM does. The indicator ">" in the above example makes these two END keywords distinguishable. For example, for the DFT calculations with ORCA, the functional should be written as follow:

OPTIONS

>%method functional b3lyp

>end

END

If electronic-structure program GAMESS is called, the lines containing keywords should begin with "!", which is an indicator of GAMESS keywords. In particular, the basis set must be specified through the OPTION list instead of through the BASIS keyword. For example, for calculations using the 6-31G basis set, the basis set could be input as follows:

OPTIONS

! $basis gbasis=n31 ngauss=6 $end

END

PROGRAM VARIABLE *g03*

This keyword indicates the electronic structure program to be called for the TEST energies, gradients, and Hessians. Currently *g03, gamess* and *orca* is the supported value, which corresponds to the *Gaussian* and ORCA electronic structure program.

6.F. *TESTMM Section

The *TESTMM section contains keywords that are specific to the single-level MM calculations (usually for tests). This function allows the user to gather energies, gradients, and Hessians at only the MM level. The user may also pre-optimize at the specified level using the algorithms enabled in QMMM. This may be helpful in testing starting point geometries and Hessian levels to be used in an optimization, before attempting to optimize with the more expensive methods. Its keywords are as follows:

FORCEFIELD VARIABLE *no default*

The FORCEFIELD keyword indicates the molecular mechanics force field to be used in the calculation. The allowed values of this keyword in the current version of the program are those force fields implemented in the TINKER 6.3 program.

OPTIONS LIST *no default*

The OPTIONS keyword may be used to specify any options necessary for the electronic structure program.

PROGRAM VARIABLE *tinker*

This keyword indicates the molecular mechanics program to be called for the TESTMM energies, gradients, and Hessians. Currently *tinker* is the supported value, which corresponds to the TINKER program. See Sections 4.P.4. Calling TINKER and 7.C.3. The Molecular Mechanics Program Input and Output Files for calling TINKER.

VERSION VARIABLE *4.2*

This keyword is used to give the version of the molecular mechanics package used for the external pre-optimization. This is required when the MM model is selected. So far, five versions of TINKER, i.e., version 3.5, version 4.1, version 4.2, version 5.1 and version 6.3. have been tested with the current version of QMMM. The name of the molecular mechanics force field should be consistent with the parameter file (see also Section 4.P.4. Calling TINKER).

6.G. *DYNAMICS Section

The *DYNAMICS section contains keywords that are specific for molecular dynamics simulations. The propagation of the trajectory is performed employing the velocity Verlet method. The gradient calculation is performed at the MM, QM, or QM/MM level of theory, for which the variables must be defined in the *TEST, *TESTMM, or *QM/MM section. Note: To use the Ewald summation in the MM calculations, it is necessary to add keyword EWALD into the option list of *TESTMM, or the option list of **MMKEY** in the *QM/MM section. Keywords for the *DYNAMICS section are as follows:

ACTIVECENTER VARIABLE *index of 1st QM atom*

This keyword specifies which atom is the center of the active zone for adaptive-partitioning simulations. If keyword is not used and if **ACTIVECENTXYZ** is not set, the first QM atom specified by keyword **QMATOM** in the *QM/MM section is used as the active-zone center. Possible values are integers greather than 0 and less than or equal to the total number of atoms.

ACTIVECENTXYZ VARIABLE *no default*

This keyword specifies the cartersian coordinates (in Å) that is used as the active-zone center for adaptive-partitioning simulations. Possible values are three real numbers.

ACTIVERADIUS VARIABLE *5.0*

This keyword specifies the radius of the active zone in Å for adaptive-partitioning simulations. Possible values are positive real numbers.

ANNEAL/NOANNEAL SWITCH *noanneal*

This keyword indicates if temperature annealing simulation is to be used. In case of annealing, the temperature is annealed linear between **TEMPERATURE** and **FINALTEMP**. Temperature control is gained with Berendsen method. Only the NVT ensemble is working with annealing. This provides a way to do global geometry optimization.

ARGONTEST VARIABLE *0*

This keyword invokes a test run of the adaptive-partitioning schemes at the dual-MM (MM/MM) level employing the model of 171 Ar atoms in a periodic cubic box.(*85*) The high- and low-levels of MM interactions between the Ar atoms are modeled by a Morse potential and by a Lenard-Jones potential. A cutoff distance is used in the evaluation of the pair-wise interactions, i.e., if the distance between a pair of Ar atoms is larger than the cutoff distance, the interaction will be omitted. There are two ways to handle the discontinuity in energy and force at the cut-off distance: force shift or switching function. In Ref. (*85*), force shift is employed. In the QMMM program, both the force shift and switching function options have been implemented. Possible values of the **ARGONTEST** keyword are integers from 0 to 3, with each value may or may not requiring additional arguments:

0: No test run, and no addition arguments required

1: Test run without force shift at the cutoff, with the cutoff distance as the only argument

2: Test run with a switch function of the force and potential, with the cutoff and switch distances as the two required arguments

3: Test run with force shift, with the cutoff distance as the only argument.

BUFFERSIZE VARIABLE *2.0*

In simulations with periodic boundary condition, the size of the periodic boundary box will be set to the difference between the minimum and maximum atomic coordinates plus **BUFFERSIZE**. In QM/MM simulations the first QM atom is set as center of the periodic box and for simulations with adaptive partitioning the active center is the center of the box. Possible values for **BUFFERSIZE** are real numbers, and the unit is Å. For example, if the minimum and maximum coordinates in x is 0.5 and 10.5, the first QM atom is at x=5.5, and the **BUFFERSIZE** is set to 6, the periodic boundary box will start at –5.5 and end at 16.5 in x. If the first atom moves to x=7 then the box will start at -3 and end at 19. In this way the QM zone is always at the center of the box.

CHGCUTOFF VARIABLE *14.0*

This keyword specifies the cutoff distance for the electrostatic interaction calculation at the MM-level. If the distance of two atoms is larger than the specified value, the electrostatic interaction is not calculated. Possible values are positive real numbers in Angstrom.

CHGTAPER VARIABLE *13.0*

This keyword specifies the switch distance for the electrostatic interactions calculated in TINKER. To have a smooth transition for electrostatic interaction when the distance between atoms reaches **CHGCUTOFF,** interactions between atoms with distances between **CHGTAPER** and **CHGCUTOFF** are scaled by a factor. The factor is 1 if the distance is **CHGTAPER** and 0 if the distance is **CHGCUTOFF**. Possible values are positive real numbers. A value between 0 and 1 is interpreted as a fraction of **CHGCUTOFF**. A number greater than 1 is used as the switching distance in Angstrom.

CONSTRAINTSCAN LIST *NOT USED*

This keyword sets the constraint scan method. In this method the specified atoms and groups are moved in the indicated direction while all other atoms are fixed. The usage can be best described by the following example:

CONSTRAINTSCAN

1 2 *# Number of atoms and groups to scan (1 atom, 2 groups)*

10 0.1 0.0 -0.2 # Atom with index 10 is displaced after each step by (0.1,0.0,-0.2) Angstrom

2 1.0 0.0 0.0 # Group with index 2 is displaced after each step by 1 Angstrom in the x direction

5 -1.0 1.0 1.0 # Group with index 5 is displaced after each step by (-1,1,1) Angstrom

END

COOLINGSTEPS VARIABLE *2000*

This keyword indicates the number of temperature annealing steps between the equilibration phases. Possible values are positive integers.

DYNAMICLINK LIST *see below*

This keyword is used to specify the cap atoms in the adaptive-partitioning QM/MM dynamics simulations. This information is needed when treating groups that are molecular fragments instead of whole molecules. For AP simulations with fragmental groups, cap atoms have to be set on the fly when fragmental groups enter or leave the buffer zone. Each line of the list contains four parameters for one cap atom, and those parameters are given in the following order: the atom center for the MM host (M1) atom, the name of the atom type for the cap atom, the atom type for the cap atom, and the scale factor C_{HL} to determine the location for the cap atom. An example is as follows:

```
DYNAMICLINK
   4   HLL   199   0.713
   10   HA    1    0.722
   16   HA    1    0.722
END
```

In the case where most cap atoms belong to the same atom type with the same scaling factor, one can also use "*default*" keyword:

```
DYNAMICLINK
   4   HLL   199 0.71
   DEFAULT   HA    1   0.72
END
```

In the above example, the firs cap atom is specified in the normal way, while all the other cap atoms are specified by the default option.

The MM charges for the cap atoms are set to zero in our QM/MM algorithm and by the QMMM program, regardless the actual values of the atom types (HLL and HA in the above examples) in the force field. Zeroing out the MM charges on the cap atoms does not affect the charges at the real MM atoms.

ENSEMBLE VARIABLE *NVE*

Indicates the ensemble. At this time just the microcanonical and the canonical ensembles are available. Possible values for the variable are “NVE” and “NVT”.

EPSIOLN VARIABLE 10^{-5}

Cutoff value used in the program values to test numerical instability in MD simulations. When a value is smaller than **EPSIOLN**, it is considered zero. The possible values are positive numbers. This value is used by expert users only, and should not be changed in most cases.

EQUILIBRATION VARIABLE *0*

This keyword indicates the number of equilibration steps between the temperature annealing Possible values are positive integers and 0.

FINALTEMP VARIABLE *1.0*

Sets the final temperature in annealing simulations in Kelvin. Possible values are positive real numbers.

FIRSTTIMESTEP VARIABLE *0*

Offset the index number of the first step to be written into the output. Value has no influence on the simulation itself. If **RESTART** is used, the first time step will be reset to the time step specified in the restart file. Allowed values are zero and positive integers.

FIXCOM/NOFIXCOM SWITCH *nofixcom*

If **FIXCOM** is set, the center of mass will be kept fixed, and the total linear momentum will be zero.

GROUPCOM/GROUPATOM SWITCH *groupcom*

This keyword indicates if a group is labeled by the center of mass of the group or by a delegate atom of the group when computing the distance between the group and the active-zone center

groupcom: by the center of mass

groupatom: by the delegate atom (the first atom of a given group as listed in the group file)

LITEST/NOLITEST **SWITCH***nolitest*

This keyword invokes a test run to reproduce the forces and smoothing functions of the sorted-AP scheme employing the model of one Li^+ ion and 8 water molecules, as described in Ref.(*85*)

OUTPUTDUMPS VARIABLE *1000*

This keyword indicates the number of steps after which the current status of the system is written into the ml.out file, i.e., it controls how often the statistical thermodynamics data is recorded. Possible values are positive integers.

PARTITIONING VARIABLE *off*

This keyword specifies how the atoms are partitioned into QM and MM subsystems during MD simulations. Possible keywords are *nobuffer*, *hotspot*, *oniomxs*, *permutedap*, *sortedap*, and *sortedcorr* as described below:

off: fixed partition for the QM and MM subsystems.

nobuffer: The adaptive-partitioning QM/MM without the buffer zone.

hotspot: The Hot-Spot method.(*34*)

oniomxs: The ONIOM-XS method.(*18*)

permutedap: The permuted-AP scheme.(*85*)

sortedap: The sorted-AP scheme.(*85*)

sortedcorr: The sorted-AP scheme with the bookkeeping term correction.(*135*)

PATH LIST *no default*

This keyword specifies the file name for a “path file” that specifies a predefined trajectory for selected atoms in a simulation. During an MD simulation, after the positions and velocities of all atoms are computed for the given time step, the coordinates of those selected atoms will be replaced by the coordinates read from the path file. Such an option is mainly for debugging use, but it might also be useful in free-energy calculations employing the thermodynamics perturbations theory. The predefined trajectory in the path file is divided into blocks, each block corresponding to one time step in the MD simulations. Each block contains multiply lines

specifying the atom index and the desired Cartesian coordinates in Å and is terminated by one line containing only one keyword **END**. No blank line is allowed. An example is given below (the meaning of each line is given by the comment followed by #):

1 2.0 3.0 4.0 # coordinates of atom 1 in step 1 set to (2.0,3.0,4.0)
3 4.0 5.0 4.0 # coordinates of atom 3 in step 1 set to (4.0,5.0,4.0)
END
1 3.0 3.0 3.0 # coordinates of atom 1 in step 2 set to (3.0,3.0,3.0)
END
2 -1.0 -2.0 -1.0 # coordinates of atom 2 in step 3 set to (-1.0, -2.0,-1.0)
1 4.0 3.0 2.0 # coordinates of atom 1 in step 3 set to (4.0,3.0,2.0)
END

The above example specifies the predefined coordinates of atoms 1 and 3 for time step 1, of atom 1 for time step 2, and of atoms 1 and 2 in time step 3. After the third time step, since no pre-defined coordinates are provided, the MD trajectory will be propagated as usual. (Warning: The QMMM program does not check if predefined coordinates of an atom is reasonable or not; it is the user's responsibility to make sure that atoms do not crash into each other and bonds are not overstretched.)

PBUFFER VARIABLE *0.5*

This keyword specifies the thickness of the buffer-zone in Å for adaptive-partitioning simulations. Possible values are positive real numbers.

PCALCULATION VARIABLE *1000*

This keyword sets the maximum number of permutation in a time step for the permuted AP simulation. Possible values are positive integers less than the program limited value 10,000. See Section 4.P.7. Limitations of the Program.

PERIODIC/NOPERIODIC SWITCH *periodic*

This keyword indicates if periodic boundary condition should be used.

PERMUTEDORDER VARIABLE *4*

This keyword sets the allowed highest order of permutation in the permuted-AP simulation. Possible values for this keyword are positive integers less than the program allowed highest order 13. If the value is larger than $\log_2$(PCALCULATION), the highest order of permutation will be given by $\log_2$(PCALCULATION). See Section 4.P.7. Limitations of the Program.

RATTLEEPSILON VARIABLE *0.00001*

Set the precision for the rattle algorithm. Possible values are positive real numbers.

RATTLEHYDRODGEN VARIABLE *off*

Enable or disable the rattle algorithm to constraint non-water bonds with hydrogen. Possible values are "*mm*" for restraints only on MM hydrogens, "*qmmm*" for all hydrogen atoms, or "*off*". RATTLEHYDRODGEN applies only on non-water hydrogen atoms and is independent from the RATTLEWATER setting.

RATTLEITERATION VARIABLE *10000*

This keyword specifies the maximum number of rattle iterations before the algorithm is aborted. Possible values are positive integers. Too small values may lead to abortion of stable simulations.

RATTLEWATER VARIABLE *off*

Enable or disable the rattle algorithm to constraint water molecules. Possible values are "*TIP*" for TIP3P water, "*SPC*" for SPC and SPC\E water models, and "*off*" for no rattle. Constraint is applied to all bonds with the atom types OW and OT. Therefore only the water oxygen should have these types.

REDUCEORDER/NOREDUCEORDER SWITCH *reduceorder*

If REDUCEORDER is set, the program will temporally reduce the highest order of permutation in a time step for permuted-AP simulations when the number of permutation is larger than that set by the keyword PCALCULATION. In that case the program will reduce the order until the number of permutation is less than PCALCULATION. If NOREDUCEORDER is set the program will not reduce the order and stop the simulation with an error message. The REDUCEORDER keyword is intended for use in debugging only. See Section 4. M. Adaptive Partitioning for more information.

RESTART/NORESTART SWITCH/VARIABLE *norestart*

If RESTART is specified, the initial coordinates and velocities are read from the file given as argument (RESTART file name). See Section 7.B.7. Restart File for the description of the restart file.

SEED VARIABLE *Unix time*

Sets the seed for the random number generator. Can be specified by user to reproduce results. If no value is set the system UNIX time at the beginning of the calculation is set as seed. Possible values are positive integers.

STEPS VARIABLE *100*

Sets the number of dynamic steps to be calculated. Allowed values are positive integers.

TCOUPLING VARIABLE *400.0*

This keyword indicates the Berendsen or Nose-Hoover temperature bath coupling parameter in femtoseconds for simulation in the NVT ensemble. Possible values are positive real numbers. Larger numbers results in weaker coupling.

TEMPERATURE VARIABLE *298.0*

This keyword indicates the initial temperature of the system in Kelvin. The initial velocities are distributed randomly as a Normal distribution with the specified temperature. In addition, this value is used as goal temperature for the Berendsen or Nose-Hoover bath in the NVT ensemble. Possible values are positive real numbers.

THERMOSTAT VARIABLE *BERENDSEN*

This keyword indicates the thermostat method. Possible values are “Berendsen” and “Nose-Hoover”.

TIMESTEP VARIABLE *1.0*

Sets the time step length in femto seconds. Possible values are positive real numbers.

TIMING/NOTIMING SWITCH *notiming*

This keyword indicates if the single steps of each time step should be timed. Timing is useful for benchmarking and don’t need to be set for productive simulations.

TRJDUMPS VARIABLE *1000*

This keyword indicates the number of steps after which the current geometry of the system is written into the trajectory file trajectory.arc, i.e., it controls how often the trajectory is recorded Possible values are positive integers.

VDWCUTOFF VARIABLE *14.0*

This keyword specifies the cutoff distance for the Van-der-Waals interaction calculations at the MM level. If the distance of two atoms is larger than the specified value, the VDW interaction is not calculated. Possible values are positive real numbers in Angstrom.

VDWTAPER VARIABLE *13.0*

This keyword specifies the switch distance for the Van-der-Waals interaction calculation in TINKER. To have a smooth transition for VDW interaction when the distance between atoms reaches **VDWCUTOFF,** interactions between atoms with distances between **VDWTAPER** and **VDWCUTOFF** are scaled by a factor. This factor is 1 if the distance is **VDWTAPER** and 0 if the distance is **VDWCUTOFF**. Possible values are positive real numbers. A value between 0 and 1 is interpreted as a fraction of **VDWCUTOFF**. A number greater than 1 is used as switching distance in Angstrom.

XSIZE, YSIZE, ZSIZE VARIABLE *0.0*

These Keywords set the *x*, *y* and *z* length of the periodic box. It can be used instead of **BUFFERSIZE** to set the length directly. Possible values for x**SIZE, YSIZE,** and **ZSIZE** are real numbers, and the unit is Å.

Chapter Seven

7

7. Description of Files in QMMM

This chapter gives a description of the files involved in compiling and running QMMM, such as: the source code needed to compile the program, files required to run QMMM, files created during a run of QMMM, and a script supplied to simplify the running of QMMM.

7.A. Source Code

The QMMM source code is composed of 23 files written in Fortran 90. Subsection 7.A.1 describes each of the files in the source code, and subsection 7.A.2 gives an alphabetical listing and description of each subprogram in QMMM.

7.A.1. Source Code Files

`ap.F`

This file contains the subprograms for adaptive-partitioning QM/MM schemes.

`cutoff.F`

This file contains the subprograms for handling the QM/MM cut-offs.

`display.F`

This file contains the subprograms for displaying most of the QMMM output.

`dynamics.F`

This file contains the subprograms for dynamics simulations.

`eepolar.F`

This file contains the subprogram for the charge equalization procedure.

`ef.F`

This file contains the driver for the eigenvector following algorithm.

`ehooks.F`

This file contains the subprograms for carrying out energy calculations as well as the formula subroutines.

`flexbound.F`

This file contains the subprograms for the flexible-boundary treatment.

`freq.F`

This file contains the subprograms for calculating the harmonic vibrational frequencies and normal mode coordinates.

`gamess.F`

This file contains the subprograms for carrying out the calculations with GAMESS.

`gau_ext_opt.F`

This file contains the subprograms for calling *Gaussian*'s optimizers.

`gau_part_opt.F`

This file contains the subprograms of doing partial optimization using *Gaussian*'s optimizer through the external option.

`ghooks.F`

This file contains the subprograms for carrying out gradient calculations.

`hhooks.F`

This file contains the subprograms for conducting Hessian calculations.

`lbfgs.F`

This file contains the subprograms for the limited-memory BFGS quasi-newton nonlinear optimization.

`lib.F`

This file contains the subprograms for solving the linear equations which are taken from the LAPACK and BLAS.

`main.F`

This file contains the driver for the QMMM program.

`ml.F`

This file contains the subprograms for parsing the QMMM input file, `ml.inp`.

`module.F`

This file contains all the modules defining parameters for the program, creating structures for the input information, and defining common blocks for the energies, gradients, and Hessians to be calculated.

`numhess.F`

This file contains subprograms for the numerical Hessian calculation.

`ohooks.F`

This file contains the subprograms for carrying out optimizations within QMMM itself.

`orca_ext_opt.F`

This file contains the subprograms for carrying out optimizations with the external optimizer in ORCA.

`orca.F`

This file contains the subprograms for carrying out the calculations with ORCA.

7.A.2. Subprogram List

The listings in this section have the subprogram name in bold. The same line has the type of subprogram and the file that contains the subprogram. The following lines then give a short description of the subprogram.

ADDCAPATOMS subroutine ap.F

Determines all covalent bonds between one QM and one MM atom in adaptive partitioning simulations. Replaces the MM atom with a capatom to for the CPS. If no specific capatom parameters are given default values are used.

ATMINFO module module.F

Contains the information used to confirm that the atomic symbols given in the input geometry are valid and to assign an atomic mass based on this symbol.

ATMPARAM subroutine flexbound.F

Determine the atomic parameters in the two-state flexible-boundary treatment.

BGCHSCALE function ehooks.F

Scales the MM point charges near the QM/MM border.

BGCHSHIFT function ehooks.F

Shifts the MM point charges near the QM/MM border if required.

BOXSIZE subroutine flexbound.F

Determines the minimum and maximum atomic coordinates and their differences and computes the size of the periodic boundary box.

BRENT function ohooks.F

Performs Brent line minimization given three points bracketing a minimum.

CALCOM subroutine dynamics.F

Calculates the center of mass.

CALCGROUPCOM subroutine ap.F

Calculates the center of mass of all groups.

CALCULATEARGONSHIFT subroutine ap.F

Calculates shift in force and potential for adaptive-partitioning simulations of the argon model with the force shift option.

CALPX subroutine flexbound.F

Calculates the reduced (oxidized) state weight in the two-state flexible-boundary treatment.

CALTYP module module.F

Contains parameters for identifying the type of calculation (i.e., optimization, energy, gradient, or Hessian calculation)

CASE function ml.F

Converts input strings to all lower case letters for case consistency.

CFLOAT function ml.F

Converts a string to a double precision number.

CFLTFMT function ehooks.F

Converts a string from a formatted checkpoint file to a double precision number. Takes advantage of the known formats of numbers in the checkpoint files.

CHARGCOOR function eepolar.F

Prepares the geometry in the charge equalization calculations using the QEQRG, QEQBT and EEM method.

CHKINP subroutine flexbound.F

Check the input variables in the two-state flexible-boundary treatment.

CHKLN subroutine ml.F

Checks a line for special characters such as a comment, a section start, or a list keyword end.

CHRESET function eepolar.F

Reset the q_{M2}, q_{M3}, and q_0 after the charge equalization calculations.

CONVG subroutine flexbound.F

Determine the convergence in the two-state flexible-boundary treatment.

COPYQMINP subroutine ap.F

Copy two qmmminf type variables for adaptive partitioning simulations.

CPSCOOR subroutine flexbound.F

Assign the appropriate geometry to the PS atoms in the two-state flexible-boundary treatment.

CPSGASI subroutine flexbound.F

Calculate the ionization potential of the gas-phase PS in the two-state flexible-boundary treatment.

CPSMU1 subroutine flexbound.F

Calculate the chemical potential of the gas-phase PS in the two-state flexible-boundary treatment.

CPSMU subroutine flexbound.F

Calculate the chemical potential of the gas-phase PS in the two-state flexible-boundary treatment.

CUTOFFGRAD subroutine cutoff.F

Assemble the QM/MM gradient when the QM/MM cut-off is used.

DATTIM subroutine display.F

Gets the date and time for the output file.

DAXPY subroutine freq.F

BLAS routine that calculates a vector according to **Cx + y**.

DEFGEN subroutine ml.F

Sets the defaults for the MULTIGEN section input information.

DEFPROG subroutine ml.F

Sets the defaults for the electronic structure programs to be called.

DEFQMMM subroutine ml.F

Sets the defaults for the QMMM section input information.

DEFTEST subroutine ml.F

Sets the defaults for the TEST section input information.

DEFTESTMM subroutine ml.F

Sets the defaults for the TESTMM section input information.

DGEDI subroutine freq.F

Computes the determinant and inverse of a matrix using the factors computed by **DGEFA**.

DGEFA subroutine freq.F

Factorizes a double-precision matrix by Gaussian elimination.

DGEMM subroutine lib.F

Performs one of the matrix-matrix operations.

DGER subroutine lib.F

Performs one of the matrix-matrix operations.

DGESV subroutine lib.F

Computes the solution to a real system of linear equations **AX** = **B**.

DGETRF subroutine lib.F

Computes an LU factorization of a general *m*-by-*n* matrix **A** using partial pivoting with row interchanges.

DGETRF2 subroutine lib.F

Computes an LU factorization of a general *m*-by-*n* matrix **A** using partial pivoting with row interchanges.

DGETRS subroutine lib.F

Solves a system of linear equations **AX** = **B** or **A**'**X** = **B**.

DIFIMU subroutine flexbound.F

Calculate the chemical potential calibration in the two-state flexible-boundary treatment.

DISCONV subroutine lbfgs.F

Display all convergence information for the LBFGS optimization.

DISXYZG subroutine lbfgs.F

Display all information for the current LBFGS optimization step.

DLASWP subroutine lib.F

Performs a series of row interchanges on the matrix **A**.

DQCON subroutine flexbound.F

Calculate MAXDQ and RMSDQ in the two-state flexible-boundary treatment.

DOUT subroutine dynamics.F

Writes the current status (statistical thermodynamics data) in the output file.

DOUT0 subroutine dynamics.F

Writes the current status (statistical thermodynamics data) in the output file and writes the current geometry in the trajectory file.

DOUTT subroutine dynamics.F

Writes the current geometry in the trajectory file.

DPIPHI subroutine ap.F

Calculates the gradient $\frac{d\prod_{j=1}^{N}(1-\phi_j)}{dP_i}$ in sorted-AP.

DSCAL subroutine lib.F

Scales a vector by a constant.

DSWAP subroutine lib.F

BLAS routine interchanges two vectors.

DTRSM subroutine lib.F

Solves the matrix equations.

DVARZERO subroutine flexbound.F

Set given double precision variables to zero.

EEAPC function eepolar.F

Calculate the atomic point charges using the QEQRG, QEQBT, and EEM methods.

EECHARGE subroutine eepolar.F

Calculate the external potential and the atomic charges using the QEQRG, QEQBT and EEM methods.

EEMCH subroutine eepolar.F

Calculate the atomic charges with the external potential using the EEM method.

EF subroutine

ef.F

Eigenvector following driver.

EFOVLP subroutine

ef.F

Determines the overlap of the geometry steps in the eigenvector following algorithm.

EHARD module module.F

Contains the parameter of the polarized charge calculations for the QEQRG, QEQBT and EEM method.

ENERGY module module.F

Defines all the COMMON block energy variables.

F1DIM function ohooks.F

A pseudo-one-dimensional function for the energy of the molecule used to perform Brent line minimization of the step size during optimization.

FBBPC subroutine flexbound.F

Calculations for the two-state flexible-boundary treatment.

FBGACOOR subroutine flexbound.F

Assign appropriate geometry to the polarizable atoms in the two-state flexible-boundary treatment.

FBGPOT subroutine flexbound.F

Assign the electric potential on the polarized atoms in the two-state flexible-boundary treatment.

FBGQTOT subroutine flexbound.F

Calculate the total charge of the polarized atoms in the two-state flexible-boundary treatment.

FBPOT subroutine flexbound.F

Calculate the electric potential produced by the PS on the SS atoms in the two-state flexible-boundary treatment.

FBTEMPCALC subroutine flexbound.F

Calculate the electronic temperature parameter in the two-state flexible-boundary treatment.

FCHAR subroutine ml.F

Finds the next character on a line.

FIFTHORDERSPLINE subroutine ap.F

Calculates the 5-th order spline.

FILES module module.F

Contains the definitions of the file handles, names, and locations used throughout the program. May be modified to change file handle numbers, basis set file locations, or

memory allocations used in *Gaussian* input files. But the filenames themselves should not be changed.

FINDMOLECULE subroutine dynamics.F

Marks separate covalently bonded molecules with unique ids. In simulations with periodic boundary condition all atoms of a molecule are shifted at once when the center of mass is left the periodic box.

FPCCP subroutine flexbound.F

Calculate the chemical potential produced by the fixed background charges at a given atom in the two-state flexible-boundary treatment.

FPCHARG subroutine flexbound.F

Assign the background charge to the SS atoms that are not polarized in the two-state flexible-boundary treatment.

FREQCAL subroutine freq.F

Calculates the harmonic vibrational frequencies and normal mode coordinates.

FSPACE subroutine ml.F

Finds the next blank space on a line.

G03BGPOTINP subroutine flexbound.F

Write a *Gaussian* input file for the calculations of the electric potential in the present of background charge in the two-state flexible-boundary treatment.

G03CHINP subroutine eepolar.F

Writes a *Gaussian* input file for the calculations of the external potential on the SS atoms.

G03STINP subroutine ehooks.F

Writes a *Gaussian* input file for the calculations of the capped small system in presence of background point charges.

G03STINP1 subroutine ehooks.F

Writes a *Gaussian* input file for the calculations of the capped small system in presence of background screened or smeared charges.

G03INP subroutine ehooks.F

Creates input files for *Gaussian*.

G03OUTE subroutine ehooks.F

Reads the energy from a *Gaussian* formatted checkpoint file.

G03OUTEP subroutine flexbound.F

Read the electric potential at the positions of the SS atoms from the G03.OUT file in the two-state flexible-boundary treatment.

G03OUTESP subroutine eepolar.F

Read the ESP charges on the small-system atoms from the G03.OUT file.

G03OUTG subroutine ghooks.F

Reads the gradient from a *Gaussian* formatted checkpoint file.

G03OUTH subroutine hhooks.F

Reads the Hessian from a *Gaussian* formatted checkpoint file.

G03OUTO subroutine ohooks.F

Reads the optimized geometry, energy, and gradient from a *Gaussian* formatted checkpoint file.

G03OUTOP subroutine eepolar.F

Read the potential on the SS atoms from the G03.OUT file.

G03PARTINP subroutine gau_part_opt.F

Writes an input file for *Gaussian* to perform QM/MM partial optimization.

G03PARTOUTO subroutine gau_part_opt.F

Reads the optimized geometry of the moving atoms (in a partial optimization) from a *Gaussian* formatted checkpoint file.

GAU_EXT_OPT subroutine gau_part_opt.F

Creates a *Gaussian* input file to perform (full or partial) QM/MM optimizations using the Gaussian's optimizer.

GAU_EXT_OPT1 subroutine gau_ext_opt.F

Creates a *Gaussian* input file to perform full QM/MM optimizations for the entire system using the Gaussian's optimizer.

GAU_EXT_OPT2 subroutine gau_part_opt.F

Creates a *Gaussian* input file to perform partial QM/MM optimizations using the Gaussian's optimizer.

GAUSSRANDOM function dynamics.F

Generates a random number based on Gaussian distribution.

GMSINP subroutine gamess.F

Creates input files for GAMESS.

GMSOUTE subroutine gamess.F

Reads the energy from a GAMESS output file.

GMSOUTG subroutine gamess.F

Reads the gradient from a GAMESS formatted checkpoint file.

GMSOUTH subroutine gamess.F

Reads the Hessian from a GAMESS formatted checkpoint file.

GMSOUTO subroutine gamess.F

Reads the optimized geometry, energy, and gradient from a GAMESS formatted checkpoint file.

GQTOT subroutine flexbound.F

Calculate the total charge of a group of polarized atoms in the two-state flexible-boundary treatment.

GRADIENT module module.F

Defines all the COMMON block gradient variables.

HESSIAN module module.F

Defines all the COMMON block Hessian variables.

HOTSPOT subroutine ap.F

Calculates the energy and gradient of a QM/MM system and uses the hot-spot method to smooth the forces at the atoms in the buffer zone.

ICINT function ml.F

Converts a string to an integer.

IDAMAX subroutine lib.F

BLAS routine finds the index of element having maximum absolute value.

IEEECK subroutine lib.F

Verify that infinity and possibly NaN arithmetic is safe.

ILAENV subroutine lib.F

Choose problem-dependent parameters for the local environment.

INITMD subroutine dynamics.F

Initialize molecular dynamics simulations.

IINPUT module module.F

Contains the structure types into which all the user input information is placed.

INSTANTMOVE subroutine dynamics.F

Applies the constraint scan method for debugging.

INSUMRY subroutine display.F

Prints out a summary of the user input information.

LBFGSOPTG subroutine lbfgs.F

Computes the energy and gradient for a LBFGS optimization.

LCASEL function gamess.F

Converts a letter to the lower case.

LENWORD subroutine gamess.F

Finds the locations of the first and last letters for the first word in a string.

LINMN subroutine ohooks.F

Optimizes the geometry step during an optimization via Brent line minimization.

LITEST subroutine ap.F

Test the model of Li^+ and 8 water molecules in the literature for sorted-AP.

LOWDISP subroutine display.F

Displays the low-level Hessian calculated for a geometry optimization.

LSAME subroutine lib.F

Returns .TRUE. if CA is the same letter as CB regardless of case.

LUDCMP subroutine ohooks.F

Conducts LU decomposition on a matrix.

LUBKSB subroutine ohooks.F

Back substitutes into an LU decomposed matrix to find a solution to a linear equation.

MG98OUTE subroutine ehooks.F

Extracts multiple energies from either a *Gaussian* formatted checkpoint file.

MINLBFGS subroutine lbfgs.F

Performs the limited memory BFGS quasi-newton nonlinear optimization.

MAXTRIXJ12 subroutine flexbound.F

Determine the coulomb interaction term in the QEQRG, QEQBT, and EEM charge calculations in the two-state flexible-boundary treatment.

MLDHOOK subroutine dynamics.F

Serves as a front end for dynamics calculations.

MLEHOOK subroutine ehooks.F

Serves as a front end for energy calculations.

MLGHOOK subroutine ghooks.F

Serves as a front end for gradient and energy calculations.

MLHEDR subroutine display.F

Prints out the program header in the output file.

MLHHOOK subroutine hhooks.F

Serves as a front end for Hessian calculations.

MLOHOOK subroutine ohooks.F

Serves as a front end for all optimization algorithms.

MMARGONTEST subroutine ap.F

Calculates the gradient and potential energy of the argon system at the dual-MM level.

MNBRAK subroutine ohooks.F

Brackets a minimum of a 1-dimensional function with three points.

MOLOUTBOX subroutine dynamics.F

Finds a molecule with center of mass outside the periodic box.

MPROGEHK subroutine ehooks.F

Carries out an energy call with the specified electronic structure program, extracting multiple energies.

MTOLTM subroutine hhooks.F

Converts a symmetric matrix to a 1-dimensional array containing the lower triangular portion of the matrix.

MUVSPX subroutine flexbound.F

Calculate the chemical potentials as a function of the PS charge in the two-state flexible-boundary treatment.

MXLNEQ subroutine ohooks.F

Calculates the inverse of a matrix.

NEWT subroutine ohooks.F

Carries out Newton-Raphson optimization with Brent line minimization using either a high-level Hessian, a low-level Hessian, or a scaled unit matrix, kept frozen when not recalculated.

NEWT2 subroutine ohooks.F

Carries out Newton-Raphson optimization with Brent line minimization using either a high-level Hessian, a low level Hessian, or a scaled unit matrix with either BFGS or DFP updates, when not recalculated.

NUMFMT module module.F

Contains information on the formats of numbers in the GAUSSIAN94, GAUSSIAN98 and GAUSSIAN formatted checkpoint files.

ONIOMXS subroutine ap.F

Calculates the energy and gradients for the ONIOM-XS method.

OPTDISP subroutine ohooks.F

Displays a step in the optimization.

OPTEHK subroutine ohooks.F

Gets the appropriate energy during an optimization.

OPTGHK subroutine ohooks.F

Gets the appropriate gradient and energy during an optimization.

OPTHHK subroutine ohooks.F

Gets the appropriate Hessian (and possibly gradient and energy, as well) during an optimization.

OPTIMIZE module module.F

Contains COMMON block variables used during optimization.

ORCAINP subroutine orca.F

Creates input files for ORCA.

ORCAOUTE subroutine orca.F

Reads the energy from an ORCA output file.

ORCAOUTG subroutine orca.F

Reads the gradient from an ORCA output gradient file.

ORCAOUTH subroutine orca.F

Reads the Hessian from an ORCA output Hessian file.

ORCAOUTO subroutine orca.F

Reads the optimized geometry from an ORCA output file.

ORCAOUTPOT subroutine eepolar.F

Reads the potential at the SS atoms from an ORCA output file.

ORCASTINP subroutine orca.F

Writes an ORCA input file for the calculations of the capped small system in presence of background point charges.

ORCASTPOTINP subroutine eepolar.F

Writes an ORCA input file for the calculations of electrostatic potentials at the SS atoms.

PARTITION subroutine ap.F

Calls the subroutines for adaptive partitioining treatments.

PERMUTEDAP subroutine ap.F

Performs permuted-AP simulations.

PERMUTEDCOMBINATION recursive sbroutine ap.F

Finds all permuations for a given time step in the permuted AP simulations.

PERMUTEDGRADIENT subroutine ap.F

Calculates the gradient and the energy for permuted-AP simulations.

PERMUTEDSMOOTHING subroutine ap.F

Calculates the smoothing fuction for permuted-AP simulations.

PERMUTEDSMOOTHING2 subroutine ap.F

Calculates the smoothing fuction for permuted-AP simulations.

POLAREECH subroutine eepolar.F

Control the iteratively procedure to do the charge equalization.

POLGCHARGCAL subroutine flexbound.F

Calculate the atomic partial charges for the polarized atoms in the two-state flexible-boundary treatment.

PRJFC subroutine ehooks.F

Calculates the projected force constant matrix.

PROGCPE subroutine flexbound.F

Calculate the electric potential produced by one oxidation state of the PS on the SS atoms in the two-state flexible-boundary treatment.

PROGDEF module module.F

Contains the default electronic structure programs called by QMMM. Automatically the default program is set to *Gaussian*, but the user may alter some or all to another supported electronic structure package.

PROGEHK subroutine ehooks.F

Carries out a single-point-energy call to the specified electronic structure program.

PROGESPP subroutine eepolar.F

Carries out a single-point-energy call to calculate the ESP charges for the CPS embedded in the ackground point charges.

PROGESTHK subroutine cutoff.F

Gets a single-point-energy for the small system in the presence of background charges with the specified electronic structure package.

PROGESTHK1 subroutine ehooks.F

Carries out a single-point-energy call for the small system in the presence of all background charges with the specified electronic-structure package.

PROGGHK subroutine ghooks.F

Carries out a gradient call to the specified electronic structure program.

PROGGSTHK subroutine cutoff.F

Gets a single point gradient for the small system in the presence of background charges with the specified electronic structure package.

PROGGSTHK1 subroutine ghooks.F

Gets a single-point-gradient for the small system in the presence of all background charges with the specified electronic-structure package.

PROGHHK subroutine hhooks.F

Carries out a Hessian call to the specified electronic structure program.

PROGHSTHK subroutine hhooks.F

Gets a single point Hessian for the small system in the presence of background charges with the specified electronic structure package.

PROGMM0EHK subroutine ehooks.F

Carries out a single energy call to the specified molecular mechanics program with all MM charges on the QM atoms set to zero.

PROGMMEHK subroutine ehooks.F

Carries out a single energy call to the specified molecular mechanics program.

PROGMM0GHK subroutine ghooks.F

Carries out a gradient call to the specified molecular mechanics program with all MM charges on the QM atoms set to zero.

PROGMM0GHKMD subroutine ghooks.F

Carries out a gradient call to the specified molecular mechanics program with all MM charges on the QM atoms set to zero. Suppresses output.

PROGMMGHK subroutine ghooks.F

Carries out a gradient call to the specified molecular mechanics program.

PROGMMGHKMD subroutine dynamics.F

Carries out a gradient call to the specified molecular mechanics program. Suppresses output.

PROGMM0GHKMD subroutine dynamics.F

Carries out a gradient call to the specified molecular mechanics program. Suppresses output.

PROGMM0HHK subroutine hhooks.F

Carries out a Hessian call to the specified molecular mechanics program with all MM charges on the QM atoms set to zero.

PROGMMHHK subroutine hhooks.F

Carries out a Hessian call to the specified molecular mechanics program.

PROGMMOHK subroutine ohooks.F

Carries out an optimization call to the specified molecular mechanics program.

PROGMMSTEHK subroutine cutoff.F

Carries out a single energy call to the specified molecular mechanics program to determine the interaction energy within background point charges.

PROGMMSTEHK1 subroutine ehooks.F

Carries out a single-point-energy call to the specified molecular-mechanics program to determine the interaction energy within all background point charges.

PROGMMSTGHK subroutine cutoff.F

Carries out a single-point-gradient call to the specified molecular-mechanics program to determine the gradient due to interactions with background point charges.

PROGMMSTGHK1 subroutine ghooks.F

Carries out a single-point-gradient call to the specified molecular-mechanics program to determine the gradient due to interactions with all background point charges.

PROGMMSTHHK subroutine hhooks.F

Carries out a single Hessian call to the specified molecular mechanics program to determine the Hessian due to interactions within background point charges.

PROGOHK subroutine ohooks.F

Carries out an optimization call to the specified electronic structure program.

QEQCHK subroutine eepolar.F

Calculate the atomic charges with the external potential using the QEQRG method.

QEQCHK subroutine eepolar.F

Calculate the atomic charges with the external potential using the QEQBT method.

QMMMBGCHGEOM subroutine ehooks.F

Adds the auxiliary point charges according to the QM/MM border treatment.

QMMMCOGEOM subroutine cutoff.F

Construct the geometry of the embedded CPS when using the QM/MM cutoff.

QMMMCUTOFF subroutine cutoff.F

Determine the SS atoms that are within the cutoff radius when using the QM/MM cutoff.

QMMMEHK subroutine ehooks.F

Carries out calls to the specified electronic structure and molecular mechanics programs to determine QM/MM energy.

QMMMGHK subroutine ghooks.F

Carries out calls to the specified electronic structure and molecular mechanics programs to determine QM/MM gradient.

QMMMGHKMD subroutine dynamics.F

Carries out calls to the specified electronic structure and molecular mechanics programs to determine QM/MM gradient in MD simulations.

QMMMGSUM subroutine ghooks.F

Compute the gradient for the entire system based on the gradients for the QM and MM fragments.

QMMMHHK subroutine hhooks.F

Carries out calls to the specified electronic structure and molecular mechanics programs to determine QM/MM Hessian.

QMMMHSUM subroutine hhooks.F

Compute the Hessian for the entire system based on the Hessians for the QM and MM fragments.

QMMMSSGEOM subroutine ehooks.F

Construct the capped small system for QM/MM model.

QUICKSORT recursive subroutine ap.F

This subroutine sorts two lists that are given as input. The list is sorted in dependent of the first list from least to greatest. The second list is sorted according to the first list. As sorting algorithm the quicksort is used.

RANDOM function dynamics.F

Generate random numbers.

RATTLEINIT subroutine dynamics.F

Initialize the RATTLE algorithm. Marks all bonds to be constrained. For water, sets the desired distance to the bond length in the stardand water model (such as TIP3P or SPC, as specified by user); for the other bonds, sets the desired distance to the inital distance.

RATTLEPOSITION subroutine dynamics.F

Constrains all marked bonds to fulfill the distance requirement.

RATTLEVELOCITY subroutine dynamics.F

Adapts the velocity of constrained bonds.

RCOEF subroutine ml.F

Reads the coefficients from the COEFFS keywords in the input file.

RCONST subroutine ml.F

Reads the CONSTANT list keyword from the MULTIGEN section.

RDYNAMICS subroutine dynamics.F

Reads the keywords from the DYNAMICS section.

READ5 subroutine ml.F

Reads the QMMM input file.

READDYNCAP subroutine dynamics.F

Reads the dynamic-link list in the *DYNAMICS sections of the input for adaptive partitioning simulations.

READGROUPS subroutine ap.F

Reads the list of atoms of each group. This subroutine is needed for adaptive-partitioning simulation and SMD simulation with restraint groups.

READINSTANTMOVE subroutine dynamics.F

Reads the constraint scan parameter list in the *DYNAMICS sections of the input file.

READPATH subroutine dynamics.F

Reads and sets the position of atoms for simulations in predefined atom paths.

READRESTRAINTS subroutine dynamics.F

Reads the restraint parameter list in the *DYNAMICS sections of the input for SMD simulations.

READRESTRAINTSFILE subroutine dynamics.F

Reads the restraint parameters from file for SMD simulations.

READLN subroutine ml.F

Finds the first non-comment and non-blank line from the input file (iunit), where each line has at most 80 characters, and chop the line into words according to the space locations.

READRESTART subroutine dynamics.F

Reads the atomic coordinates, velocities, and forces from the restart file for restarting simulations.

READZEROE subroutine ap.F

Reads the list of infinite seperate energy of group. This subroutine is needed for adaptive-partitioning simulation.

RESETQMMM subroutine ap.F

Recalculate *qmmminf* variables for adaptive-partitioning QM/MM simulations.

RESTRAINTS subroutine dynamics.F

Applies *cv* and *cf* restraint on groups and atoms for SMD simulations.

RESTRAINTSLINE subroutine dynamics.F

Reads a line with restraint parameter for SMD simulations.

REXTOPT subroutine ml.F

Reads the EXTOPT list keyword from the MULTIGEN section.

RGEOM subroutine ml.F

Reads the RGEOM list keyword from the MULTIGEN section.

RIDLIST subroutine eepolar.F

Read the atomic ID list keyword. Up to five IDs are allowed per line.

RLINE subroutine ml.F

Reads the next non-blank and non-comment line of the input file.

RLIST subroutine ml.F

Reads the OPTIONS list keywords.

RMLGEN subroutine ml.F

Reads the MULTIGEN section.

RMLOPT subroutine ml.F

Reads the MLOPT list keyword from the MULTIGEN section.

RMMGEOM subroutine ml.F

Reads the geometric data from the coordinate file.

ROTCOL subroutine ohooks.F

Given the geometry coordinates for 2 axes, rotates those coordinates about the third axis by a specified angle.

ROTMOL subroutine ohooks.F

Either rotates a molecule to place appropriate atoms at the origin, on an axis, and in a plane for an optimization, or undoes these rotations afterwards.

RQMMM subroutine ml.F

Reads the QMMM section.

RSP subroutine ef.F

EISPACK diagonalization routine. Finds the eigenvalues and eigenvectors of a real symmetric packed matrix.

RTEST subroutine ml.F

Reads the TEST section.

RTESTMM subroutine ml.F

Reads the TESTMM section.

RUNMD subroutine dynamics.F

Runs molecular dynamics simulations.

RVAR function ml.F

Returns the string following a variable keyword.

RWORD subroutine ml.F

Reads the next word on a line.

SCASE subroutine gamess.F

Converts a string to low case.

SEARCH subroutine lbfgs.F

Performs a unidimensional line search based upon parabolic extrapolation and cubic interpolation using both function and gradient values.

SETROT subroutine ohooks.F

Sets the 3 rotations to be performed based on the atoms and coordinates specified in CONSTANT.

SETGROUPAFFILATION subroutine ap.F

Calculates the distance of the groups to the active-zone center for simulations with adaptive partitioning. Based on the distance, a groups is reclassified as a QM, MM, or buffer group.

SETTSEMIEMPERICAL subroutine dynamics.F

Sets a flag if semi-emperical methods are used. If the flag is set the basis set will not be written in the Orca input file. Orca does not allow basis definition in the input file for semi empirical methods.

SHIFTMOL subroutine dynamics.F

Moves molecules so that the peiodic boundary condition is fulfill.

SHIFTSMD subroutine dynamics.F

Shifts the reference position to stay in periodic box for SMD simulations with periodic boundary.

SLINEQU subroutine flexbound.F

Solve the linear equations that determine the atomic partial charges in the two-state flexible-boundary treatment.

SMOTTHINGFUNCTION subroutine ap.F

Calculates the smoothing function and its gradient for ONIOM-XS, permuted-AP and sorted-AP simulatiuons.

SORTEDAP subroutine ap.F

Calculates the energy and gradient of a QM/MM system for sorted-AP simulations.

SORTEDSMOTTHING subroutine ap.F

Calculates the smoothing function and its gradient for sorted-AP simulatiuons.

SUMDISP subroutine display.F

Displays the summary output file `ml.sum`.

T41KEYADD subroutine ehooks.F

Adds additional keywords to TINKER keyword file for computation of a single-point interaction energy within background point charges.

T41KEYADD1 subroutine ehooks.F

Adds additional keywords to TINKER keyword file for computation of a single-point interaction energy within background point charges using screened or smeared charges.

T41INP subroutine ehooks.F

Writes the TINKER input files.

T41INPXYZ subroutine dynamics.F

Writes the atomic coordinates for gradient calculations.

T41OUTE subroutine ehooks.F

Gets a single-point energy from the TINKER output file.

T41OUTG subroutine ghooks.F

Gets a single-point gradient from the TINKER output file.

T41OUTH subroutine hhooks.F

Get a single-point Hessian from the TINKER output file.

T41OUTO subroutine ohooks.F

Get an optimized geometry out of TINKER output file.

T41OUTP subroutine ehooks.F

Read the MM point charges and print the MM parameters if needed.

TEMPANNEALING subroutine dynamics.F

Calculates the temperature in annealing simulations.

TEMPCONTROL subroutine dynamics.F

Appies the Berendsen scaling factor for temperature control.

TEMPCONTROLNH subroutine dynamics.F

Applies the Nose-Hoover scaling factor for temperature control.

TESTDISP subroutine display.F

Displays the output from a TEST calculation.

TQL2 subroutine ef.F

Computes the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the QL method [H. Bowdler, R. S. Martin, C. H. Reinsch, and J. H. Wilkinson, *Num. Math.* **1968**, *11*, 293].

TQLRAT subroutine ef.F

Finds the eigenvalues of a symmetric tridiagonal matrix by the rational QL method [C. H. Reinsch, Comm. ACM 16 (1973) 689].

TRBAK3 subroutine ef.F

Forms the eigenvectors of a real symmetric matrix by back transformation of the eigenvectors of the similar symmetric tridiagonal matrix.

TRED3 subroutine ef.F

Reduces a real symmetric matrix to a symmetric tridiagonal matrix.

UPCASE function ml.F

Converts input strings to all upper case letters for case consistency.

UPDATCHARG subroutine flexbound.F

Update the charges in the two-state flexible-boundary treatment.

WORDSTR subroutine orca.F

Finds the number of the non-comment word on a line in the ORCA input.

WRITERESTART subroutine dynamics.F

Writes the atomic coordinates, velocities, and forces in the restart file.

XERBLA subroutine lib.F

Is an error handler for the LAPACK routines.

7.B. Files Required to Run QMMM

Besides the executable, three types of files that are necessary to run QMMM: the input file, the MM force field parameter file, the coordinate file, the basis set file, and shuttle scripts. All such files are required to be in the currently working directory in which the user is running the program. (See 7.D for a script that handles this requirement.) The following two subsections give the details of the basis set files and the shuttle scripts. The input file, which must be named `ml.inp`, has been described in detail in the previous chapter.

7.B.1. Basis Set Files

Some non-standard basis sets developed in our group are provided for users' convenience. These basis set files are listed below:

Basis Set:	Ref.	File Name:
pDZ+	(*152-154*)	`pdz+.gbs`
G3large	(*155*)	`g3large.gbs`
G3Xlarge	(*156*)	`g3xlarge.gbs`
Modified G3 (MG3)	(*157*)	`mg3.gbs`
MG3-semidiffuse (MG3S)	(*158*)	`mg3s.gbs`

No additional basis set files are needed if users use standard *Gaussian* basis sets.

7.B.2. Program Shuttle Scripts

QMMM makes many calls to the specified electronic structure and/or molecular mechanics packages throughout the course of a run. In order to minimize the system calls made within the program, a C-shell shuttle script has been provided for each of the electronic structure packages supported in QMMM. The only system calls made are directly to these shuttles. The usage of each shuttle script is:

`shuttle input-file output-file`.

The seven scripts in this version of QMMM are named `g03shuttle`, `gmsshuttle`, `orcashuttle`, `t41shuttle`, `Gau_External`, `Gau_External2`, and `ex_shuttle`. The scripts `g03shuttle`, `gmsshuttle`, `orcashuttle`, and `t41shuttle` are designed for calling *Gaussian*, GAMESS, ORCA, and TINKER, respectively, and if the *Gaussian* external option is desired, the scripts `Gau_External`, `Gau_External2`, and `ex_shuttle` are needed.

A. g03shuttle

Within the `g03shuttle` script, the user must modify at least two variables – the one specifying the path of *Gaussian* and the one for the system scratch directory. For example,

```
set gausspath=/usr/local/g03/g03.e01/g03
set scratchdir=/regscratch2/linh
```

The user may also alter the handling of the *Gaussian* input, output, and scratch files. It is important to keep in mind though that the formatted checkpoint file must remain after the shuttle script has finished, for the *Gaussian* output is read by QMMM from that file.

We note that a subdirectory is created within the user-specified scratch directory for the handling of scratch files; this subdirectory is named by the process ID to ensure a unique name. All *Gaussian* scratch files are stored in that subdirectory. At the end of the *Gaussian* job, the subdirectory itself is removed. This process allows the user to run more than one QMMM job at once and not worry about the removal of the scratch files from another *Gaussian* job. This may be altered at the user's discretion.

B. ex_shuttle, Gau_External, and Gau_External2

These three shuttles are required only if the *external* option in *Gaussian* is desired for performing geometry optimization. We found that different versions of *Gaussian* (*g03.b01*, *g03.c01*, *g03.d01*, and *g03.e01*, *g09.a02*, *g09.e01*, *g16.a03*, *g16.b01*, and *g16.c01*) require slightly different procedures to invoked the *external* option. More specifically, the input and output files required for *g03.d01*, *g03.e01*, *g09.a02*, *g09.e01*, *g16.a03*, *g16.b01*, and *g16.c01* versions are in the scratch directory, while those of *g03.c01* and *g03.b01* are in the directory where the *Gaussian* input files locate. Thus, we prepare different shuttle scripts for calling different versions of *Gaussian*. Depending on which *Gaussian* version the user is using, user can select corresponding scripts for his/her calculations. See also Section 8.A. Installation Instructions for more details.

Normally user does not need to modify the content of these three shuttles.

C. orcashuttle

Within the `orcashuttle` script, the user must modify the one specifying the path of ORCA. For example, suppose that the executable binary files for ORCA are located in `~/orca/bin`; then one should specify the following in the `orcashuttle` script:

```
set orcapath = ~/orca/bin
```

D. orcapotshuttle

Within the `orcapotshuttle` script, the user must modify the one specifying the path of ORCA. For example, suppose that the executable binary files for ORCA are located in `~/orca/bin`; then one should specify the following in the `orcapotshuttle` script:

```
set orcapath = ~/orca/bin
```

This script first runs an `orca` job, and then it calculates the potential at the SS atoms using the command

`orca_vpot orca.gbw orca.scfp.tmp orca.pot.xyz orca.pot`

E. gmsshuttle

If user has his/her own script to run GAMESS for pure QM jobs, likely the `rungms` script that was provided in the GAMESS distribution (perhaps with some modification to work with your file system), the user should go to the "`Your Own Rungms`" section and follow the instructions there. In particular, user needs to modify his/her own `rungms` script by

- deleting the lines of setting `datpath`,
- adding a line at the end of the `rungms` script to move the `#.dat` file to the QMMM working directory after the GAMESS job is done.

Next, the user should uncomment the next lines until (not including) "`End of Script`".

If the user uses a locally created script to run pure QM jobs WITH GAMESS, e.g., on the Altix, Regatta, and Netfinity machines at the Supercomputing Institute of the University of Minnesota, where such a script has been provided by the computer administrators, the user should go to the

"`Institute Defined Rungms`" section and follow the instructions there. In particular, the user should uncomment the two lines containing `gmspath` and `datpath` listed below:

```
#   set gmspath=~/bin
#   set datpath=~/scr
```

If the `gmspath` and `datapath` are different from those given above, the user should revise them to the correct location. Next, the user should follow the instructions in the script to uncomment certain lines until (not including) "`End of Script`".

F. t41shuttle

Within the `t41shuttle` script, the user must modify at least one variable – the one specifying the path of TINKER executable binary files (see also section 4.P.4. Calling TINKER). For example, suppose that the executable binary files for TINKER 4.2 are located in `~/tinker4.2/tinker/bin`; then one should specify the following in the `t41shuttle` script:

```
set tinkerpath = ~/tinker4.2/tinker/bin
```

We have tested the current version of QMMM with five versions of TINKER: version 3.5, version 4.1, version 4.2, version 5.1, and version 6.3 and the test runs in the current version of QMMM are made to call a modified TINKER 6.3, which is included in the distribution package. However, it is possible (and very straightforward) to make QMMM calls other versions of TINKER without modifying the QMMM code, provided the input and output formats used by the other version of TINKER are the same as TINKER 6.3.[2] The procedure is very simple, and users can do it

[2] In the TINKER web site, there is a statement that "Please note that as with prior new releases, version x.x is neither backward nor forward compatible with earlier versions of TINKER. For example, earlier versions of parameter files should not be used with version x.x executables and vice versa." This is not always true, and in fact, this statement always stays on the web site, regardless whether there are changes in the current version compared with the previous version. (The situation in principle depends on which versions the user is comparing; for versions 4.1 4.2, 5.1, and 6.3, we found only a little change, which is described on section 4.P.4.) Users could easily verify whether there have indeed some changes to the input and output formats by running new TINKER calculations with old input files and comparing the outputs.

themselves. All one needs to do is to change in the t41 script the line that we mentioned above, i.e., the line that specifies the directory where TINKER executables locates. In general, supposed that the executable binary files for TINKER version x.y are located in `~/tinkerx.y/bin`, one can specify in the t41 script `t41shuttle`:

```
set tinkerpath = ~/tinkerx.y/bin
```

7.B.3. Molecular Mechanics Force Field Parameter File

A molecular mechanics force field parameter file `t41.prm` is needed for MM and QM/MM calculations. The molecular mechanics program used by QMMM will employ this file as the MM parameter file. One can usually use the original parameter file of TINKER without modification; however, sometimes some parameters for valence interactions between QM atoms and/or between HL and QM atoms are missing, and users should supply these parameters either by modifying the parameter file, or by supplying them through MM keywords. The valence interactions between QM atoms cancel out (see Section 4.F for details), and thus users can give any values for these without affecting the results. The interactions between QM and HL atoms do not cancel exactly, and users can "borrow" parameters from similar atom types, as is often done in QM/MM studies.

Currently, the following force fields in the TINKER implementation have been tested:

TINKER 3.5: MM3(*110-112*)

TINKER 4.1: Amber96,(*107*) Charmm27,(*108*) MM3,(*110-112*) and OPLS-AA(*109, 125, 126, 128, 129*)

TINKER 4.2: Amber96,(*107*) Charmm27,(*108*) MM3,(*110-112*) and OPLS-AA(*109, 125, 126, 128, 129*)

TINKER 6.3: Amber96,(*107*) Charmm27,(*108*) MM3,(*110-112*) and OPLS-AA(*109, 125, 126, 128, 129*)

7.B.4. Coordinate File

For QM/MM and/or MM calculations, one needs to supply for the real system not only the Cartesian coordinates but also information about valence connectivity. These are given in an input file `t41.crd` that is called coordinate file.

For calling TINKER 4.2 (or 6.3), the coordinate file takes a form slightly modified from the TINKER format. For example,

22 Alanine Dipeptide // CHARMM22 C5 Minimum

1 C CT3 –2.249880 –0.851680 –0.058940 27 2 4 5 6

2 C C –0.786160 –1.071640 –0.041920 20 1 3 7

...

22 H HA 3.670130 3.449260 –0.484700 1 18

END

Descriptions are as follows:

The first field at the first line gives the number of atoms, which is 22 in this example (this is required). The rest at the first line are comments and will be ignored by QMMM.

The second to 23rd lines are atom descriptions. Each line is for one atom, contains no more than 80 characters, and is organized as follows:

index symbol atom-type-name X Y Z atom-type connectivities

Here, *index* is the identification number, *symbol* is the atom symbol, *atom-type-name* is the name of atom type defined in the parameter file, *X*, *Y*, and *Z* are Cartesian coordinates (in Å), *atom-type* is the identification number corresponding to the *atom-type-name*, and *connectivities* are lists of atoms to which the current atom is bonded.

The last line is the keyword END (required).

Compared with the TINKER coordinate file, the only differences are the adding of the second field (*symbol*) in each line for atom description, and the adding of the keyword **END** at the end.

A special and important requirement here is that the user must set the atom index starting from 1 and increasing sequentially up to the total number of atoms.

The QMMM program will read from the file `t41.crd` the indices for the real atoms, read from the `ml.inp` file the list for QM atoms, the list for M1 atoms, and the options for setting up the QM/MM boundary. Based on this information, the QMMM program will automatically generate the other set of indices for the CPS, if ME scheme is used, or for the embedded-CPS, if an electronic embedding scheme is used.

7.B.5. Group File

The file *group.log* contains the definition of groups and their zeros of energy in hartree. It is needed for adaptive-partitioning QM/MM simulations. The format of the file can be explained by the following example:

```
6    11
# group id, atom id       (total 6 groups, 11 atoms)
2    1        # atom 1 in group 2
3    4        # atom 4 in group 3
2    3        # atom 3 in group 2
1    2        # atom 2 in group 1
1    5    8   # atom 5, 6, 7 and 8 in group 1
4    9        # atom 2 in group 1
5    10       # atom 2 in group 1
6    11       # atom 2 in group 1
END
# group id, qm energy, mm energy
2     -120.20123456    -0.00030021    # energy for group 2
1    -76.15000123     0.00000000         # energy for group 1
3     NO    0.00000022    # group 3 is never QM
E   4   6     -10.27659999     0.00025678    # groups 4 to 6 have the same energies
END
# duo super groups
1 2     -197.29765432     -0.00044652    # super-group by bonding groups 1 and 2
END
# triple super group
4 5 6     -31.58123456   0.00061333   # group 4 and 5, and 5 and 6 are bonded
END
```

The first line provides the total number of groups and the total number of atoms.

The second line is a comment line. Note: anything after # is considered comment and will not be read by the QMMM program.

The next 9 lines are the block of group definition, ended by a single line with the keyword END. Group 1 is formed by Atoms 2, 5 to 8, Group 2 by Atoms 1 and 3, Group 3 by Atom 4, Group 4 by Atom 9, Group 5 by Atom 11, and Group 6 by Atom 12.

The next line is again a comment line, which is followed by a block gives the zeros of QM and MM energies for the individual groups. The block is ended by a single line of the keyword END. Each line in the block begins with an integer number, which is the group index, or with a keyword "E" followed by two integers, which specify the range of group index. The keyword "E" indicates that the groups in the specified range have the same zeros of energy, which is typically the case for solvent molecules. After the group index(es) is the QM zero of energy of the specified group. If the group is never intended to be a QM group (even when its position is within the active zone range), replace the QM zero of energy by the keyword "NO". The last number is the MM zero of energy. If the zero of energy is not given, it will be set to zero.

The next block gives the zeros of energy for super-groups that are formed by two groups covalently bonded to each other; the supper-groups are needed for treating groups that are molecular fragments. See Section 4. M. Adaptive Partitioning for more background information. Each line begins with two integers, which are the group indexes of the two groups that are bonded to produce the super-group, followed by the QM and MM zeros of energy of the super-group. The block is end by a single line of the keyword "END".

The next block gives the zeros of energy for super-groups that are formed by three groups covalently bonded to each other. Each line begins with three integers, which are the group indexes of the three groups that are bonded to produce the super-group, followed by the QM and MM zeros of energy of the super-group. Note: the second group index must correspond to the group that bonds to the other two groups; the first and third groups do not have to bond to each other. The block is end by a single line of the keyword "END".

Currently, the QMMM program is limited to super-groups that are made of three fragmental groups.

Based on our experience, it is recommended that the zeros of energy are given up to 10^{-8} hartree accuracy.

7.B.6. Restart File

The *dynamics.restart* is created during an MD simulation, which contains the up-to-date atomic coordinates, velocities, and forces for an MD simulation. It can be used to restart a simulation job that ends abnormally or continue a simulation that ends normally. To use the restart file, user should copy the *dynamics.restart* file to the directory that contains the input files, rename it with another file name, e.g., *restart.in*, and specified the new file name in the *DYNAMICS section of the input file *ml.inp* by the **RESTART** keyword:

RESTART restart.in

7.B.7. Path File

The path file specifies a predefined trajectory for selected atoms in a simulation. The name of the file has to be specified by the **PATH** keyword in *DYNAMICS section of the input file *ml.inp*.

During an MD simulation, after the positions and velocities of all atoms are computed for the given time step, the coordinates of those selected atoms will be replaced by the coordinates read from the path file. Such an option is mainly for debugging use, but it might also be useful in free-energy calculations employing the thermodynamics perturbations theory. The predefined trajectory in the path file is divided into blocks, each block corresponding to one time step in the MD simulations. Each block contains multiply lines specifying the atom index and the desired Cartesian coordinates in Å and is terminated by one line containing only one keyword **END**. No blank line is allowed. An example is given below (the meaning of each line is given by the comment followed by #):

```
1    2.0    3.0    4.0      # coordinates of atom 1 in step 1 set to (2.0,3.0,4.0)
3    4.0    5.0    4.0      # coordinates of atom 3 in step 1 set to (4.0,5.0,4.0)
END
1    3.0    3.0    3.0      # coordinates of atom 1 in step 2 set to (3.0,3.0,3.0)
END
2   -1.0   -2.0   -1.0      # coordinates of atom 2 in step 3 set to (-1.0, -2.0,-1.0)
1    4.0    3.0    2.0      # coordinates of atom 1 in step 3 set to (4.0,3.0,2.0)
END
```

The above example specifies the predefined coordinates of atoms 1 and 3 for time step 1, of atom 1 for time step 2, and of atoms 1 and 2 in time step 3. After the third time step, since no predefined coordinates are provided, the MD trajectory will be propogated as usual. (Warning: The QMMM program does not check if predefined coordinates of an atom is reasonable or not; it is the user's responsibility to make sure that atoms do not crash into each other and bonds are not overstretched.)

7.B.8. Gau_ext.acc Script

This script is associated with the newly modified `ex_shuttle`, `g03shuttle`, and `.ml` scripts (in *QMMM 2018*), and it can be used to accelerate QM/MM optimization using the *Gaussian* external optimizer (see the revision history of *QMMM 2018*). Using this script is the recommended option when the *Gaussian* external optimizer is used. One needs to copy the `Gau_ext.acc` script in the `script` directory to the job directory where the *QMMM* input files are located and execute this script instead of the `.ml` script. (Note that the `.ml` script is still required to be located in the job directory because the `Gau_ext.acc` script calls the `.ml` script.)

Within the `Gau_ext.acc` script, the user may need to modify one variable – the one specifying the (case insensitive) indicator of the line after which the script would insert the "`guess=read`" keyword in the `.dat` input file for an optimization calculation. The purpose of setting this variable is to ask the user for an appriate place where the "`guess=read`" keyword should be placed. For example, suppose that the "`guess=read`" keyword should be placed below the line containing "SCF="; then one can specify the following in the `Gau_ext.acc` script:

```
set ind = "SCF="
```

Since the variable is case insensitive, one can also specify the following in the `Gau_ext.acc` script:

```
set ind = "scf="
```

7.C. Files Created During an QMMM Run

Several files are created in the process of running QMMM.

7.C.1. The Output File ml.out:

The output file is created with the name `ml.out`. The first portion of the file summarizes the options specified in the input file. The remainder details the results of the external optimization, each step in a QMMM optimization, all energies, gradients, and Hessians calculated, and the dynamics results including time step, temperature, and total, kinetic, and potential energy. All values are clearly identified within the output. If not noted otherwise, coordinates are given in ångstrom, and energies in hartree.

7.C.2. The Electronic Structure Program Input and Output Files

A. *Gaussian*

Currently there are seven files created that fall in this category. Each is described below. A line is also written to standard error for each electronic structure job run by QMMM.

`g03.inp`

This is the input file for *Gaussian* for all energy, gradient, and Hessian calculations made with the electronic structure package. The input file is rewritten for each call to the electronic structure program. The method and basis set are those required for the current QMMM calculation. The two options specified in every input file are *FChk=All* (in order to read the *Gaussian* output) and *NoSymm* (in order to avoid calculation failures due to a changing of the molecular symmetry). The three types of calculations that may be specified in these files are *SP*, *Force*, and *Freq* to obtain the energy, gradient, and Hessian respectively. The option *Force=EnOnly* is always chosen for those methods that do not have analytic gradients, due to the fact that *Gaussian* fails otherwise.

The user may specify additional options in the QMMM input file in a format accepted by *Gaussian*. In addition, the only non-standard basis sets able to be used in any calculation are the three listed in 7.B.1. They should be named in the QMMM input file as pDZ+, G3large, and MG3; otherwise QMMM will not recognize the basis.

g03.out

These are the output files resulting from each *Gaussian* run of `g03.inp`. As is true for the input files, these two files are overwritten with each run of the electronic structure package. They are not used in running QMMM. However, should the user encounter an error reading the checkpoint file, examination of these output files may prove useful in identifying the problem.

extopt.inp

This is the input file for the geometry optimization with an external electronic structure program. While this file and `extopt.out` (described later) are both technically going to be *Gaussian* files, these two files have unique names so that they will not be overwritten, allowing the user to examine them at a later time. As was true for `g03.inp`, the options *FChk=All* and *NoSymm* are always activated. And the memory allocation specified in Link0 is 500MB (although this may be altered as well in the module FILES). While for the other input files, the user input options are not necessary, the user *must* specify options for an external geometry optimization (*Opt* at the very minimum). Otherwise *Gaussian* will not carry out an optimization.

extopt.out

Unlike the other output files from *Gaussian*, QMMM does access this file to ensure that the optimization was successful before reading data from `Test.FChk`. But the user may want to examine the results of the geometry as well; thus this file will not be overwritten.

ml.charg

This is the output file for the polarized-embedding and flexible-boundary calculations. The file contains the detailed results of each step in the charge equalization calculations, including the charges, electrostatic potentials at the SS atoms, the energy of the embedded-QM calculations, etc.

extopt.charg

This is the output file that contains, when employing the polarized-embedding and flexible-boundary schemes, the charges on the SS atoms, in the geometry optimization using the Gaussian optimizer via the external option.

Test.FChk

This is the formatted checkpoint file created by *Gaussian*. It is integral to the operation of QMMM in that all energies, gradients, Hessians, and optimized geometries output by the electronic structure packages are read from this file.

tmp.chk

This is the temporary *Gaussian* checkpoint file to be read by the next *Gaussian* energy and gradient calculation.

B. ORCA

Currently there are five used files fall in this category. Each is described below.

orca.inp

This is the input file for all energy, gradient, and Hessian calculations made with the electronic-structure package ORCA. The input file is rewritten for each call to the electronic structure program. The method and basis set are those required for the current QMMM calculation.

orca.out

These are the output files resulting from each ORCA run of `orca.inp`. As is true for the input files, these two files are overwritten with each run of the electronic structure package. They are not used in running QMMM. However, should the user encounter an error reading the checkpoint file, examination of these output files may prove useful in identifying the problem.

orca.engrad

These are the output files of the gradient from ORCA gradient run of `orca.inp`.

orca.hess

These are the output files of the Hessian from ORCA Hessian run of `orca.inp`.

orca.pot

This is the output file that contains the electrostatic potentials at the SS atomic centers calculated by ORCA.

orca.pot.xyz

This is the ORCA input file that contains the coordinates of the SS atomic centers, at which the electrostatic potential are to be calculated.

orca.xyz

These are the output files of the optimization from ORCA optimized run of `orca.inp`.

C. GAMESS

Currently there are three used files fall in this category. Each is described below.

gms.inp

This is the input file for all energy, gradient, and Hessian calculations made with the electronic-structure package GAMESS. The input file is rewritten for each call to the electronic structure program. The method and basis set are those required for the current QMMM calculation.

gms.out

These are the output files resulting from each GAMESS run of `gms.inp`. As is true for the input files, these two files are overwritten with each run of the electronic structure package. They are not used in running QMMM. However, should the user encounter an error reading the checkpoint file, examination of these output files may prove useful in identifying the problem.

gms.dat

This is the formatted checkpoint file created by GAMESS. It is integral to the operation of QMMM in that part of energies and all gradients, Hessians, and optimized geometries output by the electronic structure packages are read from this file.

7.C.3. The Molecular Mechanics Program Input and Output Files

Currently there are four files created that fall under this category. Each will be described below.

t41.xyz

This is the input Cartesian coordinate file for TINKER 4.2 for all energy, gradient, Hessian, and optimization calculations.

t41.key

This is the input keyword file for TINKER 4.2 for all energy, gradient, and Hessian, and optimization calculations

t41.prm

This is the input parameter file for TINKER 4.2 for all energy, gradient, Hessian, and optimization calculations

t41.out

This is the output file for TINKER 4.2 for all energy, gradient, Hessian, and optimization calculations.

7.C.4. The Summary Output File: `ml.sum`

Whether a summary output file is printed is specified by the switch keyword PRSUM/NOPRSUM in the MULTIGEN section. The summary output file is created under the name `ml.sum`. It lists the final geometry, gradients and Hessians obtained from QMMM calculations.

7.C.5. The MOLDEN Input File: `.xyz`

A file for MOLDEN visualization is generated automatically during optimization if multilevel optimizer is invoked (except for the case when the *Gaussian*'s optimizer is called).

7.C.6. The DYNAMICS Input and Output Files

`dynamics.restart`

The file contains the atomic coordinates, velocities, and forces, which are needed to restart a simulation from the time step at which the restart file was written.

`trajectory.arc`

The file contains the trajectory of the atoms in the TINKER format.

`smdDummy.arc`

The file contains the trajectory of the reference point in SMD simulations in the TINKER format.

7.D. The C Shell Scripts: `run.ml` and `universal_run`

While QMMM may be run by simply typing *`qmmm.exe`*, there is a fair amount of file bookkeeping that is necessary. The shuttles and basis set files must be in the current working directory. And in addition since QMMM's input and output always use the same filenames, sequential runs of QMMM require a large amount of file handling. Thus a sample script `run.ml` has been provided to illustrate these procedures.

Within this script the user *must* alter two lines: the first line in the script where the name is set and the initial line of the *foreach* loop. The list that the *foreach* loop parses should contain the prefixes of each QMMM input file the user desires to run. Each of these input files should be named *`prefix`*`.dat`.

The usage of the script is *`run.ml directory`*, where `directory` is the location of the input files to be run. If this argument is omitted, the input files are assumed to be in the current working directory. With the script, the only file handling required of the user is to have the QMMM input files in the directory specified. The script then copies the shuttle scripts and the

basis set files from the directory containing the QMMM executable to the current working directory. Then for each *prefix* listed in *run.ml*, prefix.dat is copied to ml.inp and QMMM is called. At the end of each run of QMMM the output files are renamed as follows:

ml.out ⇒ prefix.out

extopt.out ⇒ prefix.extopt

g03.inp ⇒ prefix.g03inp

g03.out ⇒ prefix.g03out

Test.FChk ⇒ prefix.fchk

In this way the user may check the QMMM output, the external optimization output, and the final *Gaussian* input, output, and formatted checkpoint files for each QMMM run.

Note: If ORCA is selected, user can use the following lines to replace the lines for handling *Gaussian* files.

orca.inp ⇒ prefix.orcainp

orca.out ⇒ prefix.orcaout

Note: If GAMESS is selected, user can use the following lines to replace the lines for handling *Gaussian* files.

gms.inp ⇒ prefix.gmsinp

gms.out ⇒ prefix.gmsout

There is also one additional file created by the script for each *prefix* in the list: prefix.jobs. This file contains all the standard error messages, which include a list of all the electronic structure package jobs called. Also the final file line gives the system and user time spent on that QMMM run.

At the end of the script when all the QMMM calls have been made, all files are removed with the exception of those beginning with one of the *prefixes.*

The user may alter the handling of the input and output files but should keep in mind which files are required for the running of QMMM. Run.ml is provided as a template script that

illustrates these practices. Further examples of handling of the input and output files are provided in the test runs `test#.ml` scripts (see also Section 8.B. The qmmm Test Suite).

The shell script `universal_run` is provided to facilitate the execution of molecular simulations, although it can also be used for signle-point calculations and geometry optimizations. The usage of the script is as follows:

```
universal_run prefix [options]
```

The script requires the input file *`prefix.dat`* to be in the directory where the script was called. The script will create a new working directory with the name *`prefix`* (if a directory with the given name already exists, a subdirectory *`prefix/prefix`* will be created). All required files are copied into the working directory (or subdirectory) before the simulation can be started. Errors and general output are written into the file *`prefix.job`* in the working directory (or subdirectory).

The options are described as follows:

-c specifying the coordinate file, e.g., *-c mycoordinate.crd* (if the name of the coordinate file is *`prefix.crd`*, this option can be skipped). If the coordinate file is not needed, use *-c no*.

-g specifying the file containing information for all groups in the adaptive-partitioning QM/MM schemes, e.g., *-g mygroup.log* (if the name of the file is group.log, this option can be skipped). If the coordinate file is not needed, use *-g no* or omit this option.

-nocleanup suppressing the deletion of the intermediate files after a job is finished. This option is often used when searching for errors in a simulation.

-p specifying the parameter file, e.g., *-p myparameter.prm* (if the name of the coordinate file is *`prefix.prm`*, this option can be skipped.) If the parameter file is not needed, use *-p no*.

-r specifying the file containing information for restarting an MD simulation, e.g., *-r myrestart* (if the name of the file is *`dynamics.restart`*, this option can be skipped). If the restart file is not needed, use *-r no*.

-s specifying another file to be copied into the working directory. Examples of those files include the checkpoint file used in the *Gaussian* calculations. Omit this option if not needed.

-w specifying the name of the working directory other than *`prefix`*, e.g., *-w myworkdir*.

After the job is finished, all intermediate files will be deleted since they were needed for execution but not for the final results. To suppress that, one can use the option *-nocleanup*.

7.E. Utility Tools for Program Development

Several utility tools are provided for people who want to further develop the QMMM program.

7.E.1. The UNIX Script updateqmmmpath

The script is to help update the version of QMMM in the `.ml` scripts in all test runs and in some scripts.

7.E.2. The UNIX Scripts updatetestrun and updatetesto

These scripts prepare the new `testrun` and `testo` directories for distribution, respectively. In particular, two directories `newtestrun` and `newtesto` are created, which are to be renamed to `testrun` and `testo` manually.

7.E.3. The UNIX Scripts for Checking Test Runs

Seven scripts make comparisons of the results for the test runs between obtained by the user/developer and distributed in the QMMM package. In particular, `checktestrun_tinker`, `checktestrun_g03_qm`, `checktestrun_g03_qmmm`, `checktestrun_gms_qm`, `checktestrun_gms_qmmm`, `checktestrun_orca_qm`, and `checktestrun_orca_qmmm` are provided for comparisons for TINKER, *Gaussian* QM, *Gaussian* QM/MM, GAMESS QM, GAMESS QM/MM, ORCA QM, and ORCA QM/MM test runs.

7.E.4. The F95 Programs for Analyzing MD Runs

Two F95 program source files *averageT.f* and *extractEP.f* are provided. They can be compiled by using any FORTRAN compiler that compiles F95 codes. For example:

```
gfortran averageT.f -o averageT <Return>
gfortran extractEP.f -o extractEP <Return>
```

The programs are useful tools for analyzing the results by MD runs.

The `extractEP` program reads in the QMMM output file (specified by user) and writes the thermodynamic information into separate files: total energy in *Energy.dat*, momentum in *Momentum.dat*, temperature in *Temperature.dat*, volume in *Volume.dat*, kinetic energy in *E_kinetic.dat*, and potential energy in *E_potential.dat*.

The `averageT` program reads in the temperature as a function of simulation step (recorded in the file *Tepearture.dat*), computes the average temperature as a function of simulation step, and writes the results in a file *AverT.dat*.

Chapter Eight

8

8. Installing and Using QMMM

A step-by-step procedure for installing QMMM on a Unix computer and testing it is given here. Compilation of the code can be accomplished with the shell script. The test runs illustrate the proper way in which to use this program.

8.A. Installation Instructions

Step 1: Preparation

A. Install QM and MM Packages

Before installing QMMM, users should have installed *Gaussian* (GAMESS or ORCA) and TINKER. Please visit the web sites of *Gaussian* (http://www.gaussian.com/), GAMESS (http://www.msg.ameslab.gov/GAMESS/GAMESS.html), ORCA (http://www.thch.uni-bonn.de/tc/orca/), and TINKER (http://dasher.wustl.edu/tinker/) for information about obtaining and installing these programs.

In principle, modifications to the TINKER program are not needed. However, we suggest that users make a small modification to the output formats for the gradient in the `testgrad` subroutine and for the Hessian elements in the `testhess` subroutine. The current output format for used by TINKER in the `testgrad` subroutine is (*3F16.8*) if the *digits = 8* keyword is specified, and if the *digits = 6* and *digits = 4* keywords are specified, the corresponding formats are (*3F14.6*) and (*3F12.4*), respectively. Our recommendation is to change the formats to (*3(E16.8,1X)*), (*3(E14.6,1X)*), and (*3(E12.4,1X)*), respectively. The use of scientific format helps to handle special cases where the gradient and Hessian elements are very large, and the insertion of a space between the numbers makes the output more readable. Similarly, we recommend users to change the output format for used by TINKER in the `testhess` subroutine from *(4F16.8)* to *(4(E16.8,1X))* if the *digits = 8* keyword is specified, from from *(5F14.6)* to *(5(E14.6,1X))* if the

digits = 6 keyword is specified, and from *(6F12.4)* to *(6(E12.4,1X))* if the *digits = 4* keyword is specified. We have provided users in the `script` directory two scripts (`modtinker` and `pswitch`) to make such a modification. Users can simply copy these two scripts into the same directory where TINKER source files are placed, and run the `modtinker` script.

It is strongly suggested that user tests these programs to make sure that they work properly before install QMMM. In addition, if you plan to use *Gaussian* as the electronic-structure package, you need to find out which *Gaussian* version (b01, c01, d01, or e01) are used. You also need to note down the directories containing executable files of these programs; if you are not sure, consult your system administrator. You also need to find out and note down the scratch directory that you are using.

B. C Shell Start-up Configuration

It is strongly suggested that user check his/her `.cshrc` file to make sure the current directory is included in the path. If not, please include it, which can be done by adding a line

```
set path=(. $path)
```

in the `.cshrc` file. After edit the `.cshrc` file, enter a C shell by typing

csh <Return>

and then type

echo $path <Return>

to check if the current directory has been included in the path.

Sometimes a machine, as we find out, actually provides a TC shell by making a link: `csh->tcsh`; in such a case, user should check the `.tcshrc` file to make sure that the current directory is included in the path.

C. Compiler

The QMMM program was written in Fortran 90. The user needs a Fortran 90 compiler to compile the codes. Generally speaking, any Fortran 90 compiler that works with the unix or linux operating system of the on user's computer should do the job. We suggest using a reliable compiler that works best with the specific type of machine and operating system at hand, e.g., the compiler xlf for an IBM computer running AIX. If you are not sure, please consult your system administrator for advice. (See also Section 9. Computers, Operating Systems, and Fortran Compilers for machine types where QMMM has been tested.) We recommend users to set higher precision as compiler options.

D. PERL

If *Gaussian* is selected as the electronic-structure package and user wants to do geometry optimization employing its optimizer through the *external* option (see Section 5.D. Optimization with Gaussian's Optimizer), user should make sure that PERL is available (which is often the case). The PERL is required because the `Gau_External` (for the b01 and c01 versions) and `Gau_External2` (for the d01 and e01 versions) scripts in QMMM were written in PERL. If you are not using *Gaussian* as the electronic-structure package or you do not wants to do geometry optimization employing *Gaussian* optimizer through the *external* option, you do not need to install PERL. PERL is often installed at `/usr/bin` or `/usr/local/bin`; ask your computer administrator is you are not sure.

If you have done the above preparation work, you can now proceed to the installation of QMMM.

Step 2: Extract Files

The QMMM program should have been obtained in the tar format with the following file name: `qmmm`*x*`.tar.gz` or `qmmm`*x*`.`*y*`.tar.gz` or `qmmm`*x*`.`*y*`.`*z*`.tar.gz`, where *x*, *y*, and *z* specify the version. For example, *x* = 1, *y* = 3, and *z* = 5 together indicate version 1.3.5. The current version is version 2018.

This file should be placed in the directory in which the user wishes to install QMMM, and then the following two commands should be executed:

gunzip qmmm2018.tar.gz <Return>
tar –xvf qmmm2018.tar <Return>

Once these two commands have been executed the directory structure on the next page should have been created, where `PDIR` indicates the parent directory, in other words, the directory in which QMMM was untarred. The `PDIR/qmmm2018` is called the QMMM home directory. Please verify that this is true.

Step 3: Verify Content

Verify that the files have been placed into the directory structure above as follows.

In the QMMM home directory:

src/	mod/	obj/	exe/
script/	basis/	testrun/	testo/
set_path.ml	doc/	tool/	tinker_QMMM/

In the `src` directory:

ml.F	module.F	main.F	display.F
dynamics.F	ehooks.F	ghooks.F	hhooks.F
ohooks.F	ef.F	freq.F	orca.F
gamess.F	cutoff.F	lib.F	gau_ext_opt.F
eepolar.F	gau_part_opt.F	lbfgs.F	flexbound.F
numhess.F	orca_ext_opt.F	ap.F	

In the `tool/mdanalysis` directory:

averageT.f	extractEP.f

In the `script` directory:

QMMMatomlist	comp_multi.ais	comp_multi_g95.lux	
comp_multi_pgi.lux		comp_multi_intel.lux	
t41shuttle	mmol	modtinker	pswitch
orcashuttle	run.ml	qmmmhess2g03	gmsshuttle
orcapotshuttle	updateqmmmpath	updatetesrun	undatetesto
createGroups.F	extract.pl	universal_run	Gau_ext.acc

In the `script/checktestrun` directory:

```
checktesrun_g03_qm                checktesrun_g03_qmmm
checktesrun_gms_qm                checktesrun_gms_qmmm
checktesrun_orca_qm               checktesrun_orca_qmmm
checktesrun_tinker
```

In the `script/g03.b01` directory:

```
ex_shuttle        g03shuttle        Gau_External
```

In the `script/g03.c01` directory:

```
ex_shuttle        g03shuttle        Gau_External
```

In the `script/g03.d01` directory:

```
ex_shuttle        g03shuttle       Gau_External    Gau_External2
```

In the `script/g03.e01` directory:

```
ex_shuttle        g03shuttle       Gau_External    Gau_External2
```

In the `script/g09.a02` directory:

```
ex_shuttle        g03shuttle       Gau_External    Gau_External2
```

In the `script/g09.e01` directory:

```
ex_shuttle        g03shuttle       Gau_External    Gau_External2
```

In the `script/g16` directory:

```
ex_shuttle        g03shuttle       Gau_External    Gau_External2
```

In the `basis` directory:

```
g3large.gbs       mg3.gbs           mg3s.gbs          pdz+.gbs
```

In the `testrun` directory:

```
g03/              orca/           tinker/         gamess/
```

In the `testrun/g03` directory:

```
qmmm/             qm/               rgall
```

In the `testrun/orca` directory:

```
qmmm/             qm/             roall
```

In the `testrun/gamess` directory:

```
qmmm/             qm/             rgmsall
```

In the `testrun/tinker` directory:

```
mm/
```

In the `testrun/tinker/mm` directory:

```
      rtmmall
```

In the `testrun/tinker/mm/test101` directory:

```
test101.crd       test101.dat       test101.prm       test101.ml
```

In the `testrun/tinker/mm/test102` directory:

```
test102.crd       test102.dat       test102.prm       test102.ml
```

…

In the `testrun/tinker/mm/test105` directory:

```
test105.crd       test105.dat       test105.prm       test105.ml
```

In the `testrun/tinker/mm/test106` directory:

```
test106.crd       test106.dat       test106.prm       test106.ml
groups.log
```

In the `testrun/tinker/mm/test107` directory:

```
test107.crd       test107.dat       test107.prm       test107.ml
test107.path
```

In the `testrun/g03/qm` directory:

```
rgqmall           rgqmall.pbs
```

In the `testrun/g03/qm/test201` directory:

```
test201.dat       test201.ml
```

In the `testrun/g03/qm/test202` directory:

```
test202.dat       test202.ml
```

…

In the `testrun/g03/qm/test210` directory:

```
test210.dat       test210.ml
```

In the `testrun/g03/qmmm` directory:

```
rgqmmmall         rgqmmmall.pbs
```

In the `testrun/g03/qmmm/test2001` directory:

```
test2001.crd      test2001.dat      test2001.prm      test2001.ml
```

In the `testrun/g03/qmmm/test2002` directory:

```
test2002.crd      test2002.dat      test2002.prm      test2002.ml
```

…

In the `testrun/g03/qmmm/test2070` directory:

```
Gau_ext.acc     test2070.crd        test2070.dat      test2070.prm
test2070.ml
```

In the `testrun/orca/qm` directory:

```
roqmall           roqmall.pbs
```

In the `testrun/orca/qm/test301` directory:

```
test301.dat       test301.ml
```

In the `testrun/orca/qm/test302` directory:

```
test302.dat       test302.ml
```

…

In the `testrun/orca/qm/test308` directory:

```
test308.dat       test308.ml
```

In the `testrun/orca/qmmm` directory:

```
roqmmmall          roqmmmall.pbs
```

In the `testrun/orca/qmmm/test3001` directory:

```
test3001.crd      test3001.dat      test3001.prm      test3001.ml
```

In the `testrun/orca/qmmm/test3002` directory:

```
test3002.crd      test3002.dat      test3002.prm      test3002.ml
```

…

In the `testrun/orca/qmmm/test3035` directory:

```
test3035.crd      test3035.dat      test3035.prm      test3035.ml
groups.log
```

In the `testrun/gamess/qm` directory:

```
rgmsqmall     rgmsqmall.pbs
```

In the `testrun/gamess/qm/test401` directory:

```
test401.dat       test401.ml
```

In the `testrun/gamess/qm/test402` directory:

```
test402.dat       test402.ml
```

…

In the `testrun/gamess/qm/test406` directory:

```
test406.dat       test406.ml
```

In the `testrun/gamess/qmmm` directory:

```
rgmsqmmmall         rgmsqmmmall.pbs
```

In the `testrun/gamess/qmmm/test4001` directory:

```
test4001.crd      test4001.dat      test4001.prm      test4001.ml
```

In the `testrun/gamess/qmmm/test4002` directory:

```
test4002.crd      test4002.dat      test4002.prm      test4002.ml
```

…

In the `testrun/gamess/qmmm/test4022` directory:

```
test4022.crd      test4022.dat      test4022.prm      test4022.ml
```

In the `testo` directory:

```
g03/              orca/           tinker/         gamess/
```

In the `testo/g03` directory:

```
qm/               qmmm/
```

In the `testo/orca` directory:

```
qm/             qmmm/
```

In the `testo/gamess` directory:

```
qm/             qmmm/
```

In the `testo/tinker` directory:

```
mm/
```

In the `testo/tinker/mm/test101` directory:

```
test101.sum       test101.out
```

In the `testo/tinker/mm/test102` directory:

```
test102.sum       test102.out
```

…

In the `testo/tinker/mm/test107` directory:

```
test107.sum       test107.out
```

In the `testo/g03/qm/test201` directory:

```
test201.out       test201.sum
```

In the `testo/g03/qm/test202` directory:

```
test202.out       test202.sum
```

…

In the `testo/g03/qm/test210` directory:

```
test210.out       test210.sum
```

In the `testo/g03/qmmm/test2001` directory:

```
test2001.sum      test2001.out
```

In the `testo/g03/qmmm/test2002` directory:

```
test2002.sum      test2002.out
```

…

In the `testo/g03/qmmm/test2070` directory:

```
test2070.sum      test2070.out      test2070.extopt
```

In the `testo/orca/qm/test301` directory:

```
test301.out       test301.sum
```

In the `testo/orca/qm/test302` directory:

```
test302.out       test302.sum
```

…

In the `testo/orca/qm/test308` directory:

```
test308.out       test308.sum
```

In the `testo/orca/qmmm/test3001` directory:

```
test3001.sum      test3001.out
```

In the `testo/orca/qmmm/test3002` directory:

```
test3002.sum      test3002.out
```

…

In the `testo/orca/qmmm/test3035` directory:

```
Test3035.sum      test3035.out
```

In the `testo/gamess/qm/test401` directory:

```
test401.out       test401.sum
```

In the `testo/gamess/qm/test402` directory:

```
test402.out       test402.sum
```

…

In the `testo/gamess/qm/test406` directory:

```
test406.out       test406.sum      test406.extopt
```

In the `testo/gamess/qmmm/test4001` directory:

```
test4001.sum      test4001.out
```

In the `testo/gamess/qmmm/test4002` directory:

```
test4002.sum      test4002.out
```

…

In the `testo/gamess/qmmm/test4022` directory:

```
test4022.sum      test4022.out
```

In the `doc` directory:

```
QMMMmanualvx.y.z.doc                QMMMmanualvx.y.z.pdf
QMMMDevGuidex.y.z.doc               QMMMDevGuidex.y.z.pdf
In the tinker_QMMM/5.1 directory:
dampchg.i         echarge.f         echarge1.f        echarge3.f
initial.f         kcharge.f         ...
In the tinker_QMMM/6.3 directory:
```

```
dampchg.i        analyze.f        echarge.f        echarge1.f
echarge3.f       initial.f        kcharge.f        ...
```

Step 4: Set QMMM Path

Change the working directory to the QMMM home directory, and run the script `set_path.ml` by typing

set_path.ml <Return>

This script will create a file in the home directory named `.qmmm_path`*x.y.z* stating where the QMMM directory structure is located. This file is used by other scripts to locate QMMM on the user's system.

Step 5: Compile the Program

Change the working directory to the `script` directory within the QMMM home directory. Depending on the machine types, select appropriate scripts to compile the codes. For example, the script `comp_multi.ais` will have to be run to compile the program for IBM computer running AIX, where `ais` denotes the AIX operating system and options. This script may have to be modified to work with a compiler that is on the machine being used. (See also Section 9. Computers, Operating Systems, and Fortran Compilers for supported machine types.)

Run the script `comp_multi.ais` by typing

comp_multi.ais <Return>

This script will compile all of the source code for QMMM, placing the modules created into the `mod` directory, the object files created into the `obj` directory, and the executable created into the `exe` directory. Please check that 12 module files, 22 object files, and 1 executable have been created and placed in the appropriate directories; if they have not, then there was an error during the compilation.

Step 6: Configure Shuttle Scripts

Change the working directory to the `script` directory within the QMMM home directory. At this step, user needs to configure the shuttle scripts (see also Section 7.B.2. Program Shuttle Scripts).

A. *Gaussian*

If *Gaussian* is the electronic-structure program for the QMMM calculations, you need to follow the instruction below.

At this point, you should have determined which *Gaussian* version is going to be invoked. Currently, QMMM supports the following specific versions: *g03.b01*, *g03.c01*, *g03.d01*,*g03.e01*, *g09.a02*, *g09.e01*, *g16.a03*, *g16.b01*, and *g16.c01*; the shuttle scripts for these versions are placed in the `g03.b01`, `g03.c01`, `g03.d01`, `g09.a02`, `g09.e01`, `g16`, `g16`, and `g16` subdirectories, respectively. According to the *Gaussian* versions, copy the corresponding shuttle scripts from the `g03.b01`, `g03.c01`, `g03.d01`, `g03.e01`, `g09.a02`, `g09.e01` or `g16` subdirectories to the `script` directory. For example,

cp g03.e01/ . <Return>*

After doing so, the scripts `g03shuttle`, `ex_shuttle`, and `Gau_External` will be found in the `script` directory for the g03.b01 and g03.c01 cases, and scripts `g03shuttle`, `ex_shuttle`, `Gau_External`, and `Gau_External2` will be found in the `script` directory for the g03.d01, g03.e01, g09.a02, g09.e01, g16.a03, g16.b01, and g16.c01 cases.

Next, while the `script` directory is still the working directory, edit the `g03huttle` script, so that the path indicated for *Gaussian* in the script is accurate for file system used by the computer system on which QMMM has been installed. In addition, the user should specify the scratch directory to be used by *Gaussian* by properly setting the `scratchdir` environmental variable in the script. For example,

```
set gausspath=/usr/local/g03/g03.c01/g03
set scratchdir=/scratch2/linh
```

B. ORCA

If ORCA is the electronic-structure program for the QMMM calculations, you need to follow the instruction below.

Edit the `orcashuttle` and `orcapotshuttle` script, so that the path indicated for ORCA in the script is accurate for file system used by the computer system on which QMMM has been installed. For example,

```
set orcapath=~/orca/bin
```

C. GAMESS

If GAMESS is the electronic-structure program for the QMMM calculations, you need to follow the instruction below.

If you use your own script to run GAMESS (pure QM) jobs, likely the `rungms` script that was provided in the GAMESS distribution (perhaps with some modification to work with your file system), you should go to the "`Your Own Rungms`" section and follow the instruction there. First, you will need to modify your own `rungms` script by

- deleting the lines of setting `datpath`,
- adding a line at the end of the `rungms` script to move the `#.dat` file to the QMMM working directory after the GAMESS job is done.

Next, you should uncomment the next lines until (not include) "`End of Script`".

If you use an institute-defined script run GAMESS (pure QM) jobs, e.g., in the machines Altix, Regatta, and Netfinity at the Supercomputing Institute in the University of Minnesota, where such a script has been provided by computer administration, you should go to the "`Institute Defined Rungms`" section and follow the instruction there. First, you should uncomment the next two lines containing `gmspath` and `datpath` listed below:

```
#   set gmspath=~/bin
#   set datpath=~/scr
```

If the `gmspath` and `datapath` are different from those given above, please revise them to the correct location. Next, you should follow the instructions in the script to uncomment certain lines until (not include) "`End of Script`".

D. TINKER

If TINKER is the molecular-mechanism program for the QMMM calculations, you need to follow the instruction below.

Edit the `t41shuttle` script, so that the path indicated for TINKER in the script is accurate for file system used by the computer system on which QMMM has been installed. For example,

```
set tinkerpath=~/tinker4.2/bin
```

8.B. The QMMM Test Suite

The test suite has been designed to give the user a sample of the QMMM capabilities and input files and to provide examples of test input and output. It does not give examples of everything that can be done with the program, but each test run demonstrates a key feature of the program. These test runs also allow the user to familiarize himself or herself with the QMMM output format.

In order to use the test suite, change the working directory to the `testrun` directory in the QMMM home directory. Within this directory there are four subdirectories: `g03`, `orca`, `gamess`, and `tinker`.

Within the `g03` directory there are two subdirectories `qm` and `qmmm`. These two subdirectories contain the test runs for pure-QM and QM/MM types of calculations with *Gaussian*. Within each subdirectory, there are several sub-subdirectories, each of which is named `test#`, where `#` is an index for the test runs. The indexes from 201 to 299 are reserved for QM test runs, and from 2001 to 2999 for QM/MM test runs. Each of these sub-subdirectories contains a set of QMMM input files: the coordinate file `test#.crd`, the input file `test#.dat`, the parameter file `test#.prm`, and the script `test#.ml`.

Within the `orca` directory, there are two subdirectories: `qm` and `qmmm`. These two subdirectories contain the test runs for pure-QM and QM/MM types of calculations with ORCA. Within each subdirectory, there are several sub-subdirectories, each of which is named `test#`, where `#` is an index for the test runs. The indexes from 301 to 399 are reserved for QM test runs,

and from 3001 to 3999 for QM/MM test runs. Each of these sub-subdirectories contains a set of QMMM input files: the coordinate file `test#.crd`, the input file `test#.dat`, the parameter file `test#.prm`, and the script `test#.ml`.

Within the `gamess` directory there are two subdirectories: `qm` and `qmmm`. These two subdirectories contain the test runs for pure-QM and QM/MM types of calculations with GAMESS. Within each subdirectory, there are several sub-subdirectories, each of which is named `test#`, where `#` is an index for the test runs. The indexes from 401 to 499 are reserved for QM test runs, and from 4001 to 4999 for QM/MM test runs. Each of these sub-subdirectories contains a set of QMMM input files: the coordinate file `test#.crd`, the input file `test#.dat`, the parameter file `test#.prm`, and the script `test#.ml`.

Within the `tinker` directory there is one subdirectory: `mm`. This subdirectory contains the test runs for pure MM type of calculations with TINKER. Within this subdirectory, there are several sub-subdirectories, each of which is named `test#`, where `#` is an index for the test runs. The indexes from 101 to 199 are reserved for MM test runs, and each of these sub-subdirectories contains a set of QMMM input files: the coordinate file `test#.crd`, the input file `test#.dat`, the parameter file `test#.prm`, and the script `test#.ml`.

While it is recommended that the user run all test runs, users may select certain portions of the test suite to run. This may be done by changing to either the `testrun/tinker/mm/test#`, `testrun/g03/qm/test#`, `testrun/g03/qmmm/test#`, `testrun/gamess/qm/test#`, `testrun/gamess/qmmm/test#`, `testrun/orca/qm/test#`, or the `testrun/orca/qmmm/test#` directory and running the `test#.ml` script for the specific test run. In this case, the usage of the script for test2003, for example, would be

test2003.ml <Return>.

Users may also run all the QM tests in the batch mode with the `rgqmall` script in the `testrun/g03/qm` directory, the `rgmsqmall` script in the `testrun/gamess/qm` directory, and the `roqmall` script in the `testrun/orca/qm` directory, all the QMMM tests with the `rgqmmmall` script in the `testrun/g03/qmmm` directory, the `rgmsqmmmall` script in the

testrun/gamess/qmmm directory, and the roqmmmall script in the testrun/orca/qmmm directory, and all the MM tests with the rtmmall script in the testrun/tinker/mm directory. With the rgall script in the testrun/g03 directory, the rgmsall script in the testrun/gamess directory, and roall in the testrun/orca, he or she may run all the tests calculated with *Gaussian*, GAMESS, and ORCA, respectively.

Running the QM and QM/MM test runs in the batch mode may take 0.5 to 10 hours to finish, depending on which sets of test runs are selected. To help users submit the test run jobs to the PBS queuing system, we have provided the following .pbs scripts:

rgqmall.pbs in the testrun/g03/qm directory

rgmsqmall.pbs in the testrun/gamess/qm directory

roqmall.pbs in the testrun/orca/qm directory

rgqmmmall.pbs in the testrun/g03/qmmm directory

rgmsqmmmall.pbs in the testrun/gamess/qmmm directory

roqmmmall in the testrun/orca/qmmm directory

rgall in the testrun/g03 directory

rgmsall in the testrun/gamess directory

roall in the testrun/orca directory

Users may need to modify the scripts, according to the specific queuing environment of their own computers. The MM test runs will be done in seconds, and we thus do not provide such a script.

A portion of the output that these test runs should produce can be found in the directory testo. This directory contains both the QMMM output files test#.out and the test#.sum files. For geometry optimization invoking *Gaussian* optimizer through the *external* option, the external optimization output file test#.extopt, is also provided. For the most part, the output the user generates should be identical to that in the testo directory. Some minor numerical differences may arise due to round-off issues, but most output values should have similar values between your test runs and the supplied sample output. The output files distributed with the current version of the code were obtained on an IBM Regatta computer with p690 nodes for g03 runs and on a Linux machine with Athlon MP 2000+ CPU for orca and gamess runs.

When examining the output, the file of most interest to the user will probably be the output file from QMMM. This file is named `test#.out`.

Seven scripts (`checktestrun_g03_qm`, `checktestrun_g03_qmmm`, `checktestrun_gms_qm`, `checktestrun_gms_qmmm`, `checktestrun_orca_qm`, `checktestrun_orca_qmmm`, and `checktestrun_tinker`) are available in the `script/checktestrun` directory for comparisons of the test run results obtained by the user and those distributed in the directory `testo`. For example, the user can first run all the test runs in the `testrun/g03/qmmm` directory by running the `rgqmmmall` script. Next, the user can switch to the `script/checktestrun` directory and run the `checktestrun_g03_qmmm` script by typing:

checktestrun_g03_qmmm <Return>

By doing so, a new subdirectory *checktestrun_g03_qmmm* will be created. The results distributed with the QMMM program will be copied to this directory and renamed as `testrun20XY.out.old` and `test20XY.sum.old`, where `XY = 01, 02, … 68`. The results obtained by the user `testrun20XY.out` and `test20XY.sum` will be copied to this directory. The script will then invoke the UNIX command `diff` to make the comparisons.

8.B.1. Test 101(*TINKER*): MM Single-point Energy

Test run 101 performs a single-point energy calculation at the MM level for the Fe-porphyrin system containing 73 atoms, which is taken from the test suite in TINKER. The MM force field is CHARMM27.

8.B.2. Test 102(*TINKER*): MM Single-point Gradient

Test run 102 performs a single-point gradient calculation at the MM level for *n*-butane (C_4H_{10}). The MM force field is CHARMM27.

8.B.3. Test 103(*TINKER*): MM Single-point Hessian

Test run 103 performs a single-point Hessian calculation at the MM level for *n*-butane (C_4H_{10}). The MM force field is AMBER96.

8.B.4. Test 104(*TINKER*): MM Pre-optimization using the TINKER Optimizer

Test run 104 performs geometry pre-optimization for the alanine dipeptide containing 22 atoms, which is taken from the test suite in TINKER, at the MM level by calling the TINKER optimizer (newton). The MM force field is CHARMM22.

8.B.5. Test 105(*TINKER*): MM Dynamics

Test run 105 performs molecular dynamics simulation on CF_3CH_2OH + 9 H_2O in a NVE ensemble. Gradients are calculated with TINKER, at the MM level. The MM force field is OPLS_AA.

8.B.6. Test 106(*TINKER*): MM Adaptive Partitioning

Test run 106 performs molecular dynamics simulation with adaptive partitioning on 171 Argon atoms in a NVE ensemble. Gradients are calculated with Lenard Jones potetial (MM) and Morse potential (MM).

8.B.7. Test 107(*TINKER*): MM Dynamics along a Path

Test run 107 performs molecular dynamics simulation on CF_3CH_2OH + 9 H_2O in a NVE ensemble. Gradients are calculated with TINKER, at the MM level. The MM force field is OPLS_AA. One water molecule is moved along the path set in the file test107.path.

8.B.8. Test 201(*Gaussian*): QM Single-Point Energy Calculation

Test run 201 performs a QM single-point energy calculation for H_2 at the MP2/cc-pVDZ level.

8.B.9. Test 202(*Gaussian*): QM Gradient Calculation for CF_3CH_2OH

Test run 202 performs a QM gradient calculation for CF_3CH_2OH at the HF/MIDI! level.

8.B.10. Test 203(*Gaussian*): QM Hessian Calculation for $CF_3CH_2O^-$

Test run 203 performs a QM Hessian calculation for $CF_3CH_2O^-$ at the HF/MIDI! Level.

8.B.11. Test 204(*Gaussian*): QM Optimization for H_2O

Test run 204 performs a QM geometry optimization for H_2O at the HF/MIDI! level.

8.B.12. Test 205(*Gaussian*): QM Optimization for CF_3CH_2OH

Test run 205 performs a QM geometry optimization for CF_3CH_2OH at the HF/MIDI! level.

8.B.13. Test 206(*Gaussian*): QM Optimization for $CF_3CH_2O^-$

Test run 206 performs a QM geometry optimization for $CF_3CH_2O^-$ at the HF/MIDI! level.

8.B.14. Test 207(*Gaussian*): QM Optimization for Ace-His-NMe

Test run 207 performs a QM geometry optimization for Ace-His-NMe at the HF/MIDI! level.

8.B.15. Test 208(*Gaussian*): QM Optimization for the Eigen Cation $H_9O_3^+$

Test run 208 performs a QM geometry optimization for the Eigen cation $H_9O_3^+$ at the B3LYP/6-31++G** level.

8.B.16. Test 209(*Gaussian*): QM Optimization for $HS^-\ldots H_2O$

Test run 209 performs a QM geometry optimization for $HS^-\ldots H_2O$ at the B3LYP/6-31++G** level.

8.B.17. Test 210(*Gaussian*): QM Dynamics

Test run 210 performs a QM dynamics simulation of CF_3CH_2OH in a NVE ensemble at the hf/midix level using Gaussian.

8.B.18. Test 2001(*Gaussian*): QM/MM Single-point Energy

Test run 2001 performs a single-point energy calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.19. Test 2002(*Gaussian*): QM/MM Single-point Gradient

Test run 2002 performs a single-point gradient calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme. The geometry in this test is the same as that in test 2001.

8.B.20. Test 2003(*Gaussian*): QM/MM Single-point Hessian

Test run 2003 performs a single-point Hessian calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme. The geometry in this test is the same as that in test 2001.

8.B.21. Test 2004(*Gaussian*): QM/MM Optimization and Vibrational Analysis (1)

Test run 2004 performs a geometry optimization for CF_3-CH_2OH at the QM/MM level by calling the Gaussian optimizer through the GAUEXT keyword. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme. This is the one of the calculations presented in Ref. 82.

8.B.22. Test 2005(*Gaussian*): QM/MM Optimization and Vibrational Analysis (2) – ESP Charges through Atom Types

Test run 2005 performs geometry optimizations for CF_3-CH_2OH at the QM/MM level by calling the Gaussian optimizer through the gauext keyword. The PS is CH_2OH, and the SS is CF_3. The

QM level is HF/MIDI!, and the MM force field is OPLS-AA except for the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured.(*82*) The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. The QM/MM boundary is treated by using the RCD (default) scheme. This is the one of the calculations presented in Ref.(*82*)

8.B.23. Test 2006(*Gaussian*): QM/MM Single-point Energy on QM/MM Geometry (1) – ESP Charges through Atom Types

Test run 2006 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 2005. The setups for computations are the same as those in test 2005; in particular, we note that the ESP charge specification is done by changing the MM charges for the corresponding atom types.

8.B.24. Test 2007(*Gaussian*): QM/MM Single-point Energy on QM/MM Geometry (2) – ESP charges Through Atom Indices

Test run 2007 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 2005. This test is almost the same as test 2006, except that the ESP charge specification is done by changing the MM charges for the corresponding *atoms*. Users are advised to study the how the qmmm program handle atom index in a QM/MM calculation, and are also encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given atoms.

8.B.25. Test 2008(*Gaussian*): QM/MM Single-point Energy on QM Geometry (1) – RCD Scheme

Test run 2008 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.26. Test 2009(*Gaussian*): QM/MM Single-point Energy on QM Geometry (2) – RCD Scheme /CM2 Charges

Test run 2009 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, expect for the CM2 charges that are used for the CF_3 group. The CM2 charges for CF_3 are derived from CM2 charge model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The CM2 charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.27. Test 2010(*Gaussian*): QM/MM Single-point Energy on QM Geometry (3) – RCD Scheme /CM3 Charges

Test run 2010 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, expect for the CM3 charges that are used for the CF_3 group. The CM3 charges for CF_3 are derived from CM3 charge model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The CM3 charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.28. Test 2011(*Gaussian*): QM/MM Single-point Energy on QM Geometry (4) – RCD Scheme /ESP Charges

Test run 2011 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, expect for the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to

study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.29. Test 2012(*Gaussian*): QM/MM Single-point Energy on QM Geometry (5) – RCD Scheme /Löwdin Charges

Test run 2012 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, expect for the Löwdin charges that are used for the CF_3 group. The Löwdin charges for CF_3 are derived from Löwdin charge-model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The Löwdin charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.30. Test 2013(*Gaussian*): QM/MM Single-point Energy on QM Geometry (6) – RCD Scheme /Mulliken Charges

Test run 2013 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except for the Mulliken charges that are used for the CF_3 group. The Mulliken charges for CF_3 are derived from Mulliken charge-model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The Mulliken charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.31. Test 2014(*Gaussian*): QM/MM Single-point Energy on QM Geometry (7) – RC Scheme

Test run 2014 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except that the RC scheme is adopted.

8.B.32. Test 2015(*Gaussian*): QM/MM Single-point Energy on QM Geometry (8) – Shift Scheme

Test run 2015 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except that the Shift scheme is adopted.

8.B.33. Test 2016(*Gaussian*): QM/MM Single-point Energy on QM Geometry (9) – Z1 Scheme

Test run 2016 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except that the Z1 scheme is adopted.

8.B.34. Test 2017(*Gaussian*): QM/MM Single-point Energy on QM Geometry (10) – Z2 Scheme

Test run 2017 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except that the Z2 scheme is adopted.

8.B.35. Test 2018(*Gaussian*): QM/MM Single-point Energy on QM Geometry (11) – Z3 Scheme

Test run 2018 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except that the Z3 scheme is adopted.

8.B.35. Test 2019(*Gaussian*): QM/MM Single-point Energy on QM Geometry (12) – SEE Scheme

Test run 2019 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except that the SEE scheme is adopted.

8.B.37. Test 2020(*Gaussian*): QM/MM Single-point Energy on QM Geometry (13) – RCD2 Scheme

Test run 2020 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except that the RCD2 scheme is adopted.

8.B.38. Test 2021(*Gaussian*): QM/MM Single-point Energy on QM Geometry (14) – ME Scheme

Test run 2021 performs a QM/MM single-point energy calculation for CF_3-CH_2OH on the geometry optimized in test 205. The setups are almost the same as those in test 2008, except that the ME scheme is adopted.

8.B.39. Test 2022(*Gaussian*): QM/MM Single-point Energy on QM Geometry with Charged PS (1)

Test run 2022 performs a QM/MM single-point energy calculation for CF_3-CH_2O^- on the geometry optimized in test 206. The PS is CH_2O^-, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.40. Test 2023(*Gaussian*): QM/MM Single-point Energy on QM Geometry with Charged PS (2) – ESP Charges for SS

Test run 2023 performs a QM/MM single-point energy calculation for CF_3-CH_2O^- on the geometry optimized in test 206. The setups are the same as those in test 2022 except for the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.41. Test 2024(*Gaussian*): QM/MM Optimization and Vibrational Analysis with Charged PS (1)

Test run 2024 performs a QM/MM optimization for CF_3-CH_2O^-, followed by a vibrational analysis. The PS is CH_2O^-, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme. This is the one of the calculations presented in Ref.(*82*)

8.B.42. Test 2025(*Gaussian*): QM/MM Optimization and Vibrational Analysis with Charged PS (2) – ESP Charges for SS

Test run 2025 performs a QM/MM optimization for CF_3-CH_2O^-, followed by a vibrational analysis. The setups are the same as those in test 2024 except for the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. This is the one of the calculations presented in Ref. (*82*)

8.B.43. Test 2026(*Gaussian*): QM/MM Rotational Barrier

Test run 2026 calculates the rotational barrier for the C-C-C-C dihedral in *n*-butane (C_4H_{10}) at the QM/MM level. The QM/MM boundary, which goes through the central C-C bond, is treated by use of the RCD scheme (default). The QM level is MPW1K/6–31+G(d,p), and the MM force field is OPLS-AA. The partial optimization is carried out by calling the Gaussian optimizer through the GAUEXT keyword. The *Gaussian* output file `test2026.extopt` contains the results for the torsional energy profile.

8.B.44. Test 2027(*Gaussian*): QM/MM Optimization – QM/MM Boundary does not Pass through a Covalent Bond

Test run 2027 performs a QM/MM optimization for the CH_2OH-CH_2OH…H_2O complex. The PS is the one of the CH_2OH group, which has a QM/MM boundary at one side going through the covalent C–C bond and a QM/MM boundary at the other side that does not pass through a covalent bond. The SS includes the other CH_2OH group and the H_2O. The QM level is

HF/MIDI!, and the MM force field is OPLS-AA. The RCD scheme is used to treat the QM/MM boundary where it passes through the covalent bond.

8.B.45. Test 2028(*Gaussian*): QM/MM for Hydrogen Transfer Reaction (1) – Reactant

Test runs 2028 to 2032 study the H atom transfer reaction between the CH_3 and CH_3CH_2-CH_2OH at the QM/MM level: $CH_3 + CH_3CH_2CH_2OH \rightarrow CH_4 + CH_2CH_2CH_2OH$, which is reported in Ref.(*82*) The primary system is $CH_3 + CH_3CH_2$, giving rise to $CH_3 + CH_3CH_3$ as the capped primary system, and the secondary system is CH_2OH. The QM level is MPW1K/6–31+G(d,p), and the MM force field is OPLS-AA except for the ESP charges that are used for the SS; the ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. Test 2028 performs a QM/MM optimization for the reactant CH_3CH_2-CH_2OH, followed by a QM/MM vibrational analysis at the optimized geometry.

8.B.46. Test 2029(*Gaussian*): QM/MM for Hydrogen Transfer Reaction (2) – Product

Test runs 2028 to 2032 study the H atom transfer reaction between the CH_3 and CH_3CH_2-CH_2OH at the QM/MM level: $CH_3 + CH_3CH_2CH_2OH \rightarrow CH_4 + CH_2CH_2CH_2OH$, which is reported in Ref.(*82*) See the descriptions in test 2028 for computational details. Test 2029 performs a QM/MM optimization for the product CH_2CH_2-CH_2OH, followed by a QM/MM vibrational analysis at the optimized geometry.

8.B.47. Test 2030(*Gaussian*): QM/MM for Hydrogen Transfer Reaction (3) – Initial Hessian for Saddle-Point Optimization

Test runs 2028 to 2032 study the H atom transfer reaction between the CH_3 and CH_3CH_2-CH_2OH at the QM/MM level: $CH_3 + CH_3CH_2CH_2OH \rightarrow CH_4 + CH_2CH_2CH_2OH$, which is reported in Ref.(*82*) See the descriptions in test 2028 for computational details. Test 2030 calculates analytically the initial QM/MM Hessian for the starting geometry that will be used for the saddle-point optimization.

8.B.48. Test 2031(*Gaussian*): QM/MM for Hydrogen Transfer Reaction (4) – Saddle-Point Optimization

Test runs 2028 to 2032 study the H atom transfer reaction between the CH_3 and CH_3CH_2-CH_2OH at the QM/MM level: $CH_3 + CH_3CH_2CH_2OH \rightarrow CH_4 + CH_2CH_2CH_2OH$, which is reported in Ref.(*82*) See the descriptions in test 2028 for computational details. Test 2031 reads in the initial QM/MM Hessian calculated analytically in test 2030 and optimizes the saddle point geometry.

8.B.49. Test 2032(*Gaussian*): QM/MM for Hydrogen Transfer Reaction (5) – Higher QM Level for Saddle Point

Test runs 2028 to 2032 study the H atom transfer reaction between the CH_3 and CH_3CH_2-CH_2OH at the QM/MM level: $CH_3 + CH_3CH_2CH_2OH \rightarrow CH_4 + CH_2CH_2CH_2OH$, which is reported in Ref.(*82*) See the descriptions in test 2028 for computational details. Test 2032 performs a single-point energy calculation at the CCSD/6–311G(d,p):OPLS-AA level at the saddle-point geometry optimized in test 2031, i.e., optimized at the MPW1K/6–31+G(d,p):OPLS-AA level. Here, the QM and MM levels are given as QM:MM; the notation is in the same format as in the ONIOM (http://www.gaussian.com/g_ur/k_oniom.htm) calculations in *Gaussian*. The motivation in doing so is to improve the energetics for the reaction, and the calculation can be denoted CCSD/6–311G(d,p):OPLS-AA// MPW1K/6–31+G(d,p)/:OPLS-AA.

8.B.50. Test 2033(*Gaussian*): QM/MM for Hydrogen Transfer Reaction 2 (1) – Saddle Point Optimization with a Smaller QM Subsystem

Test runs 2033 to 2035 study the H atom transfer reaction between the H and $CH_3CH_2CH_2CH_3$ at the QM/MM level: $H + CH_3CH_2CH_2CH_3 \rightarrow H_2 + CH_2CH_2CH_2CH_3$. Each of these three test runs performs a QM/MM optimization for the saddle point, followed by a QM/MM vibrational analysis at the optimized geometry. The QM level is MPWB1K/6–31+G(d,p), and the MM force field is MM3. The mechanical embedding scheme is used. In test 2033, the primary system is a "small choice", $H + CH_3$, giving rise to $H + CH_4$ as the capped primary system, and the secondary system is $CH_2CH_2CH_3$. Notice that MM3 parameter file in TINKER 4.2 is different

from that in TINKER 6.3. The current parameter file works with TINKER 6.3, and the parameter file that works with TINKER 4.2 is named as test2033.prm.t41.

8.B.51. Test 2034(*Gaussian*): QM/MM for Hydrogen Transfer Reaction 2 (2) – Saddle Point Optimization with a Larger QM Subsystem

Test runs 2033 to 2035 study the H atom transfer reaction between the H and $CH_3CH_2CH_2CH_3$ at the QM/MM level: $H + CH_3CH_2CH_2CH_3 \rightarrow H_2 + CH_2CH_2CH_2CH_3$. See the descriptions in test 2033 for computational details. In test 2034, the primary system is a "bigger choice", $H + CH_3CH_2$, giving rise to $H + CH_3CH_3$ as the capped primary system, and the secondary system is CH_2CH_3. Notice that MM3 parameter file in TINKER 4.2 is different from that in TINKER 6.3. The current parameter file works with TINKER 6.3, and the parameter file that works with TINKER 4.2 is named as test2034.prm.t41.

8.B.52. Test 2035(*Gaussian*): QM/MM for Hydrogen Transfer Reaction 2 (3) – Saddle Point Optimization with a Larger QM Subsystem and a Radical Atom-type for C

Test runs 2033 to 2035 study the H atom transfer reaction between the H and $CH_3CH_2CH_2CH_3$ at the QM/MM level: $H + CH_3CH_2CH_2CH_3 \rightarrow H_2 + CH_2CH_2CH_2CH_3$. See the descriptions in test 2033 for computational details. In test 2035, the primary system is a "bigger choice", $H + CH_3CH_2$, giving rise to $H + CH_3CH_3$ as the capped primary system, and the secondary system is CH_2CH_3. The setup in test 2035 is the same as that in test 2034 except that the C atom involving in the bond breaking and bond forming is assigned an atom type of radical.

8.B.53. Test 2036(*Gaussian*): QM/MM Single-point Gradient for Ace-Lys-NMe with QM/MM Cutoff

Test run 2036 performs a single-point-gradient calculation using QM/MM cutoff for the Ace-Lys-NMe model system at the QM/MM level. The PS is the side chain, and the SS is the backbone. The center of the cutoff sphere is the 8^{th} atom (an C atom), and the cutoff radius is 3 Å. This example is solely to demonstrate how to use the QM/MM cutoff; it is by no mean to suggest the use of 3 Å as the cutoff radius. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD scheme.

8.B.54. Test 2037(*Gaussian*): QM/MM Partial Optimization for Ace-Lys-NMe using the *Gaussian* Optimizer

Test run 2037 performs a partial geometry optimization for the Ace-Lys-NMe model system at the QM/MM level. The PS is the side chain, and the SS is the backbone. The indexes of the active atoms (the atoms whose coordinates are optimized) are input. Two layers of frozen atoms are passed to the *Gaussian* optimizer. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD scheme.

8.B.55. Test 2038(*Gaussian*): QM/MM Partial Optimization for Ace-Lys-NMe using the *Gaussian* Optimizer

Test run 2038 performs a partial geometry optimization for the Ace-Lys-NMe model system at the QM/MM level. The PS is the side chain, and the SS is the backbone. The indexes of the active atoms (the atoms whose coordinates are optimized) are determined by inputting the center coordinates and radius. Two layers of frozen atoms are passed to the *Gaussian* optimizer. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD scheme.

8.B.56. Test 2039(*Gaussian*): QM/MM Partial Optimization for Ace-Lys-NMe using the *Gaussian* Optimizer and using the QM/MM Cut-off

Test run 2039 performs a partial geometry optimization using the QM/MM cutoff for the Ace-Lys-NMe model system at the QM/MM level. The PS is the side chain, and the SS is the backbone. The indexes of the active atoms (the atoms whose coordinates are optimized) are: 13, 14 …, 28. The center of the cutoff sphere is the 8^{th} atom (an C atom), and the cutoff radius is 3 Å. This example is solely to demonstrate how to use the QM/MM cutoff; it is by no mean to suggest the use of 3 Å as the cutoff radius. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD scheme.

8.B.57. Test 2040(*Gaussian*): QM/MM Optimization for the CH_2OH-CH_2OH…H_2O using the PBRC Scheme – Two Polarizable MM Groups

Test run 2040 performs a QM/MM optimization using the polarized-embedding scheme that allows the self-consistent mutual polarizations between the QM and MM subsystems near the boundary for the CH_2OH-CH_2OH…H_2O complex at the QM/MM level. The PS is the one of the CH_2OH group, for which the QM/MM boundary at one side going through the covalent C–C bond and at the other side separating from the H_2O molecule without passing through a covalent bond. The SS includes the other CH_2OH group and the H_2O. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The RC scheme is used to treat the QM/MM boundary where it passes through the covalent bond. The charge equalization is done by using the QEq-BT method with two polarizable groups H_2OH and H_2O.

8.B.58. Test 2041(*Gaussian*): QM/MM Optimization for the CH_2OH-CH_2OH…H_2O using the PBRC Scheme with Two Polarizable MM Groups and the User-Input Parameters

Test run 2041 performs a QM/MM optimization using the polarized-embedding scheme that allows the self-consistent mutual polarizations between the QM and MM subsystems near the boundary for the CH_2OH-CH_2OH…H_2O complex at the QM/MM level. The PS is the one of the CH_2OH group, for which the QM/MM boundary at one side going through the covalent C–C bond and at the other side separating from the H_2O molecule without passing through a covalent bond. The SS includes the other CH_2OH group and the H_2O. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The RC scheme is used to treat the QM/MM boundary where it passes through the covalent bond. The charge equalization is done by using the QEq-BT method with two polarizable groups H_2OH and H_2O and with the parameters input by the user.

8.B.59. Test 2042(*Gaussian*): QM/MM Geometry Optimization for Ace-His-NMe using the PBRCD Scheme with the QEq-SCT Model for Charge Equalization

Test run 2042 performs the geometry optimization for the Ace-His-NMe model system at the QM/MM level using the polarized-embedding scheme that allows the self-consistent mutual polarizations between the QM and MM subsystems near the boundary. The initial geometry is optimized in test207. The PS is the side chain, and the SS is the backbone. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD scheme. The charge equalization is done by using the QEq-SCT method.

8.B.60. Test 2043(*Gaussian*): QM/MM Geometry Optimization for Ace-His-NMe using the PBRCD Scheme with the QEq-BT Model for Charge Equalization

Test run 2043 performs the geometry optimization for the Ace-His-NMe model system at the QM/MM level using the polarized-embedding scheme that allows the self-consistent mutual polarizations between the QM and MM subsystems near the boundary. The initial geometry is optimized in test207. The PS is the side chain, and the SS is the backbone. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD scheme. The charge equalization is done by using the QEq-BT method.

8.B.61. Test 2044(*Gaussian*): QM/MM Geometry Optimization for Ace-His-NMe using the PBRCD Scheme with the QEq-EEM Model for Charge Equalization

Test run 2044 performs the geometry optimization for the Ace-His-NMe model system at the QM/MM level using the polarized-embedding scheme that allows the self-consistent mutual polarizations between the QM and MM subsystems near the boundary. The initial geometry is optimized in test207. The PS is the side chain, and the SS is the backbone. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD scheme. The charge equalization is done by using the QEq-EEM method.

8.B.62. Test 2045(*Gaussian*): QM/MM Geometry Optimization for Ace-His-NMe using the PBRC Scheme with the QEq-SCT Model for Charge Equalization

Test run 2045 performs the geometry optimization for the Ace-His-NMe model system at the QM/MM level using the polarized-embedding scheme that allows the self-consistent mutual polarizations between the QM and MM subsystems near the boundary. The initial geometry is optimized in test207. The PS is the side chain, and the SS is the backbone. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RC scheme. The charge equalization is done by using the QEq-SCT method.

8.B.63. Test 2046(*Gaussian*): QM/MM Geometry Optimization for Ace-His-NMe using the PBRC Scheme with the QEq-BT Model for Charge Equalization

Test run 2046 performs the geometry optimization for the Ace-His-NMe model system at the QM/MM level using the polarized-embedding scheme that allows the self-consistent mutual

polarizations between the QM and MM subsystems near the boundary. The initial geometry is optimized in test207. The PS is the side chain, and the SS is the backbone. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RC scheme. The charge equalization is done by using the QEq-BT method.

8.B.64. Test 2047(*Gaussian*): QM/MM Geometry Optimization for Ace-His-NMe using the PBRC Scheme with the QEq-EEM Model for Charge Equalization

Test run 2047 performs the geometry optimization for the Ace-His-NMe model system at the QM/MM level using the polarized-embedding scheme that allows the self-consistent mutual polarizations between the QM and MM subsystems near the boundary. The initial geometry is optimized in test207. The PS is the side chain, and the SS is the backbone. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RC scheme. The charge equalization is done by using the QEq-EEM method.

8.B.65. Test 2048(*Gaussian*): QM/MM Flexible-boundary Calculations for the Eigen Cation $H_9O_3^+$ using the QEQ-SCT Model for the Polarization Treatment

Test run 2048 performs the flexible-boundary calculations for the Eigen cation $H_9O_3^+$ using the QEQ-SCT Model for the polarization treatment of the SS atoms. The geometry has been optimized in test208. The central H_3O^+ is the PS, and the three hydrogen-bonding H_2O molecules are the SS. The two oxidation-states of the PS are H_3O^+ and H_3O. The electronic temperature is 30,000 K. The QM level is B3LYP/6-31++G**, and the MM force field is OPLS-AA.

8.B.66. Test 2049(*Gaussian*): QM/MM Flexible-boundary Calculations for $HS^-\cdots H_2O$ using the QEQ-SCT Model for the Polarization Treatment

Test run 2049 performs the flexible-boundary calculations for $HS^-\cdots H_2O$ using the QEQ-SCT Model for Charge Calculation. The geometry has been optimized in test209. The PS is HS^-, and the SS is H_2O. The two oxidation-states of the PS are HS^- and HS. The electronic temperature is 30,000 K. The QM level is B3LYP/6-31++G**, and the MM force field is OPLS-AA.

8.B.67. Test 2050(*Gaussian*): QM/MM Single-point Energy using Previously Obtained .chk File

Test run 2050 performs a single-point energy calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme. The initial guess of orbitals is read from a previously obtained *Gaussian* checkpoint file. Be aware that the unformatted checkpoint file is machine-dependent. We have provided a formatted checkpoint file `guess.FChk`, which can be converted to unformatted checkpoint `guess.chk` file by using the *Gaussian* utility program *formchk*.

8.B.68. Test 2051(*Gaussian*): QM/MM Optimization Using LBFGS Algorithm and Previously Obtained .chk File

Test run 2051 performs a geometry optimization for CF_3-CH_2OH at the QM/MM level by using the LBFGS algorithm. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme. The initial guess of orbitals in the gradient calculations at the every optimized step is read from the checkpoint file obtained in the previous step. Be aware that the unformatted checkpoint file is machine-dependent. We have provided a formatted checkpoint file `guess.FChk`, which can be converted to unformatted checkpoint `guess.chk` file by using the *Gaussian* utility program *formchk*.

8.B.69. Test 2052(*Gaussian*): QM/MM Optimization and Vibrational Analysis using Previously Obtained .chk File

Test run 2052 performs a geometry optimization for CF_3-CH_2OH at the QM/MM level by calling the Gaussian optimizer through the gauext keyword. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme. The initial guess of the orbitals in the gradient calculations at the every optimized step and in the final Hessian calculations are read from the checkpoint file obtained in the previous step. Be aware that the unformatted checkpoint file is machine-dependent. We have provided a formatted checkpoint file `guess.FChk`, which can be converted to unformatted checkpoint `guess.chk` file by using the *Gaussian* utility program *formchk*.

8.B.70. Test 2053(*Gaussian*): QM/MM Gradient Calculations for CF_3CH_2OH Soaked in a Tiny Water Box: Boundary Passing through a Covalent Bond

Test run 2053 performs a gradient calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3 and 26 H_2O molecules. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.71. Test 2054(*Gaussian*): QM/MM Gradient Calculations for CF_3CH_2OH Soaked in a Tiny Water Box: Boundary Does Not Passing through Covalent Bonds

Test run 2054 performs a gradient calculation for CF_3-CH_2OH at the QM/MM level. The PS is CF_3CH_2OH, and the SS is 26 H_2O molecules. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. Electronic embedding is used.

8.B.72. Test 2055(*Gaussian*): QM/MM Geometry Optimization for CH_3CH_2COOH using the FBRC Treatment with the QEq-SCT Model

Test run 2055 performs the geometry optimization for the CH_3CH_2COOH model system at the QM/MM level using the FBRC treatment that allows the charge transfer between the QM and MM subsystems across the boundary. The PS is CH_2COOH, and the SS is CH_3. The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The charge equalization is done by using the QEq-SCT method.

8.B.73. Test 2056(*Gaussian*): QM/MM Geometry Optimization for $CH_3CH_2COO^-$ using the FBRC Treatment with the QEQ-SCT Model

Test run 2056 performs the geometry optimization for the $CH_3CH_2COO^-$ model system at the QM/MM level using the FBRC treatment that allows the charge transfer between the QM and MM subsystems across the boundary. The PS is CH_2COO^-, and the SS is CH_3.The QM level is HF/MIDI!, and the MM force field is OPLS-AA. The charge equalization is done by using the QEq-SCT method.

8.B.74. Test 2057(*Gaussian*): QM/MM Dynamics

Test run 2057 performs QM/MM dynamics for CF_3CH_2OH + 9 H_2O. The QM calculation are at hf/ midix level with Gaussian and the MM calculation with TINKER with OPLS-AA force fields.

8.B.75. Test 2058(*Gaussian*): Adaptive Partitioning QM/MM

Test run 2058 performs adaptive partitioning QM/MM dynamics for a water box with one Cl- ion The QM calculation are at AM1 level with Gaussian and the MM calculation with TINKER and CHARMM27 force fields.

8.B.76. Test 2059(*Gaussian*): Adaptive Partitioning QM/MM

Test run 2059 performs adaptive QM/MM dynamics of one Butanol + H_2O. The QM calculation are at AM1 level with Gaussian and the MM calculation with TINKER with a modified OPLS-AA force fields.

8.B.77. Test 2060(*Gaussian*): Adaptive Partitioning QM/MM

Test run 2060 performs adaptive QM/MM dynamics of one Butanol + two H_2O. The QM calculation are at B3LYP/6-31G* level with Gaussian and the MM calculation with TINKER with a modified OPLS-AA force fields.

8.B.78. Test 2061 (*Gaussian*): QM/MM Optimization for $HOCH_2CH_2OOCH_3$ using the BRC scheme

Test run 2061 performs QM/MM optimization for $HOCH_2CH_2OOCH_3$ using the balanced RC (BRC) scheme. The QM calculation is at the M06-2X/6-31G* level with *Gaussian,* and the MM calculation is with MMFF94 force field. M06-2X/6-31G* CM4 charges are used for MM atoms. Note that the MMFF94 force field is used for test runs 2061-2064. Since only the TINKER 6 distributed version of TINKER has implemented MMFF94, one needs to use modified TINKER 6.3, which is included in the distribution, for test runs 2061-2064.

8.B.79. Test 2062 (*Gaussian*): QM/MM Optimization for $HOCH_2CH_2OOCH_3$ using the BRC2 scheme

Test run 2062 performs QM/MM optimization for $HOCH_2CH_2OOCH_3$ using the balanced RC2 (BRC2) scheme. The QM calculation is by M06-2X/6-31G* with *Gaussian*, and the MM

calculation is with the MMFF94 force field and M06-2X/6-31G* CM4M charges. Note that the MMFF94 force field is used for test runs 2061-2064. Since only the TINKER 6 distributed version of TINKER has implemented MMFF94, one needs to use modified TINKER 6.3 for test runs 2061-2064.

8.B.80. Test 2063 (*Gaussian*): QM/MM Optimization for $HOCH_2CH_2OOCH_3$ using TBRC scheme with charge smearing

Test run 2063 performs QM/MM optimization for $HOCH_2CH_2OOCH_3$ using the tuned and smeared balanced RC (TBSRC) scheme. The QM calculation is at the M06-2X/6-31G* level with *Gaussian*, and the MM calculation is with the MMFF94 force field. The parameter used for the tuned pseudoatom is derived by reproducing the sum of Mulliken 6-31G* charges in the QM portion of the system, and it is provided in the input file. The smearing width of the redistributed charge is 1 Å. M06-2X/6-31G* CM4M charges are used as MM charges. When using the charge smearing scheme, one needs to add a dummy atom type in the parameter file. The dummy atom will be used for the smeared charges in the calculation. Note that the MMFF94 force field is used for test runs 2061-2064. Since only the TINKER 6 distributed version of TINKER has implemented MMFF94, one needs to use modified TINKER 6.3 for test runs 2061-2064.

8.B.81. Test 2064 (*Gaussian*): QM/MM Optimization for $HOCH_2CH_2OOCH_3$ using the TBRC2 scheme

Test run 2064 performs QM/MM optimization for $HOCH_2CH_2OOCH_3$ using the tuned and balanced RC2 (TBRC2) scheme. The QM calculation is at the M06-2X/6-31G* level with *Gaussian*, and the the MM calculation is with the MMFF94 force field. The parameter used for the tuned pseudoatom is derived by reproducing the sum of Mulliken 6-31G* charges in the QM portion of the system, and it is provided in the input file. M06-2X/6-31G* CM4M charges are used as MM charges. Since only the TINKER 6 distributed version of TINKER has implemented MMFF94, one needs to use modified TINKER 6.3 for test runs 2061-2064.

8.B.82. Test 2065 (*Gaussian*): QM-MM electrostatic interaction between two H_2O molecules using screened charges

Test run 2065 performs a single-point QM/MM calculation on an H_2O dimer. One H_2O is in the QM region, and the other is in the MM region. The QM calculation is by HF/aug-cc-pVTZ with *Gaussian*, and the MM calculation is with TINKER using the OPLSAA force field. HF/aug-cc-pVTZ CHELPG charges are used as MM charges. The H_2O in the MM region is represented by screened charges. Since the vdWs parameters are not reparametrized with the screened charges, only the QM-MM electrostatic energy term is reliable, and the total QM–MM interaction energy should be used with caution, but nevertheless we provide this example to show how to use the code. The screened charge scheme can be run with either modified TINKER 5.1 or modified TINKER 6.3. The test run uses modified TINKER 6.3. We also found that *Gaussian03* sometimes cause numerical inaccuracy for screened charges, and *Gaussian09* is suggested for use with screened charge calculations.

8.B.83. Test 2066 (*Gaussian*): QM/MM partial optimization for NU-1000(MIX-S) using the NU1T force field

Test run 2066 performs a QM/MM partial optimization for the metal-organic framework NU-1000 with the MIX-S proton topology using the BRC2 scheme and H link atoms, which are the options that have been determined to be the best combination for this system (*164*). Multiple charge balancing groups are used in charge balancing and modification. The M06-L functional with the 6-31G(d,p) basis set for C, H, O and the SDD basis set for Zr is used for the QM calculation with *Gaussian 09*. The NU1T force field (*164*), which is developed for the NU-1000 based on the Bristow-Tiana-Walsh transferable force field, is used for the MM calculation. This is an example of performing QM/MM calculations on NU-1000 with recommended setups. This example also shows that the *QMMM 2017* is compatible with *Gaussian 09* in doing partial optimization.

8.B.84. Test 2067 (*Gaussian*): QM/MM single point for NU-1000(MIX-S) using NU1T, amber-2, and F*

Test run 2067 performs a QM/MM single point calculation for the metal-organic framework NU-1000 with its MIX-S proton topology using the Amber-2 scheme and tuned F link atoms. Multiple charge balancing groups are used in charge balancing and modification. The M06-L functional with the 6-311+G(df,p) basis set for C, H, O and the SDD basis set for Zr is used for

the QM calculation with *Gaussian 09*. The NU1T force field is used for the MM calculation. The parameter used for the tuned F is derived by reproducing the sum of CM5 charges in the QM portion of the system, and it is provided in the input file. This example is to show the capability of *QMMM 2017* to deal with cases including multiple charge balancing groups when using the Amber-2 scheme (this is a new capability in *QMMM 2017* – see the revision history).

8.B.85. Test 2068 (*Gaussian*): QM/MM optimization for $HOCH_2CH_2OOCH_3$ using the *Gaussian 16* QM program

Test run 2068 performs QM/MM optimization for $HOCH_2CH_2OOCH_3$ using the balanced RC (BRC) scheme. The QM calculation is at the MN15/6-31G* level with *Gaussian 16,* and the MM calculation is with the MMFF94 force field.

8.B.86. Test 2069 (*Gaussian*): QM/MM optimization on a large MOF system

Test run 2069 performs QM/MM optimization on a large MOF system (in particular, an Ir-complex installed on UiO-67) using the balanced redistributed charge-2 (BRC2) scheme. The QM subsystem contains 24 atoms, and the MM subsystem contains 592 atoms. In this QM/MM system, two M1 atoms (i.e., MM boundary atoms) are present in the MM subsystem which is considered as one charge balancing group when a balanced charge modification scheme is used. The M06-L functional with the def2-SVP basis set for C, H and the def2-TZVP basis set for Ir, N is used for the QM calculation with *Gaussian 16,* and the MM calculation is done with the modified BTW-FF force field. We used the newly supported *Gaussian* include file mechanism to specify the basis sets (see the revision history of *QMMM 2018*). This example also shows the capability of *QMMM 2018* to deal with cases including multiple M1 atoms in one charge balancing group when using the balanced charge modification schemes, e.g., BRC2 scheme (see the revision history of *QMMM 2018*).

8.B.87. Test 2070 (*Gaussian*): Using the Gau_ext.acc script to perform Test 2061

The input files (`.crd`, `.dat`, and `.prm` files) of Test 2070 and Test 2061 are exactly same. Instead of executing the `.ml` script to run the calculation in Test 2061, the `Gau_ext.acc` script is executed in Test 2070. The `Gau_ext.acc` script associated with the newly modified `ex_shuttle`, `g03shuttle`, and `.ml` scripts (in *QMMM 2018*) enable each *Gaussian* energy

and gradient calculation during optimization to read the *Gaussian* checkpoint file from the previous run (see the revision history of *QMMM 2018*). We found that there are two advantages to executing the `Gau_ext.acc` script instead of the `.ml` script to perform QM/MM optimizations using the *Gaussian* external optimizer: (1) the efficiency of the calculation can be greatly improved (for Test 2061, the geometry is converged after 110 optimization steps, while for Test 2070 only 17 optimization steps are needed to converge the geometry, and it turns out that Test 2070 is about 12 times faster than Test 2061 although they have exactly same inputs); (2) the failures of *Gaussian* energy and gradient calculations during optimization, which often happen for QM/MM optimization on a complicated system such as MOF and will always lead to geometry distortion, can be avoided.

8.B.88. Test 301(*ORCA*): QM Single-Point Energy for CF_3CH_2OH

Test run 301 performs a QM single-point energy calculation for CF_3CH_2OH at the HF/STO_3G level by using ORCA.

8.B.89. Test 302(*ORCA*): QM Single-Point Gradient for CF_3CH_2OH

Test run 302 performs a gradient calculation for CF_3CH_2OH at the HF/STO_3G level by using ORCA.

8.B.90. Test 303(*ORCA*): QM Single-Point Hessian for CF_3CH_2OH

Test run 303 performs a Hessian calculation for CF_3CH_2OH at the HF/STO_3G level by using ORCA.

8.B.91. Test 304(*ORCA*): QM Optimization for H_2O

Test run 304 performs a QM geometry optimization for H_2O at the HF/STO_3G level by using ORCA.

8.B.92. Test 305(*ORCA*): QM Optimization for CF_3CH_2OH

Test run 305 performs a QM geometry optimization for CF_3CH_2OH at the HF/STO_3G level by using ORCA.

8.B.93. Test 306(*ORCA*): QM Optimization for CF_3-CH_2O^-

Test run 306 performs a QM geometry optimization for CF_3-CH_2O^- at the HF/_6_31G by using ORCA.

8.B.94. Test 307(*ORCA*): QM Optimization for Ace-Lys-NMe

Test run 307 performs a QM geometry optimization for Ace-Lys-NMe at the HF/STO_3G level by using ORCA.

8.B.95. Test 308(*ORCA*): QM Dynamics

Test run 308 performs a QM dynamics simulation of CF_3CH_2OH in a NVE ensemble at the HF/STO_3G level by using ORCA.

8.B.96. Test 3001(*ORCA*): QM/MM Single-point Energy

Test run 3001 performs a single-point energy calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/_6_31g, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.97. Test 3002(*ORCA*): QM/MM Single-point Gradient

Test run 3002 performs a single-point gradient calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/_6_31g, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme. The geometry in this test is the same as that in test 3001.

8.B.98. Test 3003(*ORCA*): QM/MM Single-point Hessian

Test run 3003 performs a single-point Hessian calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/_6_31g, and the MM force field is

OPLS-AA. The QM/MM boundary is treated by using the ME scheme. The geometry in this test is the same as that in test 3001.

8.B.99. Test 3004(*ORCA*): QM/MM Optimization (1)

Test run 3004 performs a geometry optimization for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/STO_3G, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.100. Test 3005(*ORCA*): QM/MM Optimization – ESP Charges through Atom Types

Test run 3005 performs geometry optimizations for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/STO_3G, and the MM force field is OPLS-AA except for the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.101. Test 3006(*ORCA*): QM/MM Single-point Energy on QM/MM Geometry (1) – ESP charges through Atom Types

Test run 3006 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 3005. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/_6_31g, and the MM force field is OPLS-AA. In particular, we note that the ESP charge specification is done by changing the MM charges for the corresponding atom types.

8.B.102. Test 3007(*ORCA*): QM/MM Single-point Energy on QM/MM Geometry (2) – ESP charges Through Atom Indices

Test run 3007 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 3005. This test is almost the same as test 3006, except that the ESP charge specification is done by changing the MM charges for the corresponding *atoms*. Users are advised to study the how the qmmm program handles the atom indices in a QM/MM calculation,

and they are also encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given atoms.

8.B.103. Test 3008(*ORCA*): QM/MM Single-point Energy – RCD Scheme

Test run 3008 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/_6_31g, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.104. Test 3009(*ORCA*): QM/MM Single-point Energy – RCD Scheme /CM2 Charges

Test run 3009 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, expect for the CM2 charges that are used for the CF_3 group. The CM2 charges for CF_3 are derived from CM2 charge model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The CM2 charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.105. Test 3010(*ORCA*): QM/MM Single-point Energy – RCD Scheme /CM3 Charges

Test run 3010 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, expect for the CM3 charges that are used for the CF_3 group. The CM3 charges for CF_3 are derived from CM3 charge model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The CM3 charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.106. Test 3011(*ORCA*): QM/MM Single-point Energy – RCD Scheme /ESP Charges

Test run 3011 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, expect for

the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.107. Test 3012(*ORCA*): QM/MM Single-point Energy – RCD Scheme /Löwdin Charges

Test run 3012 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, expect for the Löwdin charges that are used for the CF_3 group. The Löwdin charges for CF_3 are derived from Löwdin charge-model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The Löwdin charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.108. Test 3013(*ORCA*): QM/MM Single-point Energy – RCD Scheme /Mulliken Charges

Test run 3013 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except for the Mulliken charges that are used for the CF_3 group. The Mulliken charges for CF_3 are derived from Mulliken charge-model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The Mulliken charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.109. Test 3014(*ORCA*): QM/MM Single-point Energy – RC Scheme

Test run 3014 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except that the RC scheme is adopted.

8.B.110. Test 3015(*ORCA*): QM/MM Single-point Energy – Shift Scheme

Test run 3015 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except that the Shift scheme is adopted.

8.B.111. Test 3016(*ORCA*): QM/MM Single-point Energy – Z1 Scheme

Test run 3016 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except that the Z1 scheme is adopted.

8.B.112. Test 3017(*ORCA*): QM/MM Single-point Energy – Z2 Scheme

Test run 3017 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except that the Z2 scheme is adopted.

8.B.113. Test 3018(*ORCA*): QM/MM Single-point Energy – Z3 Scheme

Test run 3018 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except that the Z3 scheme is adopted.

8.B.114. Test 3019(*ORCA*): QM/MM Single-point Energy – SEE Scheme

Test run 3019 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except that the SEE scheme is adopted.

8.B.115. Test 3020(*ORCA*): QM/MM Single-point Energy – RCD2 Scheme

Test run 3020 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except that the RCD2 scheme is adopted.

8.B.116. Test 3021(*ORCA*): QM/MM Single-point Energy – ME Scheme

Test run 3021 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 305. The setups are almost the same as those in test 3008, except that the ME scheme is adopted.

8.B.117. Test 3022(*ORCA*): QM/MM Single-point Energy with Charged PS (1)

Test run 3022 performs a QM/MM single-point energy calculation for CF_3-CH_2O^- at the geometry optimized in test 306. The PS is CH_2O^-, and the SS is CF_3. The QM level is HF/_6_31G, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.118. Test 3023(*ORCA*): QM/MM Single-point Energy with Charged PS (2) – ESP Charges for SS

Test run 3023 performs a QM/MM single-point energy calculation for CF_3-CH_2O^- at the geometry optimized in test 306. The setups are the same as those in test 3022 except for the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.119. Test 3024(*ORCA*): QM/MM Optimization with Charged PS (1)

Test run 3024 performs a QM/MM optimization for CF_3-CH_2O^-. The PS is CH_2O^-, and the SS is CF_3. The QM level is HF/_6_31G, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.120. Test 3025(*ORCA*): QM/MM Optimization and Vibrational Analysis with Charged PS (2) – ESP Charges for SS

Test run 3025 performs a QM/MM optimization for CF_3-CH_2O^-, followed by a vibrational analysis. The setups are the same as those in test 3024 except for the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that

the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.121. Test 3026(*ORCA*): QM/MM Optimization – QM/MM Boundary does not Pass through a Covalent Bond

Test run 3026 performs a QM/MM optimization for the CH_2OH-CH_2OH…H_2O complex. The PS is the one of the CH_2OH group, which has a QM/MM boundary at one side going through the covalent C–C bond and a QM/MM boundary at the other side that does not pass through a covalent bond. The SS includes the other CH_2OH group and the H_2O. The QM level is HF/STO_3G, and the MM force field is OPLS-AA. The RCD scheme is used to treat the QM/MM boundary where it passes through the covalent bond.

8.B.122. Test 3027(*ORCA*): QM/MM Single-point Gradient for Ace-Lys-NMe with QM/MM Cutoff

Test run 3027 performs a single-point-gradient calculation for the Ace-Lys-NMe model system at the QM/MM level with the QM/MM cutoff. The center of the cutoff sphere is the 8^{th} atom (an C atom), and the cutoff radius is 3 Å. This is solely to show how to use the QM/MM cutoff; it is by no mean to suggest the use of 3 Å as the cutoff radius. The geometry is optimized in test307. The PS is the side chain, and the SS is the backbone. The QM level is HF/STO_3G, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD scheme.

8.B.123. Test 3028(*ORCA*): QM/MM Geometry Optimization for Ace-Lys-NMe using the PBRC Scheme with the QEq-SCT Method for Charge Equalization

Test run 3028 performs the geometry optimization for the Ace-Lys-NMe model system using the polarized-embedding scheme that allows the self-consistent mutual polarizations between the QM and MM subsystems near the boundary at the QM/MM level. The PS is the side chain, and the SS is the backbone. The QM level is HF/_6_31G, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RC scheme. The charge equalization is done by use of the QEq-SCT method.

8.B.124. Test 3029(*ORCA*): QM/MM Optimization

Test run 3029 performs a geometry optimization for CF_3-CH_2OH at the QM/MM level with the LBFGS algorithm. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/STO_3G, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.125. Test 3030(*ORCA*): QM/MM Optimization

Test run 3030 performs a geometry optimization for CF_3-CH_2OH at the QM/MM level using the internal optimizer with the LBFGS algorithm. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/STO_3G, and the MM force field is OPLS-AA. In the optimization, the CF_3 Cartesian coordinates are frozen and do not change. The QM/MM boundary is treated by using the RCD (default) scheme.

8.B.126. Test 3031(*ORCA*): QM/MM Dynamics

Test run 3031 performs a QM/MM dynamics simulation for CF_3CH_2OH + 9 H_2O in a NVE ensemble. The QM level is HF/6-31G, and the MM force field is OPLS-AA.

8.B.127. Test 3032(*ORCA*): Adaptive Partitioning QM/MM

Test run 3032 performs an adaptive QM/MM dynamics simulation on a water box with one Cl- ion in a NVE ensemble. The QM level is AM1, and the MM force field is CHARMM27.

8.B.128. Test 3033(*ORCA*): Adaptive Partitioning QM/MM

Test run 3033 performs an adaptive QM/MM dynamics simulation on a butanol + 2 H_2O in a NVE ensemble. The QM level is MP2/6-31G*, and the MM force field is a modified OPLS-AA.

8.B.129. Test 3034(*ORCA*): Adaptive Partitioning QM/MM

Test run 3034 performs an adaptive QM/MM dynamics simulation on a butanol + H_2O in a NVE ensemble. The QM level is AM1, and the MM force field is a modified OPLS-AA.

8.B.130. Test 3035(*ORCA*): Adaptive Partitioning QM/MM

Test run 3035 performs an adaptive QM/MM simulation on a Li+ ion + 8 H_2O in a NVE ensemble. One water molecule is moved while everything else is fixed.

8.B.131. Test 401(*GAMESS*): QM Single-Point Energy for CF_3CH_2OH

Test run 401 performs a QM single-point energy calculation for CF_3CH_2OH at the RHF/STO-3G level by using GAMESS.

8.B.132. Test 402(*GAMESS*): QM Single-Point Gradient for CF_3CH_2OH

Test run 402 performs a gradient calculation for CF_3CH_2OH at the RHF/STO-3G level by using GAMESS.

8.B.133. Test 403(*GAMESS*): QM Single-Point Hessian for CF_3CH_2OH

Test run 403 performs a Hessian calculation for CF_3CH_2OH at the RHF/STO-3G level by using GAMESS.

8.B.134. Test 404(*GAMESS*): QM Optimization for H_2O

Test run 404 performs a QM geometry optimization for H_2O at the RHF/STO-3G level by using GAMESS.

8.B.135. Test 405(*GAMESS*): QM Optimization for CF_3CH_2OH

Test run 405 performs a QM geometry optimization for CF_3CH_2OH at the RHF/STO-3G level by using GAMESS.

8.B.136. Test 406(*GAMESS*): QM Optimization for CF_3-CH_2O^-.

Test run 406 performs a QM geometry optimization for CF_3-CH_2O^- at the RHF/6-31G level by using GAMESS.

8.B.137. Test 4001(*GAMESS*): QM/MM Single-point Energy

Test run 4001 performs a single-point energy calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is RHF/6-31g, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.138. Test 4002(*GAMESS*): QM/MM Single-point Gradient

Test run 4002 performs a single-point gradient calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is RHF/6-31g, and the MM force field is OPLS-AA. The QM/MM boundary is treated using the mechanical embedding scheme. The geometry in this test is the same as that in test 4001.

8.B.139. Test 4003(*GAMESS*): QM/MM Single-point Hessian

Test run 4003 performs a single-point Hessian calculation for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is RHF/6-31g, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the mechanical embedding scheme. The geometry in this test is the same as that in test 4001.

8.B.140. Test 4004(*GAMESS*): QM/MM Optimization (1)

Test run 4004 performs a geometry optimization for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is RHF/STO-3G, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.141. Test 4005(*GAMESS*): QM/MM Optimization – ESP Charges through Atom Types

Test run 4005 performs geometry optimizations for CF_3-CH_2OH at the QM/MM level. The PS is CH_2OH, and the SS is CF_3. The QM level is RHF/STO-3G, and the MM force field is OPLS-AA except for the ESP charges that are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters

for given *atom types*. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.142. Test 4006(*GAMESS*): QM/MM Single-point Energy on QM/MM Geometry (1) – ESP charges through Atom Types

Test run 3006 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 4005. The PS is CH_2OH, and the SS is CF_3. The QM level is HF/6-31g, and the MM force field is OPLS-AA. In particular, we note that the ESP charge specification is done by changing the MM charges for the corresponding atom types.

8.B.143. Test 4007(*GAMESS*): QM/MM Single-point Energy on QM/MM Geometry (2) – ESP charges Through Atom Indices

Test run 4007 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 4005. This test is almost the same as test 4006, except that the ESP charge specification is done by changing the MM charges for the corresponding *atoms*. Users are advised to study the how the qmmm program handles atom indices in a QM/MM calculation, and they are also encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given atoms.

8.B.144. Test 4008(*GAMESS*): QM/MM Single-point Energy

Test run 4008 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The PS is CH_2OH, and the SS is CF_3. The QM level is RHF/3-21g, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.145. Test 4009(*GAMESS*): QM/MM Single-point Energy –CM2 Charges

Test run 4009 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The QM level is RHF/6-311g. The CM2 charges for CF_3 are derived from CM2 charge model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The CM2 charge specification is done by changing the MM charges for the corresponding *atom*

types. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.146. Test 4010(*GAMESS*): QM/MM Single-point Energy –CM3 Charges

Test run 4010 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The QM level is CCD/6-31g. The CM3 charges for CF_3 are derived from CM3 charge model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The CM3 charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.147. Test 4011(*GAMESS*): QM/MM Single-point Energy –ESP Charges

Test run 4011 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The QM level is MP2/STO-3G. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.148. Test 4012(*GAMESS*): QM/MM Single-point Energy –Löwdin Charges

Test run 4012 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The QM level is MP2/6-31G. The Löwdin charges for CF_3 are derived from Löwdin charge-model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The Löwdin charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.149. Test 4013(*GAMESS*): QM/MM Single-point Energy –Mulliken Charges

Test run 4013 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The QM level is B3LYP/STO-3G. The Mulliken charges for CF_3 are derived from Mulliken charge-model calculations for C_2F_6, so that the neutrality for CF_3 is assured. The Mulliken charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.150. Test 4014(*GAMESS*): QM/MM Single-point Energy

Test run 4014 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The QM level is SLATER/6-31G. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.151. Test 4015(*GAMESS*): QM/MM Single-point Energy

Test run 4015 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The QM level is CCSD/6-31G. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.152. Test 4016(*GAMESS*): QM/MM Single-point Energy

Test run 4016 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The QM level is RHF/DH. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.153. Test 4017(*GAMESS*): QM/MM Single-point Energy – Z2 Scheme

Test run 4017 performs a QM/MM single-point energy calculation for CF_3-CH_2OH at the geometry optimized in test 405. The setups are almost the same as those in test 4008, except that The QM level is CCSD(T)/6-31G.

8.B.154. Test 4018(*GAMESS*): QM/MM Single-point Energy with Charged PS (1)

Test run 4018 performs a QM/MM single-point energy calculation for CF_3-CH_2O^- at the geometry optimized in test 406. The PS is CH_2O^-, and the SS is CF_3. The QM level is RHF/6-31G, and the MM force field is OPLS-AA The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.155. Test 4019(*GAMESS*): QM/MM Single-point Energy with Charged PS (2) – ESP Charges for SS

Test run 4019 performs a QM/MM single-point energy calculation for CF_3-CH_2O^- at the geometry optimized in test 406. The setups are the same as those in test 4018 except that ESP charges are used for the CF_3 group. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality of CF_3 is assured. The ESP charge specification is made by changing the MM charge parameters for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*.

8.B.156. Test 4020(*GAMESS*): QM/MM Optimization with Charged PS (1)

Test run 4020 performs a QM/MM optimization for CF_3-CH_2O^-. The PS is CH_2O^-, and the SS is CF_3. The QM level is RHF/6-31G, and the MM force field is OPLS-AA. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.157. Test 4021(*GAMESS*): QM/MM Optimization and Vibrational Analysis with Charged PS (2) – ESP Charges for SS

Test run 4021 performs a QM/MM optimization for CF_3-CH_2O^-, followed by a vibrational analysis. The ESP charges for CF_3 are derived from ESP charge fitting for C_2F_6, so that the neutrality for CF_3 is assured. The ESP charge specification is done by changing the MM charges for the corresponding *atom types*. Users are encouraged to study the TINKER manual for the TINKER keywords in changing MM parameters for given *atom types*. The QM/MM boundary is treated by using the mechanical embedding scheme.

8.B.158. Test 4022(*GAMESS*): QM/MM Optimization – QM/MM Boundary does not Pass through a Covalent Bond

Test run 4022 performs a QM/MM optimization for the CH_2OH-CH_2OH…H_2O complex. The PS is the one of the CH_2OH group, which has a QM/MM boundary at one side going through the covalent C–C bond and a QM/MM boundary at the other side that does not pass through a covalent bond. The SS includes the other CH_2OH group and the H_2O. The QM level is RHF/STO-3G, and the MM force field is OPLS-AA. The mechanical embedding is used to treat the QM/MM boundary where it passes through the covalent bond.

8.C. Viewing the History of Geometry Optimization

If users use the QMMM internal optimizer, the history of geometry optimizations is given in the QMMM output file. In addition, a MOLDEN format file `molden.xyz` is created, where the history of the optimizations is stored. The `molden.xyz` file can be read by the MOLDEN program for visualization.

If users use the Gaussian external optimizer, the history of geometry optimizations is given in the *Gaussian* output file (e.g., `test2033.extopt`), which is directly viewable with the program *Gaussview*.

Chapter Nine

9

9. Computers, Operating Systems, anf Fortran Compilers

In each case we give the QMMM version number and the platforms (computers and operating systems) on which QMMM was tested and supported. For each computer and operating system, we also specify the Fortran compiler and the compiling script that were used for testing. Also listed are the QM and MM packages tested.

A. QMMM–V1.0

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
IBM Power3 (SP)	**AIX 5.1**	**XLF for AIX (comp_multi.ais)**	**TINKER 4.1 & 4.2**	***G03.c01***
IBM Power4 (Regatta)	**AIX 5.2**	**XLF for AIX (comp_multi.ais)**	**TINKER 3.5, 4.1, & 4.2**	***G03.c01***

B. QMMM–V 1.1

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
IBM Power4 (Regatta)	**AIX 5.2**	**XLF for AIX (comp_multi.ais)**	**TINKER 4.2**	***G03.c01* & *G03.d01.b1***
Athlon MP 2000+ (Zappa)	**Linux RedHat (Fedora Core x86)**	**G95 for Linux (comp_multi_g95.lux)**	**TINKER 4.2**	**ORCA 2.45**
SGI Altix (Altix)	**SGI Linux 3 with SGI Propack 3.4**	**G95 for Linux (comp_multi_g95.lux)**	**TINKER 4.2**	***G03.d01***
Intel Pentium 4/III Cluster (Netfinity)	**Linux RedHat (Enterprise 3)**	**PGF90 for Linux (comp_multi_pgi.lux)**	**TINKER 4.2**	***G03.b01.2*** [a)]

a) We found that the scratch files for the tests 2032 and 2033 are very large (> 11 GB) with the basis set specified in the tests, probably exceeding the allowed volume in the Netfinite computer where we tested the program, and this crashed the running of tests 2032 and 2033. When the basis set was reduced to a smaller one, e.g., STO-3G, and the sizes of the scratch files were correspondingly educed, and tests 2032 and 2033 ran correctly.

C. QMMM–V 1.2

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
IBM Power4 (Regatta)	**AIX 5.2**	**XLF for AIX (comp_multi.ais)**	**TINKER 4.2**	***G03.c01, G03.d01.b1,* & *GAMESS***
Athlon MP 2000+ (Zappa)	**Linux RedHat (Fedora Core x86)**	**G95 for Linux (comp_multi_g95.lux)**	**TINKER 4.2**	**ORCA 2.45 & *GAMESS***
SGI Altix (Altix)	**SGI Linux 3 with SGI Propack 3.4**	**G95 for Linux (comp_multi_g95.lux)**	**TINKER 4.2**	***G03.d01* & *GAMESS***
Intel Pentium 4/III Cluster (Netfinity)	**Linux RedHat (Enterprise 3)**	**PGF90 for Linux (comp_multi_pgi.lux)**	**TINKER 4.2**	***G03.b01.2*[a] & *GAMESS***

a) We found that the scratch files for the tests 2032 and 2033 are very large (> 11 GB) with the basis set specified in the tests, probably exceeding the allowed volume in the Netfinite computer where we tested the program, and this crashed the running of tests 2032 and 2033. When the basis set was reduced to a smaller one, e.g., STO-3G, and the sizes of the scratch files were correspondingly educed, and tests 2032 and 2033 ran correctly.

D. QMMM–V 1.3

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
IBM Power4 (Regatta)	**AIX 5.2**	**XLF for AIX (comp_multi.ais)**	**TINKER 4.2**	***G03.c01, G03.d01.b1,* & *GAMESS***

Athlon MP 2000+ (Zappa)	**Linux RedHat (Fedora Core x86)**	**G95 for Linux (comp_multi_g95.lux)**	**TINKER 4.2**	**ORCA 2.45 & *GAMESS***
SGI Altix (Altix)	**SGI Linux 3 with SGI Propack 3.4**	**G95 for Linux (comp_multi_g95.lux)**	**TINKER 4.2**	***G03.d01* & *GAMESS***
Intel Pentium 4/III Cluster (Netfinity)	**Linux RedHat (Enterprise 3)**	**PGF90 for Linux (comp_multi_pgi.lux)**	**TINKER 4.2**	***G03.b01.2*[a] & *GAMESS***
IBM Power5 (Ncip595)	**AIX 5L (5.3)**	**XLF for AIX (comp_multi.ais)**	**TINKER 4.2**	***G03.b04*[b]**

a) We found that the scratch files for the tests 2032 and 2033 are very large (> 11 GB) with the basis set specified in the tests, probably exceeding the allowed volume in the Netfinite computer where we tested the program, and this crashed the running of tests 2032 and 2033. When the basis set was reduced to a smaller one, e.g., STO-3G, and the sizes of the scratch files were correspondingly educed, and tests 2032 and 2033 ran correctly.

b) On this computer, one needs to add the keyword GDIIS in the GAUEXTOPTIONS keyword to make the test2033 run successfully.

E. QMMM−V 1.3.5

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
IBM Power4 (Regatta)	**AIX 5.2**	**XLF for AIX (comp_multi.ais)**	**TINKER 4.2**	***G03.c01, G03.d01.b1***
IBM Power5 (Ncip595)	**AIX 5L (5.3)**	**XLF for AIX (comp_multi.ais)**	**TINKER 4.2**	***G03.b04*[a]**

a) On this computer, one needs to add the keyword GDIIS in the GAUEXTOPTIONS keyword to make the test2033 run successfully.

F. QMMM−V 1.3.6

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
IBM Blade Center Cluster (AMD Opteron)	SuSe Linux Enterprise 9	Intel ifort version 11.0 (comp_multi_intel.lux)	TINKER 4.2	*G03.e01*, ORCA 2.6.35
SGI Calhoun Altix 1300 Cluster (Intel Xeon)	SuSe Linux Enterprise 10	Intel ifort version 11.0 (comp_multi_intel.lux)	TINKER 4.2	*G03.e01*, ORCA 2.6.35

G. QMMM−V 1.3.7

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
IBM Power5	AIX 5.2	XLF for AIX (comp_multi.ais)	TINKER 4.2	ORCA 2.5 & *GAMESS*
SGI Altix (Altix)	SGI Linux 3 with SGI Propack 3.4	G95 for Linux (comp_multi_g95.lux)	TINKER 4.2	*G03.e01* & *G03.d01*
IBM Blade Center Cluster (AMD Opteron)	SuSe Linux Enterprise 9	Portland 6.2 (comp_multi_pgi.lux) Intel ifort version 11.0 (comp_multi_intel.lux)	TINKER 4.2	*G03.e01*, ORCA 2.7.0
SGI Calhoun Altix 1300 Cluster (Intel Xeon)	SuSe Linux Enterprise 10	Intel ifort version 11.0 (comp_multi_intel.lux)	TINKER 4.2	*G03.e01*, ORCA 2.6.35

H. QMMM−V 1.3.8

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
IBM Blade Center Cluster	SuSe Linux Enterprise 9	Portland 6.2 (comp_multi_pgi.lux)	TINKER 5.1	*G03.e01*, ORCA 2.7.0

(AMD Opteron)		**Intel ifort version 11.0 (comp_multi_intel.lux)**		
SGI Calhoun Altix 1300 Cluster (Intel Xeon)	**SuSe Linux Enterprise 10**	**Intel ifort version 11.0 (comp_multi_intel.lux)**	**TINKER 5.1**	***G03.e01*, ORCA 2.6.35**

I. QMMM–V 1.4.0

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
Compchem2010 (Intel Xeon)	**RedhatLinux Enterprise 9**	**Gfortran, G95**	**TINKER 4.2, TINKER 5.1**	**ORCA 2.7.0, ORCA 2.8.0**
SGI Calhoun Altix 1300 Cluster (Intel Xeon)	**SuSe Linux Enterprise 10**	**Gfortran, Intel ifort version 11.0 (comp_multi_intel.lux)**	**TINKER 5.1**	***G03.e01*, ORCA 2.6.35**

J. QMMM 2015

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
HP ProLiant BL280c G6 Linux Cluster	**Linux**	**Intel ifort version 2013 (comp_multi_intel.lux)**	**TINKER 5.1 TINKER 6.3**	***G03.e01 G09.a02***

K. QMMM 2017

Computer	Operating System	Fortran Compiler (Script)	MM Program	QM Program
HP Linux distributed cluster (Intel Haswell E5-2680v3)	**CentOS 6.9**	**Intel ifort version 2017 (comp_multi_intel.lux)**	**TINKER 6.3**	***G16.a03 G09.e01 G09.a02***

L. QMMM 2018

Computer	Operating	Fortran Compiler	MM	QM

	System	(Script)	Program	Program
HP Linux distributed cluster (Intel Haswell E5-2680v3)	**CentOS 7.5**	**Intel ifort version 2017 (comp_multi_intel.lux)**	**TINKER 6.3**	***G16.c01*** ***G16.b01*** ***G16.a03*** ***G09.e01*** ***G09.a02***

Chapter Ten
10

10. Bibliography

1. Warshel, A., and Levitt, M. (1976) 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.
2. Singh, U. C., and Kollmann, P. A. (1986) A combined ab initio quantum mechanical and molecular mechanical method for carrying out simulations on complex molecular systems: Applications to the CH3Cl + Cl- exchange reaction and gas phase protonation of polyethers, J. Comput. Chem. 7, 718-730.
3. Field, M. J., Bash, P. A., and Karplus, M. (1990) A combined quantum mechanical and molecular mechanical potential for molecular dynamics simulations, J. Comput. Chem. 11, 700-733.
4. Gao, J., and Xia, X. (1992) A prior evaluation of aqueous polarization effects through Monte Carlo QM-MM simulations, Science 258, 631-635.
5. Gao, J. (1996) Methods and applications of combined quantum mechanical and molecular mechanical potentials, Rev. Comput. Chem. 7, 119-185.
6. Ferenczy, G. G., Rivail, J.-L., Surjan, P. R., and Naray-Szabo, G. (1992) NDDO fragment self-consistent field approximation for large electronic systems, J. Comput. Chem. 13, 830-837.
7. Thery, V., Rinaldi, D., Rivail, J.-L., Maigret, B., and Ferenczy, G. G. (1994) Quantum-mechanical computations on very large molecular systems: the local self-consistent field method, J. Comput. Chem. 15, 269-282.
8. Assfeld, X., and Rivail, J.-L. (1996) Quantum chemical computations on parts of large molecules: the ab initio local self consistent field method, Chem. Phys. Lett. 263, 100-106.
9. Monard, G., Loos, M., Thery, V., Baka, K., and Rivail, J.-L. (1996) Hybrid classical quantum force field for modeling very large molecules, Int. J. Quantum Chem. 58, 153-159.
10. Ferré, N., Assfeld, X., and Rivail, J.-L. (2002) Specific force field parameters determination for the hybrid Ab Initio QM/MM LSCF method, J. Comput. Chem. 23, 610-624.
11. Maseras, F., and Morokuma, K. (1995) IMOMM: a new integrated ab initio + molecular mechanics geometry optimization scheme of equilibrium structures and transition states, J. Comput. Chem. 16, 1170-1179.
12. Svensson, M., Humbel, S., Froese, R. D. J., Matsubara, T., Sieber, S., and Morokuma, K. (1996) ONIOM: A multilayered integrated MO+MM method for geometry optimizations and single point energy predictions. A test for Diels-Alder reactions and Pt(P(t-Bu)(3))(2)+H-2 oxidative addition, J. Phys. Chem. 100, 19357-19363.

13. Froese, R. D. J., Humbel, S., Svensson, M., and Morokuma, K. (1997) IMOMO(G2MS): A new high-level G2-like method for large molecules and its applications to Diels-Alder reactions, J. Phys. Chem. A 101, 227-233.
14. Dapprich, S., Komiromi, I., Byun, K. S., Morokuma, K., and Frisch, M. J. (1999) A new ONIOM implementation in Gaussian98. Part I. The calculation of energies, gradients, vibrational frequencies and electric field derivatives, THEOCHEM 461-462, 1-21.
15. Vreven, T., and Morokuma, K. (2000) On the application of the IMOMO (integrated molecular orbital + molecular orbital) method, J. Comput. Chem. 21, 1419-1432.
16. Vreven, T., Mennucci, B., da Silva, C. O., Morokuma, K., and Tomasi, J. (2001) The ONIOM-PCM method: Combining the hybrid molecular orbital method and the polarizable continuum model for solvation. Application to the geometry and properties of a merocyanine in solution, J. Chem. Phys. 115, 62-72.
17. Morokuma, K. (2002) New challenges in quantum chemistry: quests for accurate calculations for large molecular systems, Philos. Trans. R. Soc. London, A 360, 1149-1164.
18. Kerdcharoen, T., and Morokuma, K. (2002) ONIOM-XS: an extension of the ONIOM method for molecular simulation in condensed phase, Chem. Phys. Lett. 355, 257-262.
19. Stanton, R. V., Hartsough, D. S., and Merz Jr., K. M. (1994) An examination of a density functional / molecular mechanical coupled potential, J. Comput. Chem. 16, 113-128.
20. Monard, G., and Merz, K. M., Jr. (1999) Combined quantum mechanical/molecular mechanical methodologies applied to biomolecular systems, Acc. Chem. Res. 32, 904-911.
21. Gogonea, V., Westerhoff, L. M., and Merz, K. M., Jr. (2000) Quantum mechanical/quantum mechanical methods. I. A divide and conquer strategy for solving the Schroedinger equation for large molecular systems using a composite density functional-semiempirical Hamiltonian, J. Chem. Phys. 113, 5604-5613.
22. Gogonea, V., and Merz Jr., K. M. (2000) A quantum mechanical-Poisson–Boltzmann equation approach for studying charge flow between ions and a dielectric continuum, J. Chem. Phys. 112, 3227-3235.
23. Thompson, M. A. (1995) Hybrid quantum mechanical/molecular mechanical force field development for large flexible molecules: a molecular dynamics study of 18-crown-6, J. Phys. Chem. 99, 4794-4804.
24. Thompson, M. A., and Schenter, G. K. (1995) Excited states of the bacteriochlorophyll b dimer of rhodopseudomonas viridis: A QM/MM study of the photosynthetic reaction center that includes MM polarization, J. Phys. Chem. 99, 6374-6386.
25. Bakowies, D., and Thiel, W. (1996) Hybrid models for combined quantum mechanical and molecular mechanical approaches, J. Phys. Chem. 100, 10580-10594.
26. Bakowies, D., and Thiel, W. (1996) Semiempirical treatment of electrostatic potentials and partial charges in combined quantum mechanical and molecular mechanical approaches, J. Comput. Chem. 17, 87-108.
27. Antes, I., and Thiel, W. (1999) Adjusted connection atoms for combined quantum mechanical and molecular mechanical methods, J. Phys. Chem. A 103, 9290-9295.
28. Lennartz, C., Schaefer, A., Terstegen, F., and Thiel, W. (2002) Enzymatic reactions of triosephosphate isomerase: A theoretical calibration study, J. Phys. Chem. B 106, 1758-1767.

29. Eurenius, K. P., Chatfield, D. C., Brooks, B. R., and Hodoscek, M. (1996) Enzyme mechanisms with hybrid quantum and molecular mechanical potentials. I. Theoretical considerations, Int. J. Quantum Chem. 60, 1189-1200.
30. Das, D., Eurenius, K. P., Billings, E. M., Sherwood, P., Chatfield, D. C., Hodoscek, M., and Brooks, B. R. (2002) Optimization of quantum mechanical molecular mechanical partitioning schemes: Gaussian delocalization of molecular mechanical charges and the double link atom method, J. Chem. Phys. 117, 10534-10547.
31. Cummins, P. L., and Gready, J. E. (1997) Coupled semiempirical molecular orbital and molecular mechanics model (QM/MM) for organic molecules in aqueous solution, J. Comput. Chem. 18, 1496-1512.
32. Titmuss, S. J., Cummins, P. L., Rendell, A. P., Bliznyuk, A. A., and Gready, J. E. (2002) Comparison of linear-scaling semiempirical methods and combined quantum mechanical/molecular mechanical methods for enzymic reactions. II. An energy decomposition analysis, J. Comput. Chem. 23, 1314-1322.
33. Bersuker, I. B., Leong, M. K., Boggs, J. E., and Pearlman, R. S. (1997) A method of combined quantum mechanical (QM)/molecular mechanics (MM) treatment of large polyatomic systems with charge transfer between the QM and MM fragments, Int. J. Quantum Chem. 63, 1051-1063.
34. Kerdcharoen, T., Liedl, K. R., and Rode, B. M. (1996) A QM/MM simulation method applied to the solution of Li+ in liquid ammonia, Chem. Phys. 211, 313-323.
35. Tongraar, A., Liedl, K. R., and Rode, B. M. (1997) Solvation of Ca2+ in water studied by Born-Oppenheimer ab Initio QM/MM dynamics, J. Phys. Chem. A 101, 6299-6309.
36. Woo, T. K., Cavallo, L., and Ziegler, T. (1998) Implementation of the IMOMM methodology for performing combined QM/MM molecular dynamics simulations and frequency calculations, Theor. Chem. Acc. 100, 307-313.
37. Woo, T. K., Blöchl, P. E., and Ziegler, T. (2000) Towards solvation simulations with a combined ab initio molecular dynamics and molecular mechanics approach, THEOCHEM 506, 313-334.
38. Gao, J., Thompson, M. A., and Eds. (1998) Combined quantum mechanical and molecular mechanical methods: ACS Symp. Ser. 712, American Chemical Society, Washington, DC.
39. Gao, J., Amara, P., Alhambra, C., and Field, M. J. (1998) A generalized hybrid orbital (GHO) method for the treatment of boundary atoms in combined QM/MM calculations, J. Phys. Chem. A 102, 4714-4721.
40. Byun, K., and Gao, J. (2000) A combined QM/MM study of the nucleophilic addition reaction of methanethiolate and N-methylacetamide, Journal of Molecular Graphics and Modelling 18, 50-55.
41. Amara, P., Field, M. J., Alhambra, C., and Gao, J. (2000) The generalized hybrid orbital method for combined quantum mechanical/molecular mechanical calculations: formulation and tests of the analytical derivatives, Theor. Chem. Acc. 104, 336-343.
42. Gao, J., and Truhlar, D. G. (2002) Quantum mechanical methods for enzyme kinetics, Annu. Rev. Phys. Chem. 53, 467-505.
43. Truhlar, D. G., Gao, J., Alhambra, C., Garcia-Viloca, M., Corchado, J., Sanchez, M. L., and Villa, J. (2002) The incorporation of quantum effects in enzyme kinetics modeling, Acc. Chem. Res. 35, 341-349.

44. Devi-Kesavan, L. S., Garcia-Viloca, M., and Gao, J. (2003) Semiempirical QM/MM potential with simple valence bond (SVB) for enzyme reactions. Application to the nucleophilic addition reaction in haloalkane dehalogenase, Theor. Chem. Acc. 109, 133-139.
45. Pu, J., Gao, J., and Truhlar, D. G. (2004) Generalized hybrid orbital (GHO) method for combining ab Initio hartree-fock wave functions with molecular mechanics, J. Phys. Chem. A 108, 632-650.
46. Pu, J., Gao, J., and Truhlar, D. G. (2004) Combining self-consitent-charge density-functional tight-binding (SCC-DFTB) with molecular-mechanics by the generalized hybrid orbital (GHO) method, J. Phys. Chem. A 108, 5454-5463.
47. Åqvist, J., and Warshel, A. (1993) Simulation of enzyme reactions using valence bond force fields and other hybrid quantum/classical approaches, Chem. Rev. 93, 2523-2544.
48. Lyne, P. D., Hodoscek, M., and Karplus, M. (1999) A hybrid QM-MM potential employing hartree-fock or density functional methods in the quantum region, J. Phys. Chem. A 103, 3462-3471.
49. Reuter, N., Dejaegere, A., Maigret, B., and Karplus, M. (2000) Frontier bonds in QM/MM methods: A comparison of different approaches, J. Phys. Chem. A 104, 1720-1735.
50. Cui, Q., Elstner, M., Kaxiras, E., Frauenheim, T., and Karplus, M. (2001) A QM/MM implementation of the self-consistent charge density functional tight binding (SCC-DFTB) method, J. Phys. Chem. B 105, 569-585.
51. Riccardi, D., Schaefer, P., Yang, Y., Yu, H., Ghosh, N., Prat-Resina, X., König, P., Li, G., Xu, D., Guo, H., Elstner, M., and Cui, Q. (2006) Development of effective quantum mechanical/molecular mechanical (QM/MM) methods for complex biological process J. Phys. Chem. B 110, 6458-6469.
52. Hillier, I. H. (1999) Chemical reactivity studied by hybrid QM/MM methods, THEOCHEM 463, 45-52.
53. Hall, R. J., Hindle, S. A., Burton, N. A., and Hillier, I. H. (2000) Aspects of hybrid QM/MM calculations: the treatment of the QM/MM interface region and geometry optimization with an application to chorismate mutase, J. Comput. Chem. 21, 1433-1441.
54. Nicoll, R. M., Hindle, S. A., MacKenzie, G., Hillier, I. H., and Burton, N. A. (2001) Quantum mechanical/molecular mechanical methods and the study of kinetic isotope effects: modeling the covalent junction region and application to the enzyme xylose isomerase, Theor. Chem. Acc. 106, 105-112.
55. Kairys, V., and Jensen, J. H. (2000) QM/MM boundaries across covalent bonds: A frozen localized molecular orbital-based approach for the effective fragment potential method, J. Phys. Chem. A 104, 6656-6665.
56. Eichinger, M., Tavan, P., Hutter, J., and Parrinello, M. (1999) A hybrid method for solutes in complex solvents: Density functional theory combined with empirical force fields, J. Chem. Phys. 110, 10452-10467.
57. Röthlisberger, U., Carloni, P., Doclo, K., and Parrinello, M. (2000) A comparative study of galactose oxidase and active site analogs based on QM/MM Car-Parrinello simulations, J. Bio. Inorg. Chem. 5, 236-250.
58. Colombo, M. C., Guidoni, L., Laio, A., Magistrato, A., Maurer, P., Piana, S., Rohrig, U., Spiegel, K., Sulpizi, M., VandeVondele, J., Zumstein, M., and Röthlisberger, U. (2002)

Hybrid QM/MM Car-Parrinello simulations of catalytic and enzymatic reactions, Chimia 56, 13-19.

59. Laio, A., VandeVondele, J., and Rothlisberger, U. (2002) D-RESP: Dynamically generated electrostatic potential derived charges from quantum mechanics/molecular mechanics simulations, J. Phys. Chem. B 106, 7300-7307.
60. Sulpizi, M., Laio, A., VandeVondele, J., Cattaneo, A., Röthlisberger, U., and Carloni, P. (2003) Reaction mechanism of caspases: insights from QM/MM Ca-Parrinello simulations, Proteins: Struct., Funct., Genet. 52, 212-224.
61. Laio, A., Gervasio, F. L., VandeVondele, J., Sulpizi, M., and Röthlisberger, U. (2004) A variational definition of electrostatic potential derived charges, J. Phys. Chem. B 108, 7963-7968.
62. Zhang, Y., Lee, T.-S., and Yang, W. (1999) A pseudobond approach to combining quantum mechanical and molecular mechanical methods, J. Chem. Phys. 110, 46-54.
63. DiLabio, G. A., Hurley, M. M., and Christiansen, P. A. (2002) Simple one-electron quantum capping potentials for use in hybrid QM/MM studies of biological molecules, J. Chem. Phys. 116, 9578-9584.
64. Yang, W., and Drueckhammer, D. G. (2003) Computational study of the citrate synthase catalyzed deprotonation of acetyl-coenzyme a and fluoroacetyl-coenzyme A: Demonstration of a layered quantum mechanical approach, J. Phys. Chem. B 107, 5986-5994.
65. Field, M. J., Albe, M., Bret, C., Martin, F. P.-D., and Thomas, A. (2000) The dynamo library for molecular simulations using hybrid quantum mechanical and molecular mechanical potentials, J. Comput. Chem. 21, 1088-1100.
66. Amara, P., and Field, M. J. (2003) Evaluation of an ab initio quantum mechanical/molecular mechanical hybrid-potential link-atom method, Theor. Chem. Acc. 109, 43-52.
67. Philipp, D. M., and Friesner, R. A. (1999) Mixed ab initio QM/MM modeling using frozen orbitals and tests with alanine dipeptide and tetrapeptide, J. Comput. Chem. 20, 1468-1494.
68. Murphy, R. B., Philipp, D. M., and Friesner, R. A. (2000) A mixed quantum mechanics/molecular mechanics (QM/MM) method for large-scale modeling of chemistry in protein environments, J. Comput. Chem. 21, 1442-1457.
69. Murphy, R. B., Philipp, D. M., and Friesner, R. A. (2000) Frozen orbital QM/MM methods for density functional theory, Chem. Phys. Lett. 321, 113-120.
70. de Vries, A. H., Sherwood, P., Collins, S. J., Rigby, A. M., Rigutto, M., and Kramer, G. J. (1999) Zeolite structure and reactivity by combined quantum-chemical-classical calculations, J. Phys. Chem. B 103, 6133-6141.
71. Sherwood, P. (2000) Hybrid quantum mechanics/molecular mechanics approaches, in Modern Methods and Algorithms of Quantum Chemistry (Grotendorst, J., Ed.), pp 285-305, John von Neumann-Instituts, Jülich.
72. Turner, A. J., Moliner, V., and Williams, I. H. (1999) Transition-state structural refinement with GRACE and CHARMM: Flexiable QM/MM modeling for lactate dehydrogenase, Phys. Chem. Chem. Phys. 1, 1323-1331.
73. Moliner, V., and Williams, I. H. (2000) Flexiable QM/MM modelling embraces alternative mechanisms for lactate dehydrigenase, J. Chem. Soc., Chem. Commun., 1843-1844.

74. Hu, H., Elstner, M., and Hermans, J. (2003) Comparison of a QM/MM force field and molecular mechanics force fields in simulations of alanine and glycine "dipeptides" (Ace-Ala-Nme and Ace-Gly-Nme) in water in relation to the problem of modeling the unfolded peptide backbone in solution, Proteins: Struct., Funct., Genet. 50, 451-463.
75. Pitarch, J., Pascual-Ahuir, J. L., Silla, E., Tunon, I., and Ruiz-Lopez, M. F. (1999) Modeling beta-lactam interactions in aqueous solution through combined quantum mechanics-molecular mechanics methods, J. Comput. Chem. 20, 1401-1411.
76. Swart, M. (2003) AddRemove: A new link model for use in QM/MM studies, Int. J. Quantum Chem. 91, 177-183.
77. Loferer, M. J., Loeffler, H. H., and Liedl, K. R. (2003) A QM-MM interface between CHARMM and TURBOMOLE: Implementation and application to systems in bulk phase and biologically active systems, J. Comput. Chem. 24, 1240-1249.
78. Mordasini, T., Curioni, A., and Andreoni, W. (2003) Why do divalent metal ions either promote or inhibit enzymatic reactions? - The case of BamHI restriction endonuclease from combined quantum-classical simulations, J. Biol. Chem. 278, 4381-4384.
79. Worthington, S. E., and Krauss, M. (2001) The claisen rearrangement of an unusual substrate in chorismate mutase, J. Phys. Chem. B 105, 7096-7098.
80. Poteau, R., Ortega, I., Alary, F., Solis, A. R., Barthelat, J. C., and Daudey, J. P. (2001) Effective group potentials. 1. Method, J. Phys. Chem. A 105, 198-205.
81. Kongsted, J., Osted, A., Mikkelsen, K. V., and Christiansen, O. (2003) Coupled cluster/molecular mechanics method: implementation and application to liquid water, J. Phys. Chem. A 107, 2578-2588.
82. Lin, H., and Truhlar, D. G. (2005) Redistributed charge and dipole schemes for combined quantum mechanical and molecular mechanical calculations, J. Phys. Chem. A 109, 3991-4004.
83. Lin, H., and Truhlar, D. G. (2007) QM/MM: What have we learned, where are we, and where do we go from here?, Theor. Chem. Acc. 117, 185-199.
84. Zhang, Y., Lin, H., and Truhlar, D. G. (2007) Self-consistent polarization of the boundary in the redistributed charge and dipole scheme for combined quantum-mechanical and molecular-mechanical calculations, J. Chem. Theory Comput. 3, 1378-1398.
85. Heyden, A., Lin, H., and Truhlar, D. G. (2007) Adaptive partitioning in combined quantum mechanical and molecular mechanical calculations of potential energy functions for multiscale simulations, J. Phys. Chem. B 111, 2231-2241.
86. Zhang, Y., and Lin, H. (2008) Flexible-boundary quantum-mechanical/molecular-mechanical calculations: Partial charge transfer between the quantum-mechanical and molecular-mechanical subsystems, J. Chem. Theory Comput. 4, 414-425.
87. Zhang, Y., and Lin, H. (2010) Flexible-boundary QM/MM calculations: II. Partial charge transfer across the QM/MM boundary that passes through a covalent bond Theor. Chem. Acc. 126, 315-322.
88. Senn, H. M., and Thiel, W. (2007) QM/MM studies of enzymes, Current Opinion in Chemical Biology 11, 182-187.
89. Senn, H. M., and Thiel, W. (2007) QM/MM methods for biological systems, Top. Curr. Chem. 268, 173-290.

90. Altun, A., Kumar, D., Neese, F., and Thiel, W. (2008) Multireference ab initio quantum mechanics/molecular mechanics study on intermediates in the catalytic cycle of cytochrome P450cam, The Journal of Physical Chemistry A 112, 12904-12910.
91. Kästner, J., Senn, H. M., Thiel, S., Otte, N., and Thiel, W. (2006) QM/MM free-energy perturbation compared to thermodynamic integration and umbrella sampling: Application to an enzymatic reaction, J. Chem. Theory Comput. 2, 452-461.
92. Hans Martin Senn, W. T. (2009) QM/MM methods for biomolecular systems, Angew. Chem. Int. Ed. 48, 1198-1229.
93. Sherwood, P., Brooks, B. R., and Sansom, M. S. P. (2008) Multiscale methods for macromolecular simulations, Curr. Opin. Struct. Biol. 18, 630-640.
94. Lu, Z., and Yang, W. (2004) Reaction path potential for complex systems derived from combined ab initio quantum mechanical and molecular mechanical calculations, J. Chem. Phys. 121, 89-100.
95. Hu, H., Lu, Z., and Yang, W. (2007) QM/MM minimum free-energy path: Methodology and application to triosephosphate isomerase, J. Chem. Theory Comput. 3, 390-406.
96. Hu, H., and Yang, W. (2008) Free energies of chemical reactions in solution and in enzymes with ab initio quantum mechanics/molecular mechanics methods, Annu. Rev. Phys. Chem. 59, 573-601.
97. Shao, Y., and Kong, J. (2007) YinYang atom: A simple combined ab Initio quantum mechanical molecular mechanical model, J. Phys. Chem. A 111, 3661-3671.
98. Zhang, Y. (2005) Improved pseudobonds for combined ab initio quantum mechanical'molecular mechanical methods, J. Chem. Phys. 122, 024114/024111-024124/024117.
99. Zhang, Y., Liu, H., and Yang, W. (2000) Free energy calculation on enzyme reactions with an efficient iterative procedure to determine minimum energy paths on a combined ab initio QM/MM potential energy surface, J. Chem. Phys. 112, 3483-3492.
100. Lev, B., Zhang, R., de la Lande, A., Salahub, D., and Noskov, S. Y. (2010) The QM-MM interface for CHARMM-deMon, J. Comput. Chem. 31, 1015-1023.
101. Zeng, X., Hu, H., Hu, X., Cohen, A. J., and Yang, W. (2008) Ab initio quantum mechanical/molecular mechanical simulation of electron transfer process: Fractional electron approach, J. Chem. Phys. 128, 124510.
102. Walker, R. C., Crowley, M. F., and Case, D. A. (2008) The implementation of a fast and accurate QM/MM potential method in Amber, J. Comput. Chem. 29, 1019-1031.
103. Pu, J., Gao, J., and Truhlar, D. G. (2005) Generalized hybrid-orbital method for combining density functional theory with molecular mechanicals, ChemPhysChem 6, 1853-1865.
104. Zerner, M. C. (1991) Semiempirical molecular orbital methods, Rev. Comput. Chem. 2, 313-365.
105. Hehre, W. J., Radom, L., Schleyer, P. v. R., and Pople, J. A. (1986) Ab initio molecular orbital theory, Wiley, New York.
106. Kohn, W., Becke, A. D., and Parr, R. G. (1996) Desity functional theory of electronic structure, J. Phys. Chem. 100, 12974-12980.
107. Pearlman, D. A., Case, D. A., Caldwell, J. W., Ross, W. S., Cheatham, T. E., III, DeBolt, S., Ferguson, D., Seibel, G., and Kollman, P. A. (1995) "AMBER", a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics

and free energy calculations to stimulate the structural and energetic properties of molecules, Comput. Phys. Commun. 91, 1-42.

108. MacKerell, A. D., Jr., Bashford, D., Bellott, M., Dunbrack, R. L., Evanseck, J. D., Field, M. J., Fischer, S., Gao, J., Guo, H., Ha, S., Joseph-McCarthy, D., Kuchnir, L., Kuczera, K., Lau, F. T. K., Mattos, C., Michnick, S., Ngo, T., Nguyen, D. T., Prodhom, B., Reiher, W. E., III, Roux, B., Schlenkrich, M., Smith, J. C., Stote, R., Straub, J., Watanabe, M., Wiorkiewicz-Kuczera, J., Yin, D., and Karplus, M. (1998) All-atom empirical potential for molecular modeling and dynamics studies of proteins, J. Phys. Chem. B 102, 3586-3616.
109. Jorgensen, W. L., Maxwell, D. S., and Tirado-Rives, J. (1996) Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids, J. Am. Chem. Soc. 118, 11225-11236.
110. Lii, J. H., and Allinger, N. L. (1989) Molecular mechanics. The MM3 force field for hydrocarbons. 3. The van der Waals' potentials and crystal data for aliphatic and aromatic hydrocarbons, J. Am. Chem. Soc. 111, 8576-8582.
111. Lii, J. H., and Allinger, N. L. (1989) Molecular mechanics. The MM3 force field for hydrocarbons. 2. Vibrational frequencies and thermodynamics, J. Am. Chem. Soc. 111, 8566-8575.
112. Allinger, N. L., Yuh, Y. H., and Lii, J. H. (1989) Molecular mechanics. The MM3 force field for hydrocarbons. 1., J. Am. Chem. Soc. 111, 8551-8566.
113. Schmidt, M. W., Baldridge, K. K., Boatz, J. A., Elbert, S. T., Gordon, M. S., Jensen, J. H., Koseki, S., Matsunaga, N., Nguyen, K. A., Su, S. J., Windus, T. L., Dupuis, M., and Montgomery, J. A. (1993) General atomic and molecular electronic structure system, J. Comput. Chem. 14, 1347-1363.
114. Frisch, M. J., Trucks, G. W., Schlegel, H. B., Scuseria, G. E., Robb, M. A., Cheeseman, J. R., Montgomery, J., J. A., Vreven, T., Kudin, K. N., Burant, J. C., Millam, J. M., Iyengar, S. S., Tomasi, J., Barone, V., Mennucci, B., Cossi, M., Scalmani, G., Rega, N., Petersson, G. A., Nakatsuji, H., Hada, M., Ehara, M., Toyota, K., Fukuda, R., Hasegawa, J., Ishida, M., Nakajima, T., Honda, Y., Kitao, O., Nakai, H., Klene, M., Li, X., Knox, J. E., Hratchian, H. P., Cross, J. B., Adamo, C., Jaramillo, J., Gomperts, R., Stratmann, R. E., Yazyev, O., Austin, A. J., Cammi, R., Pomelli, C., Ochterski, J. W., Ayala, P. Y., Morokuma, K., Voth, G. A., Salvador, P., Dannenberg, J. J., Zakrzewski, V. G., Dapprich, S., Daniels, A. D., Strain, M. C., Farkas, O., Malick, D. K., Rabuck, A. D., Raghavachari, K., Foresman, J. B., Ortiz, J. V., Cui, Q., Baboul, A. G., Clifford, S., Cioslowski, J., Stefanov, B. B., Liu, G., Liashenko, A., Piskorz, P., Komaromi, I., Martin, R. L., Fox, D. J., Keith, T., Al-Laham, M. A., Peng, C. Y., Nanayakkara, A., Challacombe, M., Gill, P. M. W., Johnson, B., Chen, W., Wong, M. W., Gonzalez, C., and Pople, J. A. (2003) Gaussian03, Version E.01 ed., Gaussian, Inc., Pittsburgh PA.
115. Neese, F. (2008) ORCA, Version 2.6 ed., University of Bonn, Bonn.
116. Ponder, J. W. (2004) TINKER, Version 4.2 ed., Washington University, St. Louis, MO.
117. Rodgers, J. M., Lynch, B. J., Fast, P. L., Chuang, Y.-Y., Pu, J., Zhao, Y., and Truhlar, D. G. (2003) MULTILEVEL, Version 3.1/G03 ed., University of Minnesota, Minneapolis.
118. Corchado, J. C., Chuang, Y.-Y., Fast, P. L., Villà, J., Hu, W.-P., Liu, Y.-P., Lynch, G. C., Nguyen, K. A., Jackels, C. F., Melissas, V. S., Lynch, B. J., Rossi, I., Coitiño, E. L., Fernandez-Ramos, A., Pu, J., Albu, T. V., Steckler, R., Garrett, B. C., Isaacson, A. D., and

Truhlar, D. G. (2002) POLYRATE, Version 9.1 ed., University of Minnesota, Minneapolis.

119. Rappé, A. K., and Goddard, W. A. (1991) Charge equilibration for molecular dynamics simulations, J. Phys. Chem. 95, 3358-3363.
120. Mortier, W. J., Ghosh, S. K., and Shankar, S. (1986) Electronegativity equalization method for the calculation of atomic charges in molecules, J. Am. Chem. Soc. 108, 4315-4320.
121. Bultinck, P., Langenaeker, W., Lahorte, P., De Proft, F., Geerings, P., Waroquier, M., and Tollenaere, J. P. (2002) The electronegativity equalization method I: Parameterization and validation for atomic charge calculations, J. Phys. Chem. A 106, 7887-7894.
122. Perdew, J. P., Parr, R. G., Levy, M., and Balduz, J. L. (1982) Density-Functional Theory for Fractional Particle Number: Derivative Discontinuities of the Energy, Phys. Rev. Lett. 49, 1691-1694.
123. Tavernelli, I., Vuilleumier, R., and Sprik, M. (2002) Ab initio dynamics for molecules with variable numbers of electrons, Phys. Rev. Lett. 88, 213002/213001-213002/213004.
124. Dapprich, S., Ujaque, G., Maseras, F., Lledos, A., Musaev, D. G., and Morokuma, K. (1996) Theory does not support an osmaoxetane intermediate in the osmium-catalyzed dihydroxylation of olefins, J. Am. Chem. Soc. 118, 11660-11661.
125. McDonald, N. A., and Jorgensen, W. L. (1998) Development of an all-atom force field for heterocycles. Properties of liquid pyrrole, furan, diazoles, and oxazoles, J. Phys. Chem. B 102, 8049-8059.
126. Jorgensen, W. L., and McDonald, N. A. (1998) Development of an all-atom force field for heterocycles. Properties of liquid pyridine and diazenes, THEOCHEM 424, 145-155.
127. Price, M. L. P., Ostrovsky, D., and Jorgensen, W. L. (2001) Gas-phase and liquid-state properties of esters, nitriles, and nitro compounds with the OPLS-AA force field, J. Comput. Chem. 22, 1340-1135.
128. Rizzo, R. C., and Jorgensen, W. L. (1999) OPLS all-atom model for amines: Resolution of the amine hydration problem, J. Am. Chem. Soc. 121, 4827-4836.
129. Kaminski, G. A., Friesner, R. A., Tirado-Rives, J., and Jorgensen, W. L. (2001) Evaluation and reparametrization of the OPLS-AA force field for proteins via comparison with accurate quantum chemical calculations on peptides, J. Phys. Chem. B 105, 6474-6487.
130. Swope, W. C., Andersen, H. C., Berens, P. H., and Wilson, K. R. (1982) A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters, J. Chem. Phys. 76, 637-649.
131. Berendsen, H. J. C., Postma, J. P. M., Gunsteren, W. F. v., DiNola, A., and Haak, J. R. (1984) Molecular dynamics with coupling to an external bath, J. Chem. Phys. 81, 3684-3690.
132. Jorgensen, W. L., Chandrasekhar, J., Madura, J. D., Impey, R. W., and Klein, M. L. (1983) Comparison of simple potential functions for simulating liquid water, J. Chem. Phys. 79, 926-935.
133. Berendsen, H. J. C., Postma, J. P. M., Gunsteren, W. F. v., and Hermans, J. (1981) Interaction Models for Water in Relation to Protein Hydration, in Intermolecular Forces (Pullman, B., Ed.), pp 331-342, D. Reidel Publishing Company, Dordrecht.

134. Andersen, H. C. (1980) Molecular dynamics simulations at constant pressure and/or temperature, J. Chem. Phys. 72, 2384-2393.
135. Bulo, R. E., Ensing, B., Sikkema, J., and Visscher, L. (2009) Toward a practical method for adaptive QM/MM simulations, J. Chem. Theory Comput. 5, 2212-2221.
136. Ferré, N., and Olivucci, M. (2003) The amide bond: pitfalls and drawbacks of the link atom scheme, THEOCHEM 632, 71-82.
137. Kahn, K., and Bruice, T. C. (2002) Parameterization of OPLS–AA force field for the conformational analysis of macrocyclic polyketides, J. Comput. Chem. 23, 977-996.
138. Singh, U. C., and Kollman, P. A. (1984) An approach to computing electrostatic charges for molecules, J. Comput. Chem. 5, 129-145.
139. Besler, B. H., Merz, K. M., Jr., and Kollman, P. A. (1990) Atomic charges derived from semi-empirical methods., J. Comput. Chem. 11, 431-439.
140. Francl, M. M., Carey, C., Chirlian, L. E., and Gange, D. M. (1996) Charges fit to electrostatic potentials. ii. can atomic charges be unambiguously fit to electrostatic potentials?, J. Comput. Chem. 17, 367-383.
141. Bayly, C. I., Cieplak, P., Cornell, W. D., and Kollman, P. A. (1993) A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: The RESP model, J. Phys. Chem. 97, 10269-10280.
142. Thompson, J. D., Cramer, C. J., and Truhlar, D. G. (2003) Parameterization of charge model 3 for AM1, PM3, BLYP, and B3LYP, J. Comput. Chem. 24, 1291-1304.
143. Winget, P., Thompson, J. D., Xidos, J. D., Cramer, C. J., and Truhlar, D. G. (2002) Charge model 3: A class IV charge model based on hybrid density functional theory with variable exchange, J. Phys. Chem. A 106, 10707-10717.
144. Li, J., Williams, B., Cramer, C. J., and Truhlar, D. G. (1999) A class IV charge model for molecular excited states., J. Chem. Phys. 110, 724-733.
145. Li, J., Zhu, T., Cramer, C. J., and Truhlar, D. G. (1998) New class IV charge model for extracting accurate partial charges from wave functions, J. Phys. Chem. A 102, 1820-1831.
146. Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992) Numerical Recipes in Fortran77, 2nd ed., p 406ff, Cambridge University Press, Cambridge.
147. Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992) Numerical Recipes in Fortran77, 2nd ed., p 418ff, Cambridge University Press, Cambridge.
148. Baker, J. (1985) An algorithm for the location of transition states, J. Comput. Chem. 7, 385-395.
149. Culot, P., Dive, G., Nguyen, V. H., and Ghuysen, J. M. (1992) A quasi-Newton algorithm for first-order saddle-point location Theor. Chem. Acc. 82, 189-205.
150. Powell, M. S. D. (1971) On the convergence of the variable metric algorithm, J Inst. Math. Appl. 7, 21-36.
151. Ypma, T. J. (1995) Historical development of the Newton-Raphson method, SIAM Review 37, 531-551.
152. Fast, P. L., Corchado, J. C., Sánchez, M. L., and Truhlar, D. G. (1999) Multi-coefficient correlation method for quantum chemistry, J. Phys. Chem. A 103, 5129-5136.
153. Woon, D. E., and Thom H. Dunning, J. (1994) Gaussian basis sets for use in correlated molecular calculations. IV. Calculation of static electrical response properties, J. Chem. Phys. 100, 2975-2988.

154. Kendall, R. A., Thom H. Dunning, J., and Harrison, R. J. (1992) Electron affinities of the first-row atoms revisited. Systematic basis sets and wave functions, J. Chem. Phys. 96, 6796-6806.
155. Curtiss, L. A., Raghavachari, K., Redfern, P. C., Rassolov, V., and Pople, J. A. (1998) Gaussian-3 (G3) theory for molecules containing first and second-row atoms, J. Chem. Phys. 109, 7764-7776.
156. Curtiss, L. A., Redfern, P. C., Raghavachari, K., and Pople, J. A. (2001) Gaussian-3X (G3X) theory: Use of improved geometries, zero-point energies, and Hartree–Fock basis sets, J. Chem. Phys. 114, 108-117.
157. Fast, P. L., Sánchez, M. L., and Truhlar, D. G. (1999) Multi-coefficient Gaussian-3 method for calculating potential energy surfaces, Chem. Phys. Lett. 306, 407-410.
158. Lynch, B. J., Zhao, Y., and Truhlar, D. G. (2003) Effectiveness of diffuse basis functions for calculating relative energies by density functional theory., J. Phys. Chem. A 107, 1384-1388.
159. Wang, B., Truhlar, D. G. (2010) Combined Quantum Mechanical and Molecular Mechanical Methods for Calculating Potential Energy Surfaces: Tuned and Balanced Redistributed-Charge Algorithm, J. Chem. Theory Comput. 6, 359–369.
160. Wang, B.; Truhlar, D. G. (2011) Geometry Optimization Using Tuned and Balanced Redistributed Charge Schemes for Combined Quantum Mechanical and Molecular Mechanical Calculations, Phys. Chem. Chem. Phys. 13, 10556–10564.
161. Wang, B.; Truhlar, D. G (2010) Including Charge Penetration Effects in Molecular Modeling, J. Chem. Theory Comput. 6, 3330–3342.
162. Wang, B.; Truhlar, D. G. (2013) Tuned and Balanced Redistributed Charge Scheme for Combined Quantum Mechanical and Molecular Mechanical (QM/MM) Methods and Fragment Methods: Tuning Based on the CM5 Charge Model, J. Chem. Theory Comput. 9, 1036–1042.
163. Walker, R. C.; Crowley, M. F.; Case, D. A. (2008) The Implementation of a Fast and Accurate QM/MM Potential Method in Amber, J. Comput. Chem. 29, 1019–1031.
164. Wu, X.-P.; Gagliardi, L.; Truhlar, D. G. (2018) Combined Quantum Mechanical and Molecular Mechanical Method for Metal-Organic Frameworks: Proton Topologies of NU-1000, Phys. Chem. Chem. Phys. 20, 1778–1786.

Chapter Eleven

11

11. Revision History and Version Information

11.A. Version 1.0

Finalized on Feb. 28, 2005

Released on Feb. 28, 2005

Authors: Hai Lin and Donald G. Truhlar

Released in 2005, this is the first distributed version. This version is based in part on the previous MULTILEVEL code, and it has been tested with *Gaussian*, version c.0.1 for the QM packages and with TINKER for the MM package. We have tested the current version of QMMM with three versions of TINKER: version 3.5,[3] version 4.1, and version 4.2, and the test runs in the current version of QMMM are made to call TINKER 4.2, which is the current version of TINKER (http://dasher.wustl.edu/tinker/).

11.B. Version 1.0.1

This is a bug-fixed version of the QMMM version 1.0. Two bugs in the internal optimizer were fixed.

1. There was a bug is in the internal optimizer (subroutine *ef*), and it terminates the optimization at the last step. In the *ef* subroutine, the *nvar* is the number of variable to be optimized, which should be smaller than $3N - 5$ for a linear molecule or $3N - 6$ for a nonlinear molecule. Thus the *eigval(i)* should be ranging from 1 to *nvar* instead of from 1

[3] For TINKER 3.5, we have so far only tested the mechanical embedding scheme using MM3 force field.

to *n3tm* during the initialization. The correction is straightforward by changing *n3tm* to *nvar*, and this was done in the subroutine *ef*.

2. There was a bug in the test runs using internal optimizer. The internal optimizer was designed for multi-level (QM/MM) optimization, and it did not support single-level optimization, as stated in the User's Manual. However, by mistake, the testruns 201 – 204 contain QM optimizations. Correction was made by replacing these test runs by QM single-point energy (test201), gradient (test202), and Hessian (test203) calculations, as well as a QM (pre)-optimization done by *Gaussian*.

11.C. Version 1.1

Finalized on May 18, 2006

Released on May 18, 2006

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

Released in 2006, this is an enhanced version of the QMMM version 1.0.1. The most noticeable enhancement is the implementation of calls to ORCA as the electronic-structure program. Also Gaussian versions b.01, c.01, and d.01 are supported. Improvement was also made to allow the use of MM Hessian for geometry optimization.

1 Calls to ORCA electronic structure package is added in the present QMMM package. These results in the following changes.

- In the revision, seven newly added subroutines are: `orcastipn`, `orcainp`, `orcaoute`, `orcaoutg`, `orcaouth`, `orcaouto`, and `wordstr` in the file named `orca.F`.

- In the subroutines `progehk`, `progesthk`, `progghk`, `proggsthk`, `proghhk`, and `progohk`, calls to ORCA package were added.

- The subroutines ef, `progehk`, `progesthk`, `progghk`, `proggsthk`, `proghhk`, qmmmhhk, rmlopt, opthhk, `progohk`, and two modules `files` and `para` were revised.

- A shuttle script called `orcashuttle` was added in the `/script` subdirectory.

- Currently, ORCA only works on our linux cluster running REDHAT operating system, where the calls to ORCA have been tested.

- The `testrun` and `testo` subdirectories were reorganized, both of which contain three subsubdirectories: `tinker`, `g03`, and `orca`. The `tinker` subsubdirectory contains single-level MM test calculations calling TINKER. The `g03` subsubdirectory has two subsubsubdirectories: `qm` and `qmmm`, where single-level QM and multi-level QM/MM test calculations calling *Gaussian* are given, respectively. The `orca` subsubdirectory is similar to the `g03` subsubdirectory; it contains however QM and QM/MM test calculations calling ORCA.

- Currently, QM/MM single-point energy is available for QM methods of HF, DFT, and MP2. The MP2 energy is search by the keyword “MP2 energy” instead of “Total energy” in the ORCA output file. We are hoping that in future ORCA will put the energy in a statement of “Final energy” in the output file.

- Currently, QM/MM single-point gradient is available for QM methods of HF and DFT, for which ORCA provide analytic gradients on both the QM atoms and the background point charges.

- Currently, QM/MM single-point Hessian is available for QM methods of HF and DFT with mechanical embedding. Currently, ORCA provides numerical Hessian on the QM atoms, but does not provide Hessian for the background point charges. Therefore QM/MM Hessian with electric embedding is not calculated.

- When DFT method is selected as the QM method, to be in consistent with the ORCA input format, the METHOD keyword (in the QMKEY list) must be set to *DFT*, and the functional is specified in the OPTION list as follow:

```
QMKEY
        Method    dft
    Options
            >! b3lyp
        End
End
```

or

```
QMKEY
        Method    dft
    Options
            >%method functional B3lyp
        >       end
        End
End
```

The format of the input functional is the same as ORCA, except that each line begins with ">", which is an indicator of ORCA keywords. The reason of using this indicator is that ORCA also uses END as keyword as QMMM does. The indicator ">" in the above example makes these two END keywords distinguishable.

2 The HESSIAN keyword in the MULTIOPT section, which specifies the Hessian to be used in the geometry optimization, are revised to include these four options: *unitmat* (a scaled unit matrix), *mm* (a Hessian calculated at the molecular mechanics level), *qmmm* (a Hessian calculated at the QM/MM level), and *lowqm* (a Hessian at a user-specified lower QM level). The initial Hessian can be obtained by any one of these four options,

but the Hessian recalculated during geometry optimization can only be obtained with the *mm*, *qmmm*, and *lowqm* options.

3 Our tests for geometry optimizations for stable molecules show that the internal optimizer work well for locating minima, but we have some difficulty in optimizing the saddle point using the *newt*, *newt2*, or *ef* options of the ALGORITHM keyword in the MULTIOPT section. We suspected that it was due to poor guess of the saddle point geometry. This should be verified in future; in particular after the surface scan (constrained optimization) functionality is implemented in the QMMM program.

4 In this version, there are four tests for the MM calculations, in comparison with nine MM tests in the version 1.0. The five tests that were deleted are geometry optimizations employing an external (*Gaussian*) optimizer. The present five tests for the MM calculations are: a single-point energy calculation (test101), a single-point gradient calculation (test102), a single-point Hessian calculation (test103), and a geometry pre-optimization employing the optimizer in TINKER (test104).

5 We found that different versions of *Gaussian* (*g03.b01*, *g03.c01*, and *g03.d01*) require slightly different procedures to invoked the *external* option. More specifically, the input and output files required for *g03.d01* version are in the scratch directory, while those of *g03.c01* and *g03.b01* are in the directory where the input files locate. Thus, we place the shuttle scripts for calling different versions of *Gaussian* in the different subdirectories: `g03.b01`, `g03.c01`, and `g03.d01` under the `script` directory. Depending on which *Gaussain03* version the user is using, user should copy the corresponding scripts into the `script` directory during the installation.

11.D. Version 1.1.1

This is a bug-fixed version of the QMMM version 1.1. Two bugs concerning parsing ORCA output files were fixed.

1. For the capped primary systems (CPS) embedded in a distribution of background point charges, the energy can be formally written as a sum of three components:

$$E(\text{QM**;CPS}) = E(\text{QM;CPS|CPS}) + E(\text{QM;CPS|BGC}) + E(\text{Coul;BGC|BGC})$$

Here, E(QM;CPS|CPS) denotes interactions within CPS atoms, E(QM;CPS|BGC) denotes interactions between PS atoms and background point charges, and E(MM;BGC|BGC) denotes Coulombic interactions within background point charges. The last component E(MM;BGC|BGC) however is also calculated as part of the MM energy for the entire system, E(MM;ES). Therefore, a double counting is presented.

In *Gaussian* all three components are calculated, and the double counting is indeed presented. In ORCA, however, only the first two components are computed, and a double counting is Not presented. (The energy derivatives are calculated in the same manner as energy in *Gaussian* and in ORCA). Therefore, different treatments are needed to work out the QM/MM energy for *Gaussian* and for ORCA:

For *Gaussian*, one needs to do an additional calculation to get the electrostatic interactions within the background point charges, i.e., E(Coul;BGC:BGC). Then this energy E(BGC:BGC) is subtracted from the total QM/MM energy to avoid double counting. For ORCA such an addition step is not needed.

2. Currently, two kinds of QM calculations by ORCA are supported by qmmm, the first kind being variational methods including semi-empirical, HF, and DFT, and the second kind being perturbation methods, specifically, MP2. The keywords for searching the final energy are different for these two kinds of QM methods. For the variational methods, the keywords are "*total energy*". For the perturbation method MP2, the keyword is "*mp2 total energy*". A conditional branching is now introduced to check whether the QM calculation is variational or perturbation using the searching keywords "*orca mp2 calculation*" before really search for the energy using appropriate keywords.

3. There was a bug in the `comp_multi_pgi.lux` and `comp_multi_g95.lux` scripts in the `qmmm1.2/script` directory. The variable *`multidir`* was incorrectly set as "*`~/test`*". Actually, it should be set to the QMMM path. That is one should use

```
multidir = `cat ~/.qmmm_path1.1`
```

4. There was a mistake in the Section 8.A Installation Instructions (step 5) in the manual. The numbers of the files in the `obj` and `mod` directories were wrong. The correct statement will be: There are 11 files in the `mod` directory and 12 files in the `obj` directory.

5. There was a bug in the `test#.ml` scripts for running the QMMM tests with *g03* in the `qmmm1.2/testrun/g03/qmmm/test#` directories. The line

```
cp $scriptdir/Gau_External2 $wrkdir/Gau_External2
```

is used in the QMMM calculations with the *g03.d01* version. For the *g03.b01* and *g03.c01*, the above line should be commented out.

11.E. Version 1.2

Finalized on May 30, 2006

Released on Jun. 1, 2006

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

Released in 2006, this is an enhanced version of the QMMM version 1.1.1. The most noticeable enhancement is the implementation of calls to GAMESS(-US) as the electronic-structure program. As a result, GAMESS, ORCA, and *Gaussian* (versions b.01, c.01, and d.01) are supported.

1. Call to GAMESS electronic structure package is added in the present QMMM package. These results in the following changes.

a. In the revision, nine newly added subroutines are: `gmsinp`, `gmsmeth`, `gmsoute`, `gmsoutg`, `gmsouth`, `gmsouto`, `scase`, `lcasel`, and len`word` in the file named `gamess.F`.

- In the subroutines `progehk`, `progghk`, `proghhk`, and `progohk`, calls to GAMESS package were added.

- The subroutines qmmmehk, `progehk`, `progghk`, `qmmmghk`, `proghhk`, qmmmhhk, `progohk`, and one `module files` were revised.

- A shuttle script called `gmsshuttle` was added in the `/script` subdirectory. In the calculation with GAMESS, the single-point energy is read from the output file: `gms.out`, and the Gradient, Hessian and optimized geometry are read from the `#.dat` file. Thus, after finishing the QM calculation, the `#.dat` file must be copied to the QMMM working directory.

- In the `testrun` and `testo` subdirectories, there was an additional `gamess` directory, which contains two subsubdirectories: `qm` and `qmmm`, where single-level QM and multi-level QM/MM test calculations calling GAMESS are given, respectively.

- Currently, QM/MM calculation with GAMESS is available for QM methods of HF, DFT, MP2, semiempirical and CC (coupled cluster) with mechanical embedding. Currently, GAMESS does not support gradient calculations with the background point charges, and actually GAMESS discourages users to do calculations with background point charges. Therefore QM/MM calculation with electronic embedding is not available in the current version of QMMM if GAMESS is selected to be the QM package.

- The basis set specification in GAMESS is quite complicated and we have adopted a convention to keep the QMMM input in consistent with the GAMESS input format as much as possible.

  First, the value of BASIS keyword (in the QMKEY list) is ignored. Next, the basis sets are actually specified in the OPTION list. An example of doing a calculation using the 6-31G basis set is as follow:

  QMKEY
  Basis 6-31g
  Options
  ! $basis gbasis=n31 ngauss=6 $end
  End
  End

  Here, the *6-31g* following *Basis* is a comment and will be ignored by the QMMM program. However, we suggest user to keep this comment, because it helps people to recognize the basis set. The line

  ! $basis gbasis=n31 ngauss=6 $end

  listed as the options are the actual specification of the basis set in the GAMESS format, except that "*!*" given at the very beginning of this line; the "*!*"is an indicator of GAMESS keywords.

- The QM method specification in GAMESS is rather complicated. We have tried hard to simplify the method specification so that it is in consistent with QMMM input format as much as possible. Below, we list the QM methods in GAMESS supported by QMMM and the corresponding values to be given for the METHOD keyword (in the QMKEY list):

1) The SCF type calculations (those specified in the SCFTYP keyword in the $CONTRL group of GAMESS):

   RHF UHF ROHF GVB MCSCF

2) MP2 calculation (in GAMESS, one needs to give the value “2” of the MPLEVL keyword in the $CONTRL group):

   MP2

3) The CI calculations (those specified in the CITYP keyword in the $CONTRL group of GAMESS):

   CIS ALDET ORMAS FSOCI GENCI
   GUGA

4) The coupled-cluster (CC) calculations (those specified in the CCTYP keyword in the $CONTRL group of GAMESS):

   LCCD CCD CCSD CCSD(T)
   CR-CC R-CC CR-CCL CCSD(TQ)
   EOM-CCSD CR-CC(Q) CR-EOM

5) The DFT calculations (those specified in the DFTTYP keyword in the $DFT group of GAMESS):

   SLATER BECKE GILL PBE VWN
   LYP OP SVWN SLYP SOP
   GLYP GVWN GOP PBEVWN

PBELYP	PBEOP	BHHLYP	B3LYP	BVWN
BLYP	BOP	XALPHA	DEPRISTO	
CAMA	BALF	PWLOC	BPWLOC	
GAMB	XVWN	XPWLOC	SPWLOC	
WIGNER	WS	WIGEXP		

6) The semi-empirical calculation (those specified in the GBASIS keyword in the $BASIS group of GAMESS):

MNDO AM1 PM3

11.F. Version 1.3

Finalized on Oct. 30, 2006

Released on Jan. 8, 2007

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

The most noticeable enhancement in version 1.3 is the implementation of the polarized-boundary redistributed charge (PBRC) and the polarized-boundary redistributed charge and dipole (PBRCD) schemes that account for the self-consistent mutual polarization between the QM and MM subsystems near the boundary. Both Gaussian and ORCA are supported electronic-structure programs for all method in version 1.3. Improvement was also made to allow the use of a cutoff in embedded-CPS calculations, i.e., to include in the embedded-CPS calculations only those SS partial atomic charges that are within a (preset) distance from the PS. In addition, the new version of QMMM also permits partial optimizations where only subsets of atoms are optimized (when the *Gaussian* external options are invoked). We also implemented a low-storage energy minimization scheme, which does not require Hessian information and is useful for large molecules such as proteins. Finally, more flexibility is introduced such that the charges and multiplicities of CPS and ES can be different. More details are given below:

1. The CHARGE and MULTIPLICITY keywords specifying the charge and multiplicity of the CPS, respectively, were added to the QM keyword list in the QM/MM section. With these two new

keywords, the charge and multiplicity of the CPS are allowed to differ from those of the ES in QM/MM calculations.

a. By default, the charge and multiplicity of the CPS are set to those of the ES.

b. For full-QM calculations that are carried out through the QMMM interface, which is allowed but not recommended, the charge and multiplicity are read from the MULTIGEN section because only the ES is concerned.

c. The subroutines `defqmmm`, `rqmmm`, `mlehook`, `progehk`, `g03inp`, `qmmmehk`, `progesthk`, `g03stinp`, `mlghook`, `progghk`, `qmmmghk`, `proggsthk`, `mlhook`, `proghhk`, `qmmmhhk`, `proghsthk`, `progohk`, and one module `input` were revised.

d.

2. In the MULTIOPT section, the PARTIAL keyword was added to allow carrying out a partial optimization using the *Gaussian* optimizer (via *Gaussian*'s external option).

a. In a partial optimization, only selected atoms (called moving atoms) are allowed to move, while the other atoms (called fixed atoms) are fixed to their present coordinates When the PARTIAL keyword is in effect, QMMM only passes the coordinates of the moving atoms to the *Gaussain* optimizer, and the *Gaussian* optimizer does not "see" those fixed atoms. However, the energies and gradients that are required by the *Gaussian* optimizer to determine the coordinate displacements for the moving atoms are still calculated by QMMM in the presence of the fixed atoms. That is, although the *Gaussian* optimizer does not see the fixed atoms, it can still "feel" the existence of the fixed atoms.

b. In the new implementation of partial optimization, the original `gau_ext_opt` subroutine was renamed as `gau_ext_opt1`, and four new subroutines were added: `gau_ext_opt`, `gau_ext_opt2`, `g03partinp`, and `g03parto`, all in the file named `gau_ext_opt.F`.

c. The branching between the full and partial optimizations is determined in the `gau_ext_opt` subroutine, where the full optimization procedure calls the

`gau_ext_opt1` subroutine, and the partial optimization procedure calls the `gau_ext_opt2` subroutine. The `g03parto` subroutine reads the geometry of the moving atoms and passes the geometry to the whole system, so that the whole system geometry is updated, and single-point calculations for QMMM energies and gradients can be done.

d. The two PERL scripts calling *Gaussian* are revised: GAU_EXTERNAL for the *g03.c01* and *g03.b01* versions and GAU_EXTERNAL2 for the *g03.d01* version.

e. The subroutines `defgen`, `rmlopt`, and one module `input` were revised.

f. The PARTIAL keyword has three valid options: PARTCHARGE, PARTMULT and PARTATM, whose meanings are given below:

PARTCHARGE — The charge of the moving atoms
PARTMULT — The multiplicity of the moving atoms
PARTATM — The list of the IDs for all the moving atoms

It should be noted that the charge and multiplicity of the moving atoms can be different from the charge and multiplicity of the CPS or the ES.

An input example for the partial-optimization calculation employing the *Gaussian* optimizer is as follow:

**MULTIOPT*
ALGORITHM GAUEXT
METHOD QMMM
PARTIAL
Partcharge 0
Partmult 1
Partatm

5 9 11 13 19

23 25

End

END

3. The newly added QMMMCUTOFF keyword in the QM/MM section allows the embedded-QM calculation includes only a subset of the MM background point charges that are within a distance from a user-defined center. The center is not necessarily an atom, although one can specify an atom to be the center. If one specifies an atom as the center, the center may change coordinates, e.g., in a geometry optimization. Alternatively, one can specify the center by providing its Cartesian coordinates, and those coordinates will be fixed in all calculations. One must select either one of the two options at a time. The cutoff distance is specified by the user.

   a. In the revision implementing the QM/MM cutoff, seven newly added subroutines are: `qmmmcutoff`, `qmmmcogeom`, `progmmstehk`, `progesthk`, `progmmstghk`, `proggsthk`, and `cutoffgrad`, all in the file named `cutoff.F`.

   b. The subroutines `defqmmm`, `rqmmm`, `read5`, and one module `input` were revised.

   c. The subroutines, whose original names were `progmmstehk`, `progesthk`, `progmmstghk`, and `proggsthk`, were renamed `progmmstehk1`, `progesthk1`, `progmmstghk1`, and `proggsthk1`, respectively.

   d. Currently, the QM/MM cutoff option is not available in the calculations of QM/MM Hessians or in geometry optimizations where QM/MM Hessians are needed.

   e. The QMMMCUTOFF keyword has three valid options: CUTOFFCENTID, CUTOFFCENTXYZ and CUTOFFRAD.

      CUTOFFCENTID — The ID of the atom as the cutoff center

CUTOFFCENTXYZ — The coordinates of the cutoff center

CUTOFFRAD — The cutoff radius

An example of the calculation using QMMMCUTOFF keyword is as follow:

QMMMCUTOFF
Cutoffcentxyz 1.2 3.2 4.1
Cutoffrad 10.0
END

4. In the QM/MM section, the POLAR keyword was added. The polarized-boundary calculations using this keyword account for self-consistent mutual polarizations between the QM and MM subsystems near the boundary. The polarization option is available in the calculations with the RC or RCD schemes employing *Gaussian* and ORCA. The implementations for the other schemes such as Shift and SEE are in plan.

   a. Seventeen newly added subroutines are: `chargcoor`, `chreset`, `eeapc`, `eecharge`, `eemch`, `g03chinp`, `g03potc`, `g03outfp`, `g03outop`, `g03outesp`, `orcastpotinp`, `orcaoutpot`, `polareech`, `progespp`, `qeqchk`, `qeqchg`, and `ridlist` in the file `eepolar.F`. One newly added module, which contains the parameters of the charge calculation, is `ehard` in the file `module.F`. One newly added script is `orcapotshuttle`. The subroutines solving the linear equation, which were taken from LAPACK and BLAS library, are in the file `lib.F`.

   b. The subroutines `defqmmm`, `rqmmm`, `qmmmehk`, `t41keyadd`, `qmmmghk`, `qmmmhhk`, and one module `input` were revised.

   c. In the polarized-boundaary calculations using ORCA, the subroutine orcastpotinp writes three input file: `orca.inp`, `orca.pc` and `orca.pot.xyz`. Here, `orca.inp` is the

`orca` input file. In order to calculate the electrostatic potential on the SS atoms, the `keepdensity` keyword was added in the input file as follows:

%scf keepdensity true end

The file `orca.pc` contains the background charge input file for the embedded-QM calculations for the CPS. The file `orca.pot.xyz` contains the grid input file for the electrostatic potential calculations, i.e., it contains the positions of the SS atoms. In the electrostatic potential calculations, the script `orcapotshuttle` first runs an `orca` job to obtain the QM wavefunction, and then it calculates the electrostatic potentials at the SS atoms using the command

orca_vpot orca.gbw orca.scfp.tmp orca.pot.xyz orca.pot

The file `orca.pot` contains the electrostatic potentials calculated at the positions of the SS atoms.

d. The POLAR keyword has seven valid options: METHOD, MPOT, PARAMETER, MAXDQ, RMSDQ, CYCLE, GROUPNUM, and GROUP. Using the PARAMETER keyword, one could input the parameters for the charge calculation; the user-input parameter will override the default parameters implemented in QMMM.

e. We implemented three literature methods that are based on the principle of electronegativity equalization for the determination of the background charges in the polarization treatments: the charge equalization method proposed by Rappé and Goddard (QEQRG), a modified version of the charge equalization method by Bakowies and Thiel (QEQBT), and the electronegativity equalization method by Mortier and coworkers (EEM). Bultinck et al. had listed in Table 1 of Ref. several sets of EEM parameters, and we found that the set of parameters developed by Mortier and coworkers showed best agreements between the EEM- and QM-calculated dipole moments for small organic molecules in our test calculations. Thus, we adopt in the present study the EEM

parameters by Mortier and coworkers, and the corresponding QM/MM calculations are denoted EEM.

f. The MPOT keyword has two valid options: UM1 and UQ0. In the calculations using the UM1 keyword, the external electric field is calculated before one redistributes the charge on the M1 atom (q_{M1}), while in the calculations using the UQ0 keyword, the field is calculated after q_{M1} is redistributed (q_0). In either way, the redistributed charges q_0 do not change value during the mutual polarization treatment, and the redistribution of q_{M1} can be done only once before entering the loop of the self-consistent polarization calculations.

We found that actually the results calculated by the two methods (UM1 and UQ0) are very close to each other, e.g., the proton affinities and the C–C bond distances for the seven small organic molecules in our test suite. We recommend the UQ0 scheme because it is easier to understand. Therefore the UQ0 is the default.

6. The limited-memory BFGS quasi-Newton algorithm was added in the MULTIOPT section. The code is based on the TINKER MINIMIZE subroutine with necessary modifications. The keyword of the algorithm is LBFGS. In the optimizations using this algorithm, only the gradients are calculated. The initial Hessian is a unit matrix. This optimizer is particularly designed for energy minimization for large-size molecules such as proteins. One should not expect very tight convergence by using this algorithm. The goal of using this algorithm is to provide a quick minimization of the whole protein so as to remove unfavorable contacts (e.g., two side chains in very close contacts). The resulting geometry can be further studied by partial optimizations where only the active site is optimized while keeping the surroundings fixed. A full optimization for the whole protein using standard second-order algorithms seems impractical at this stage.

a. In the revision adding the LBFGS keyword, five newly added subroutines are: `minlbfgs`, `search`, `lbfgsoptg`, `dispxyzg` and `discov` in the file named `lbfgs.F`.

b. The subroutines `mlohook, defgen, rmlopt, insumry` and on module `input` were revised.

c. In the LBFGS optimization, twelve keywords GCOMP, RMSGRAD, MAXDX, RMSDX, MAXSTEP, MINSTEP, CAPPA, MAXSLOPE, MAXANG, INTPOLAT, NITER, and MINENR are used. Four of them, GCOMP, RMSGRAD, MAXDX and RMSDX define the convergence criteria.

11.G. Version 1.3.1

Finalized on Nov. 20, 2007

Released on Nov. 20, 2007

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

This is a bug-fixed version of QMMM version 1.3. In this version, the `t41shuttle` is revised such that large systems can be calculated. The TINKER program automatically outputs gradient components at all atoms during gradient calculations if the number of atoms are equal to or smaller than 999. When the number of atoms are larger than 999, the TINKER program will ask user to decide whether the gradient components at all atoms are printed out, and this cause the QMMM program stop running. Our previous test runs and applications are small systems, and thus we did not discover the problem. In the version 1.3.1, this bug is fixed.

11.H. Version 1.3.2

Finalized on Nov. 20, 2007

Released on Nov. 20, 2007

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

This is a bug-fixed version of QMMM version 1.3.1. In the previous versions, the maximum connectivity of one atom, as specified in the `.crd` file was 4. This is now increased to 6 in version 1.3.2. Also, now the QMMM input files (the `.crd` and `.inp` and `.dat` files) allow up to 80 characters in each line.

11.I. Version 1.3.3

Finalized on Nov. 20, 2007

Released on Nov. 20, 2007

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

This is a bug-fixed version of QMMM version 1.3.2. In the QM/MM section, the QMKEY keyword allows several options to specify how the QM calculations should be done, one of which is lines contained in the OPTIONS list. When calling *Gaussian*, those lines beginning with

"!2" are going to be written in the *Gaussian* input file after the keyword NONBOND. In the version 1.3, this does not work properly; in the version 1.3.3, this bug is fixed.

11.J. Version 1.3.4

Finalized on Nov. 20, 2007

Released on Nov. 20, 2007

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

This is an enhanced version of QMMM version 1.3.3. In this version, for QMMM test runs that calls *Gaussian* for QM calculations, the `ex_shuttle` and `.ml` scripts were modified such that users can provide previously obtained *Gaussian* checkpoint files for the QM calculations.

The `.ml` script copies the `.chk` file to the working directory and renames it to `g03.chk`. This allows both single-point calculations and optimizations using the QMMM internal optimizer read necessary information from the *Gaussian* checkpoint file. Accordingly, one should add in the OPTIONS of the QMKEY keyword the following *Gaussian* keyword:

```
%chk=g03
```

The `ex_shuttle` script, which is used in optimizations employing the *Gaussian* optimizer via the `external` option, copies the `g03.chk` file to the directory where the QMMM gradient calculations are going to perform and renames it `g03.chk`.

11.K. Version 1.3.5

Finalized on Nov. 20, 2007

Released on Nov. 20, 2007

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

This is an enhanced version of QMMM version 1.3.4. The major changes are: Partial optimization using *Gaussian* optimizer was modified, and several keywords are added to make it more user-friendly. A noticeable enhancement is the implementation of the flexible-boundary

treatment that allows partial charge transfers between the PS and SS (i.e., between the QM and MM subsystems); currently this option works only in the situations where the QM/MM boundary does not go through a covalent bond. Another notable enhancement is in the Hessian calculations, where we have implemented numerical (full or partial) QM/MM Hessian. Calling ORCA optimizer as the external optimizer has also been added, but that is waiting for full test because the corresponding ORCA version has not been finalized yet.

1. In this version, the partial optimization using *Gaussian* optimizer was modified. As in the version 1.3, the atoms that are frozen at their positions are called frozen atoms, while the atoms that are allowed to change coordinates during the optimization are called active atoms. In the version 1.3 of QMMM, only the active atoms are passed to the *Gaussian* external optimizer, and that creates some problems since *Gaussian* does the optimization in internal coordinates. In this version, QMMM can also pass to *Gaussian* the frozen atoms that are surrounding the active atoms to avoid those problems.

One has three options in specifying those frozen atoms that will be passed to Gaussian. The first option is not passing such frozen atoms. The second option is to pass the so-called 1st layer frozen atoms, which are frozen atoms that are directly bonded to the active atoms. The third option is to pass both the 1st and 2nd layers of frozen atoms; the 2nd layer frozen atoms are those frozen atoms that are directly bonded to the 1st layer frozen atoms. The selection of the frozen atoms is done by specifying the EXTLAYERNUM option in the PARTIAL keyword of the MULTIOPT section.

Another change in the partial optimization procedure is that now users have two ways to specify the active atoms. The first way is to list the atomic index for the active atoms, which has been implemented in version 1.3. The second way, which is new in this version, is to specify a sphere via its center and radius, and the QMMM program would make the atoms within the sphere active atoms. The center can be either an atomic index or a point specified by its Cartesian coordinate.

A. Five new subroutines have been added in the file named `gau_part_opt.F`: `dispartoptinf`, `partaftm`, `partatm`, `partextlayer`, and `partoptatm`.

B. The subroutines `defgen`, `gau_ext_opt2`, `g03partinp`, `g03partouto`, `rmlopt` and the module `input` were revised.

C. Five keywords EXTLAYERNUM, PARTCENTID, PARTCENTXYZ, PARTINIT, and PARTRAD were added.

D. The keyword EXTLAYERNUM has three options: 0, 1, or 2. Option 0 means only the active atoms are passed to the *Gaussian* optimizer. Option 1 means both the active atoms and the 1st layer of frozen atoms are passed to the *Gaussian* optimizer. Option 2 means the active atoms, the 1st layer frozen atoms, and the 2nd layer of frozen atoms are passed to the *Gaussian* optimizer.

E. If the keyword PARTINIT is specified, the QMMM program will print out the list of active atoms in partial optimization and stop. This is useful for users to check and decide which atoms are included in the partial optimization.

F. The script `Gau_external` was revised.

2. In this version, the flexible-boundary treatment is implemented, which allows partial charge to be transferred between the PS and the SS. Currently this option works only in the situations where the QM/MM boundary does not go through a covalent bond. The treatment is invoked by specifying the FLEXBOUND keyword in the *QM/MM section.

A. In the file named `flexbound.F`, 27 new subroutine are added: `atmparam`, `calpx`, `chkinp`, `convg`, `cpsgasi`, `cpscoor`, `cpsmu`, `cpsmu1`, `difimu`, `dqcon`, `dvarzero`, `fbbpc`, `fbgacoor`, `fbpot`, `fbqtot`, `fpccp`, `fpcharg`, `fbgpot`, `g03bgpotinp`, `g03outep`, `gqtot`, `maxtrixj12`, `muvspx`, `polgchargcal`, `progcpe`, `slinequ`, and `updatcharg`.

B. The subroutines `defqmmm`, `rqmmm`, `qmmmehk` and the module `input` were revised.

C. In the FLEXBOUND keyword, 11 options are added: CALCHARGMETH, CHARGE, FBGROUPID, GROUP, MAXCHARGTRANS, MAXCYCLE, MAXDQ, MULT, PARAMETER, RMSDQ, and TEMPERATURE.

D. In the flexible-boundary treatment, one needs to specify the charges and multiplicities for the PS in both the reduced state and the oxidized state. The first state, which can be either the reduced state or the oxidized state, is often set to the state where the PS carries the normal formal charge, e.g., the Na^+ state where the Na center carries a formal charge of +1 e. The charge and multiplicity of the first state are specified by the QMKEY keyword. For the second state, e.g., the Na state where the Na center carries a formal charge of 0, the charge and multiplicity are specified in the FLEXBOUND keyword by the `charge` and `multi` options.

E. The subroutine `fbbpc` is called by the subroutine `qmmmehk`.

F. In the present implementation of the flexible-boundary treatment, only single-point energy calculations are possible, and the PS is not covalently bonded to the SS.

G. In the flexible-boundary calculation using *Gaussian*, the subroutine `g03bgpotinp` is called to write the input file for the electronic-structure calculations with background charges using the *Gaussian* keyword `charge` instead of using the *Gaussian* keyword ONIOM. The *Gaussian* calculations using the `charge` keyword avoid the need of at least a covalent bond connecting the PS and SS, but do not provide gradients at the background point charges.

3. In the version 1.3.5, only the test runs calling electronic structure package *Gaussian* are fully tested. Test runs calling ORCA are waiting for the corresponding orca version to be finalized and officially released.

11.L. Version 1.3.6

Finalized on May 7, 2009

Released on May 7, 2009

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

This is an enhanced version of QMMM version 1.3.5. The code has been revised in many aspects. The most important improvement is the acceleration of QM/MM gradient calculations and geometry optimizations, which are now much faster than in the previous versions. Also, the definitions for the maximum number of atoms that can be handled in the energy, gradient, and hessian calculations and geometry optimizations are clarified.

1. The subroutines `g03stinp` in the file `ehooks.F` and `g03chinp` in the `eepolar.F` file are revised, such that gradient calculations are significantly faster than in the previous versions for electronic embedding schemes invoking *Gaussian* as the electronic-structure package.

2. The subroutine `g03stinp` in the file `ehooks.F` and `g03chinp` in the `eepolar.F` file are revised, such that no covalent bond is needed to cut for electronic embedding schemes invoking *Gaussian* as the electronic-structure package.

3. Two new test runs (`test2053` and `test2054`) of QM/MM gradient calculation invoking *Gaussian* as the electronic-structure package are added. Both describe a CF_3CH_2OH molecule surrounding by 26 H_2O molecules. In one test run, the QM/MM boundary passes through the C–C covalent bond and the CH_2OH moiety is the PS, while no covalent bond is cut in the other test run, where the whole CF_3CH_2OH molecule is the PS. The H_2O molecules are always in the SS.

4. Partial optimization is implemented for internal optimizer employing the limited-memory BFGS algorithm; the FROZEATM keyword specifying atoms whose coordinates are to be frozen during optimizations. The subroutines `minlbfgs`, `search`, `lbfgsoptg`, `rmlopt` and one module `input` were revised. A new test run (`test3030`) using the

keyword was added. However, this option has not yet been extensively tested. Please use with caution.

5. Comments on the maximum numbers of atoms are added in the code, in particular, the `module.F` file, to explicitly indicate the separation of maximum number of atoms for energy and gradient calculations from the maximum number of atoms for hessian calculations and geometry optimization using internal optimizer. The maximum number of atoms in energy and gradient calculations is set to 9900, while the maximum number of atoms for hessian calculations and internal optimization is set to 500. For geometry optimizations using the *Gaussian* optimizer, the maximum number of atoms is the same as in energy calculations, i.e., 9900; however, please aware that Gaussian will require huge amount of memory for doing geometry optimizations for very large system due to the size of the hessian matrix.

   A section (Section 4.L.6) is added to the User's Manual to clarify this point and help users in modifying the maximum numbers of atoms for even larger systems.

6. In this version, the subroutines in the `numhess.F` file are revised. Initially, the maximum numbers of atoms in the input parsing and in the hessian calculations use temporally assigned numbers; now they are changed such that they agree with the number defined in the `module.F` file.

7. In the version 1.3.6, only the test runs calling electronic structure package *Gaussian* are fully tested. Test runs calling ORCA are waiting for the corresponding orca version to be finalized and officially released.

8. A number of scripts have been developed to help program developers and users in updating the QMMM path in test runs, in comparing the new and previously obtained test run results, and in preparing inputs and results of test runs for distribution:

a) `updateqmmmpath` is an UNIX script to update the QMMM path: `qmmmx.y.z` in the *`.ml`* scripts of all test runs and many scripts in the script directory. Here, `x.y.z` indicates the version number, e.g., `1.3.6`. This greatly reduces the time needed in editing.

b) `checktestrun_tinker`, `checktestrun_g03_qm`, `checktestrun_g03_qmmm`, `checktestrun_gms_qm`, `checktestrun_gms_qmmm`, `checktestrun_orca_qm`, and `checktestrun_orca_qmmm` are scripts to help compare the results for the corresponding test runs obtained by the user/developer with the results distributed with the program.

c) `updatetestrun` and `updatetesto` are scripts to help, respectively, make the new directories `newtestrun` and `newtesto`, which contain the latest inputs and results of test runs for distribution.

11.M. Version 1.3.7

Finalized on Aug 7, 2009

Released on Aug 17, 2009

Authors: Hai Lin, Yan Zhang, and Donald G. Truhlar

This is an enhanced version of QMMM version 1.3.6. The most important improvement is the development and implementation of the flexible-boundary RC and RCD schemes. Some bugs are also fixed. More details are as follows:

1. In this version, the version number and address output in the `mlhedr` subroutine and `perl` script `Gau_External` used in the QM/MM external optimization by the *B* or *C* version of `Gaussian` (or script `Gau_External2` by the *D* or *E* version of `Gaussian`) were modified.

2. In the version 1.3.6, when running the QM/MM test, the scripts `test1XX.ml` and `test2XXX.ml` try to delete some files that do not exist in the disk. This would produce

"segment error" for certain types of machines, e.g., IBM Power5 machine running AIX5.2. Thus, the scripts `test1XX.ml` and `test2XXXX.ml` have been revised in this version.

3. In this version, the major changes are the implementation of the flexible-boundary RC (FBRC) and flexible-boundary RCD (FBRCD) schemes.

A. In the file named `flexbound.F`, 1 new subroutine was added: `fbtempcal`. In the flexible-boundary treatment, the electronic temperature parameter can be determined by QMMM automatically as follows.

Assume the equilibrium $X^+ + e^- \leftrightarrow X$ exists.

$$\mu = -I + k_B T \ln[(1 - x) / x] \quad (1)$$

where x is the molar fraction of the oxidized state X^+.

$$q = q(\mathrm{CPS}) = q_0(1 - x) + q_1 x \quad (2)$$

where q_1 is the charge of the oxidized state X^+, and q_0 the reduced state X.

$$d\mu/dq = -k_B T[x(1 - x)]^{-1}(q_1 - q_0)^{-1} \quad (3)$$

Alternatively, if the equilibrium is between X and X^-, $X + e^- \leftrightarrow X^-$

$$\mu = -A + k_B T \ln[x/(1 - x)] \quad (4)$$

where x is the fraction of X^-.

$$q = q(\mathrm{CPS}) = q_0(1 - x) + q_1 x \quad (5)$$

where q_1 is the charge of the reduced state X^- and q_0 the oxidized state X.

$$d\mu/dq = k_B T\,[x(1-x)]^{-1}(q_1 - q_0)^{-1} \quad (6)$$

The two situations can be unified as:

$$d\mu/dq = k_B T\,[x(1-x)]^{-1}(q_{red} - q_{ox})^{-1} \quad (7)$$

$$d^2\mu/dq^2 = [4k_B T/(q_{red} - q_{ox})^2][(1-2x)/x^2(1-x)^2] \quad (8)$$

where q_{red} and q_{ox} are the charges of the reduced and oxidized states, respectively.

at $x = 0.5$, where the molar fraction of the reduced and oxidized states are equal,

$$d\mu/dq = 4k_B T/(q_{red} - q_{ox}) \quad (9)$$

$$d^2\mu/dq^2 = 0$$

This tells us that the $d\mu/dq$ curves are linear at $x = 0.5$.

The QEq method (or other electronegativity equalization classic models) yields a linear line for μ as a function of charge q on the CPS. Therefore, for calibration purpose, we let the slope computed by Equation (9) equal the chemical potential slope calculated by QEq model:

$$d\mu/dq \,\big|_{x=0.5} = d\mu(\mathrm{QEq})/dq \quad (10)$$

The $d\mu(\mathrm{QEq})/dq$ can be computed numerically by

$$d\mu/dq = (\mu_{red,QEq} - \mu_{ox,QEq})/(q_{red} - q_{ox}) \quad (11)$$

where $\mu_{red,QEq}$ and $\mu_{ox,QEq}$ are the chemical potentials calculated by the QEq model for the reduced and oxidized states.

B. The subroutines `fbbpc`, `qmmmehk`, `qmmmghk`, and `qmmmhhk` were revised for the calculations of the QM/MM energy and gradient of the flexible-boundary treatments.

C. Two test runs, test2055 and test2056, have been added for the FBRC treatment.

D.

11.N. Version 1.3.8

Finalized on Sep 24, 2010

Released on Sept 27, 2010

Authors: Hai Lin, Yan Zhang, Soroosh Pezeshki, and Donald G. Truhlar

This is an enhanced version of QMMM. The possibility to run molecular dynamics simulations is added. Dynamics is simulated with Born-Oppenheimer molecular dynamics. The integration of the equations of motion is carried out with the velocity Verlet algorithm. The parameters of the calculation have to be set in the *DYNAMICS section. Rattle constraints can be used for covalent bonds to hydrogen. The temperature of the system can be controlled through the Berendsen thermostat. Temperature annealing can be used for global geometric optimization. Each dynamics simulation produces a trajectory file and a restart file for restarting aborted jobs.

A. In the file `dynamics.F`, 23 new subroutines are added: `boxsize`, `calccom`, `dout`, `dout0`, `doutt`, `findmolecules`, `gaussrandom`, `initmd`, `mldhook`, `moloutbox`, `progmm0ghkmd`, `progmmghkmd`, `random`, `rattleinit`, `rattleposition`, `rattlevelocity`, `readrestart`, `rdynamics`, `runmd`, `shiftmol`, `t41inpplus`, `t41inpxyz`, `tempannealing`, `tempcontrol`, and `writerestart`.

B. The modules `caltyp`, `files`, and `input` as well as the subroutines `bgchscale`, `bgchshift`, `insumry`, `main`, `mlghook`, `mlhedr`, `qmmmbgchgeom`, `qmmmghk`,

qmmmssgeom, progghk, read5, and t41outg were revised for the calculation of dynamics.

X. The testruns Test105, test210, test2057, test308 and test3031 have been added.

Δ. A new directory tool/mdanalysis is added, where two F95 programs extractER.f and averageT.f are provided. Those two programs are not necessarily for running QMMM, but they can be useful in analyzing the results produced in MD runs.

11.O. Version 1.4.0

Finalized on Nov 24, 2011

Released on Dec 13, 2011

Authors: Hai Lin, Yan Zhang, Soroosh Pezeshki, and Donald G. Truhlar

This is an enhanced version of QMMM. The first new feature of this version is the adaptive partioining (AP) QM/MM. The Hotspot, ONIOM-XS, sorted AP, and permuted AP methods are implemented. Fragmental groups, i.e., groups that are fragments of molecules can be treated in AP QM/MM. The program automatically determines the link atoms and the charge of the QM subsystem on the fly.

A. In the new file ap.F, 26 subroutines are added: partition, litest, setgroupaffiliation, calcgroupcom, quicksort, addcapatoms, readgroups, readzeroenergy, hotspot, smoothingfunction, fifthorderspline, oniomxs, sortedap, dpiphi, sortedsmoothing, permutedap, permutedcombination, binomial, permutedgradient, permutedsmoothing, permutedsmoothing2, calculateArgonShift, mmargontest, resetqmmm, copyqminp, and calculatezeroenergy.

   In the file dynamics.F, 15 new subroutines are added: readdyncap, readrestraints, readrestraintfile, restraintsline, restraints, instantmove, shiftsmd, readinstantmove, progmmghkmd, progmm0ghkmd, qmmmghkmd, readpath, settsemiemperical, g03stinpesp, and tempcontrolnh

B. The modules caltyp, files, and input as well as the subroutines qmmmghk, orcastinp, orcainp, rtest. t41inp, g03inp, cflot, rqmmm, rtestmm,

rmglen, restopt, rmlopt, rmgeom, and proggsthk were revised for MD calculations.

X. 9 test runs have been added.

Δ. The two F95 programs extractER.f and averageT.f are replaced by the perl script extract.pl. A new F95 program createGroups.F is added to create a group list. A shell script univeral_run is added, which can be used to copy all necessary files to the working directory and start the simulation.

11.P. Version 2015

Finalized on March 31, 2015

Released on April 3, 2015

Authors of changes in this revision: Bo Wang and Donald G. Truhlar

Authors of this version of code: Hai Lin, Yan Zhang, Soroosh Pezeshki, Bo Wang, and Donald G. Truhlar

Version 2015 is an enhanced version of QMMM version 1.4.0. The most important improvements are the development and implementation of the balanced RC, balanced RC2, and balanced RCD schemes, the implementation of the screened charge scheme and smeared charge scheme for QM–MM interactions in QM/MM calculations, and the implementation of the fixed-bond-distance scheme to define the link atom. Tuned F atoms can be used as the link atoms in simulations. More details are as follows:

1. In this version, one major change is the implementation of the balanced RC (BRC) and balanced RC2 (BRC2) schemes.
   A. The subroutines qmmmssgeom, bgchshift, qmmmbgchgeom, g03stinp, qmmmgsum, qmmmhsum were revised for the calculations of the QM/MM energy and gradients in the balanced RC scheme and other balanced schemes, including balanced SEE, balanced RC2, Amber-1, balanced RC3, Amber-2, balanced RCD, and balanced Shift. All these schemes are called balanced schemes in the following descriptions.

   qmmmssgeom: Different types of atoms (H, F, Cl, C) can be used as the link atom.

   bgchshift: The balanced schemes are implemented.

`qmmmbgchgeom`: The balanced schemes are implemented.

`g03stinp`: The number of digits for the charges in the *Gaussian* input file is increased.

`qmmmgsum`: The formula to sum up the gradients for the balanced schemes is added.

B. Test runs test2061, test2062, test2063, and test2064 have been added for the BRC, BRC2, TBRC, and TBRC2 schemes.

2. The screened charge scheme and smeared charge shemes are implemented in version 2015.
   A. Files `echarge.f, echarge1.f and echarge3.f, initial.f, kcharge.f` in TINKER were modified to allow the calculation of a correction term to QM–MM electrostatic interactions of the QM subsystem with the MM subsystem using the screened charge model and smeared charge model. In the current implementation of the screened charge and smeared charge models in QMMM program, the screened or smeared MM charges only interact with the QM electrons in an embedding QM calculation from an electronic structure package. TINKER will calculate the interactions between the screened or smeared MM charges and the QM nuclei, and add this correction to the total energy.
   B. Subroutines `t41keyadd1, g03stinp1` are newly added to implement the screened charge scheme for the interaction energy.

      `t41keyadd1`: This subroutine is based on `t41keyadd` to add keywords to t41.key to calculate a correction term to QM–MM electrostatic interaction energy in the screened charge scheme.

      `g03stinp1`: This subroutine is based on `g03stinp` to write Gaussian input file for the calculation of the primary system in the presence of background screened charges using the screened charge scheme.
   C. Test run test2065 has been added for the screened charge scheme.

3. In this version, we implement the fix-bond-distance scheme to define the position of the link atom. Analytic gradients have been added. Analytic Hessians are postponed to a later version.
   A. The subroutines `qmmmgsum` was revised.

      `qmmmgsum`: The formula to sum up the gradients for fix-bond-distance scheme is added.

4. In this version, versions 5.1 and 6.3 of the TINKER program are modified to do screened and smeared charge calculations. The modified versions are in qmmm2015/tinker_QMMM/ folder. The following keywords are added in the modified TINKER program.

LAMBDA VARIABLE *1.0*

The LAMBDA keyword is used to specify the smearing width of the redistributed charges.

NGTO1 VARIABLE *3*

The NGTO1 keyword is used to specify the number of Gaussian functions to fit a Slater-type function for the screened charges.

NGTO2 VARIABLE *6*

The NGTO2 keyword is used to specify the number of Gaussian functions to fit a Slater-type function for the smeared redistributed charges.

DAMPRC VARIABLE *0*

The DAMPRC keyword is used to specify whether the redistributed charges are smeared.

0: redistributed charges are represented by point charges

1: redistributed charges are represented by smeared charges, with the smearing width defined by LAMBDA.

ODS switch no ODS

This keyword turns on the usage of the outer density screening (ODS) model with the ODS parameters to calculate the correction term to QM–MM interactions used in the screened QM/MM method in the QMMM program.

DAMPING [1 integer]

This keyword set provides the atoms that need to be screened in the ODS model. The number is the atom number of the screened atom in the coordinate file. The program assigns the zeta value and number of screened electrons from Table 4.F.1 and eq. 4.F.3. Currently only the elements in Table 4.F.1 have parameters. For all other elements, the point charge model is used even if that atom number is specified here.

DAMPING3 [1 integer & 2 reals]

This keyword set provides an option to change the ζ value and the number of screened electrons in the screening region of the screened atom. The first number is the atom number of the screened atom in the coordinate file, the second number is the ζ value, the third number is the

number of valence electrons; the number of electrons in the screening region n_{screen} will be the third number minus *q*, where *q* is the partial atomic charge on the atom.

DAMPING4 [1 integer & 3 reals]

This keyword set provides another possible way to change the ζ value and the number of screened electrons in the screening region of the screened atom. The first number is the atom number of the screened atom in the coordinate file, the second number is the ζ value, the third number is the number of valence electrons, and the last number is the extra charge included in the screening region. The number of electrons in the screening region n_{screen} will be the third number minus the fourth number. If one wants to turn off the screening of a screened atom, use 0.0 and 0.0 for the third and fourth numbers (no electrons in the screening region).

11.Q. Version 2017

Finalized on October 20, 2017

Released on October 22, 2017

Authors of changes in this revision: Xin-Ping Wu, Laura Gagliardi, and Donald G. Truhlar

Authors of this version of code: Hai Lin, Yan Zhang, Soroosh Pezeshki, Bo Wang, Xin-Ping Wu, Laura Gagliardi, and Donald G. Truhlar

This is an enhanced version of *QMMM* version 2015. The major changes are:

- Modifications were made to make the program to be compatible with *Gaussian 16* as well as *Gaussian 09*.
- A bug in the Amber-2 scheme was fixed.
- The Amber-2 scheme can now work for cases including multiple charge balancing groups.
- The maximum connectivity of an atom is increased to 8.

More details are as follows:

1. The *QMMM* program writes the "`fchk`" keyword in the *Gaussian* input file to generate a *Gaussian* formatted checkpoint file which includes energy, Cartesian coordinates, gradient, and Hessian that will be further read by the *QMMM* program. However, in *Gaussian 16*, the use of "`fchk`" keyword in *Gaussian* input file cannot generate a *Gaussian* formatted checkpoint file including energy, gradient, and Hessian. To make the *QMMM* program compatible with *Gaussian 16*, the "`%chk=Test.chk`" keyword is used instead of the "`fchk`"

keyword in the *Gaussian* input file; in addition to that, the `g03shuttle` scripts are modified to be able to convert a *Gaussian* checkpoint file to a *Gaussian* formatted checkpoint file. The *Gaussian* formatted checkpoint file generated through such a way includes all the necessary information (i.e., energy, Cartesian coordinates, gradient, and Hessian).

2. The `g03partouto` subroutine was modified such that the program can read the energy and Cartesian coordinates in the *Gaussian* formatted checkpoint file during partial optimization using the *Gaussian 09* and *Gaussian 16* optimizers. Note that the energy and Cartesian coordinates appear in different orders on the *Gaussian 03*, *Gaussian 09*, and *Gaussian 16* formatted checkpoint files.
3. A bug in the Amber-2 charge modification scheme in the `bgchshift` subroutine was fixed: There was a bug in the charge balancing step when using the Amber-2 scheme; for the Amber-2 scheme, the modified M1 charges should be redistributed evenly to all MM atoms except M1 atoms. However, by mistake, the M1 atoms were not excluded in calculating the redistributed charges.
4. The Amber-2 charge modification scheme was implemented in the previous version 2015. In *QMMM 2015*, all the charge modification schemes except Amber-2 can deal with cases including multiple charge balancing groups; in *QMMM 2017*, Amber-2 is able to deal with such cases; in version 2017, the Amber-2 scheme in the `bgchshift` subroutine was modified to work for such cases. This is helpful when users want to use the Amber-2 scheme and there are multiple subsystems in the MM subsystem.
5. In the previous versions, the maximum connectivity of one atom, as specified in the `.crd` file was 6. This is increased to 8 in version 2017 to allow calculations on complex systems where atoms have high coordination numbers.
6. Two new test runs (test2066 and test2067) have been added to illustrate the application of QM/MM calculations to the metal-organic framework NU-1000. The test2067 also illustrates the Amber-2 scheme with multiple charge balancing groups.
7. The `NU1T` force field is provided in the directories of test2066 and test2067 to allow QM/MM calculations on NU-1000.
8. One new test run (test2068) has been added to show that version 2017 is compatible with *Gaussian 16*.

11.R. Version 2018

Finalized on September 19, 2018

Released on October 19, 2018

Authors of changes in this revision: Xin-Ping Wu, Laura Gagliardi, and Donald G. Truhlar

Authors of this version of code: Hai Lin, Yan Zhang, Soroosh Pezeshki, Bo Wang, Xin-Ping Wu, Laura Gagliardi, and Donald G. Truhlar

This is an enhanced version of *QMMM* version 2017. The major changes are:

- Modifications were made to make the balanced charge modification schemes (BRC, BRCD, BRC2, BRC3, and Amber-2) work for cases where the whole MM region includes multiple M1 atoms and is considered as one charge balancing group.
- The *Gaussian* include file mechanism to specify the basis sets for *QMMM* calculations using the *Gaussian* external feature is supported.
- An error message will be written to the *QMMM* output file and the *QMMM* calculation will be terminated when *Gaussian* energy and gradient calculations fail during QM/MM optimization.
- The `Gau_ext.acc` script is provided, and the `ex_shuttle`, `g03shuttle`, and `.ml` scripts were modified to allow both QM/MM single-point calculations and optimizations using the *Gaussian* external optimizer to read necessary information from the *Gaussian* checkpoint file to accelerate the QM/MM calculation.

More details are as follows:

1. In versions 2015 and 2017, for the balanced charge modification schemes (BRC, BRCD, BRC2, BRC3, and Amber-2), only one M1 atom is allowed in a charge balancing group. If there are multiple M1 atoms in a charge balancing group, the program will just “see” the M1 atom that appears first in the input file, and this will lead to the incorrect redistribution of modified M1 charges. The `bgchshift` subroutine was modified to make the balanced charge modification schemes work for cases where the MM region includes multiple M1 atoms and is considered as one charge balancing group.
2. This version supports using the *Gaussian* include file mechanism to specify the basis sets for *QMMM* calculations using the *Gaussian* external program. (Note that the name and the path of the basis set file have to be lowercase.) This capability is implemented for convenience. For example, if a set of QM/MM calculations use the GEN option or the GENECP option for

basis sets or ECPs, it is convenient to store the basis set and/or ECP information in a separate file and just use the @ include mechanism in the `.dat` input file.

3. The maximum number of atoms in a group that is involved in any group-based treatment (such as charge balancing) is increased from 300 to 9999. This is necessary when the QM/MM system is large.
4. The maximum number of atoms for a Hessian calculation is changed from 500 to 800.
5. The character length mismatch in the variable 'upcse' is corrected.
6. In old versions, when the *Gaussian* energy and gradient calculations during QM/MM optimization with the *Gaussian* external optimizer fails, the *QMMM* program will ignore such failure and feed the previously obtained gradients to the *Gaussian* external optimizer to perform geometry optimization, which will always lead to geometry distortion. Such a problem can be difficult to detect. In version 2018, the *QMMM* program can detect the failure of a *Gaussian* energy and gradient calculation. When the calculation fails, an error message will be written into the *QMMM* output file, and the *QMMM* calculation will be terminated immediately.
7. Beginning with version 1.3.4 and continuing up to version 2015, the *QMMM* program could allow both single-point calculations and optimizations using the *Gaussian* external optimizer to read necessary information from the *Gaussian* checkpoint file to accelerate the calculations. To do this, users have to manually: (1) do a QM/MM single-point calculation using the *Gaussian* external program; (2) copy the *Gaussian* checkpoint file to the job directory where the input files locate; (3) rename the *Gaussian* checkpoint file to `name.chk` (`name` is the variable in the `.ml` script); (4) add the "`%chk=g03.chk`" and "`guess=read`" keywords in the "`OPTIONS`" of the "`QMKEY`" section in the `.dat` input file; (5) do a QM/MM optimization using the *Gaussian* external optimizer. However, this capability was lost in version 2017 because – to make the *QMMM* program compatible with *Gaussian 16* – the *QMMM* program was changed to write the "`%chk=Test.chk`" keyword, which conficts with the "`%chk=g03.chk`" keyword, in *Gaussian* input files to generate *Gaussian* checkpoint files, and then the generated *Gaussian* checkpoint files will be converted to *Gaussian* formatted checkpoint files which contain necessary information to be readed by the *QMMM* program. In version 2018, the lost capability is restored, and *QMMM* becomes more user-friendly by providing the `Gau_ext.acc` script and modifying the `ex_shuttle`,

`g03shuttle`, and `.ml` scripts. The newly provided `Gau_ext.acc` script (in the `script` directory) automatically does the operations that used to have to be done manually by users. Specifically, the `Gau_ext.acc` script: (1) generates a `.dat` input file for the single-point calculation based on a normal `.dat` input file for QM/MM optimization provided by the user (an error message will be written into the `Gau_ext.acc.err` file if the `.dat` input file provided by the user is problematic); (2) calls the `.ml` script to do a single-point calculation; (3) checks if the single-point calculation is finished normally when the calculation is terminated (if the single-point calculation fails, an error message and suggestions will be written into both the `Gau_ext.acc.err` file and the *QMMM* output file); (4) copies the correct *Gaussian* checkpoint file to the job directory where the input files locate and renames it to `guess.chk` (non-expert users who are not familiar with the flow process of the *QMMM* program may be confused of copying which *Gaussian* checkpoint file since multiple *Gaussian* checkpoint files may be generated); (5) generates a `.dat` input file for the optimization based on a normal `.dat` input file for QM/MM optimization provided by the user (an error message will be written into the `Gau_ext.acc.err` file if the `.dat` input file provided by the user is problematic); (6) calls the `.ml` script to do a QM/MM optimization. Therefore, the `Gau_ext.acc` script in stead of the `.ml` script should be executed in order to accelerate the QM/MM optimization using the *Gaussian* external optimizer. The newly modified `ex_shuttle`, `g03shuttle`, and `.ml` scripts work with the `Gau_ext.acc` script to store and move the *Gaussian* checkpoint files. (Note that only the `Gau_ext.acc` and `.ml` scripts need to be put in the job directory.) It is recommended to execute the `Gau_ext.acc` script instead of the `.ml` script to perform QM/MM optimizations using the *Gaussian* external optimizer for two reasons: (1) the efficiency of optimizations can be greatly improved (according to our tests, the calculation can be at least two times faster than a normal calculation); (2) the failures of *Gaussian* energy and gradient calculations during QM/MM optimization using the *Gaussian* external optimizer, which often happen for QM/MM optimization on a complicated system such as MOF and will always lead to geometry distortion, can be avoided.

8. One new test run (test2069) has been added to show the capabilities 1 and 2.
9. One new test run (test2070) has been added to show the capability 7.

Chapter Twelve

12

12. Acknowledgments

We thank Chris Cramer, Jiali Gao, Andreas Heyden, Casey Kelly, Benjamin Lynch, Keiji Morokuma, Jingzhi Pu, Christopher Riplinger, Walter Thiel, Jason Thompson, and Yan Zhao for helpful discussions. This work was supported in part by the U.S. Department of Energy (Office of Basic Energy Sciences), the National Science Foundation, the Office of Naval Research, and Research Cooperation. Hai Lin thanks the Minnesota Supercomputing Institute for a Research Scholarship (2004 – 2005).

Chapter Thirteen

13

13. Index

13.1. List of Keywords

13.2. Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

Z

==========End of Manual ==========