Supporting information for: An extensible interface for QM/MM molecular dynamics simulations with AMBER

Andreas W. Götz,[*,†] Matthew A. Clark,[†] and Ross C. Walker[*,†,‡]

San Diego Supercomputer Center, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0505, USA, and Department of Chemistry and Biochemistry, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0505, USA

E-mail: agoetz@sdsc.edu; ross@rosswalker.co.uk

1 Additional details of the AMBER implementation

The interface to electronic structure software has been integrated into the QM/MM code of the MD engine SANDER and made available with release version 12 of the AMBER[1–3] software package for biomolecular simulations. For integration with SANDER, the existing QM/MM code has been refactored such that the QM/MM driver subroutine obtains the QM contribution to the energy and forces from a single call to a subroutine that initializes the built-in semiempirical code. A new driver routine has been written in Fortran 90 that is called instead of the semiempirical code if an external electronic structure program is used for a QM/MM calculation, as specified by setting the `qm_theory` variable of the `qmmm` namelist in the AMBER input file `mdin` to `'EXTERN'`.

[*]To whom correspondence should be addressed
[†]San Diego Supercomputer Center
[‡]UCSD Department of Chemistry and Biochemistry

The driver subroutine for the new QM/MM interface passes/retrieves all data to/from the interface driver subroutines, as required for each supported electronic structure program (see Section 2). In case of parallel runs with multiple replicas, such as replica exchange molecular dynamics (REMD) or path integral molecular dynamics (PIMD) simulations, the driver subroutine determines the ID of the replica for each of the parallel threads.

2 Interface to electronic structure software

The interface consists of one Fortran 90 module for each of the supported electronic structure programs. At the time of release of version 12 of AMBER, the following electronic structure programs are supported for mechanical embedding:

- ADF[4–6] (tested versions: 2009 to 2012)
- GAMESS[7,8] (tested version: release October 1, 2010 R1)
- NWChem[9] (tested version: 6.1)

and the following programs are supported for mechanical and electronic embedding:

- Gaussian[10] (tested versions: g03 and g09)
- Orca[11] (tested versions: 2.7.0 to 2.9.1)
- TeraChem[12,13] (tested versions: 1.5)

Utility routines that are common to all electronic structure programs are collected in a separate utility module.

2.1 Data communication with electronic structure software

Two different communication protocols are implemented for data exchange between the interface and the QM programs. Communication via files and system calls is implemented for all supported

software packages, while the MPI-2 based client/server model, at the time of writing, is only supported by TeraChem.

Communication via files and system calls The standard mode of data communication between the interface and the supported electronic structure programs proceeds via data files and system calls for the execution of the electronic structure program. In this case the interface proceeds by

1. writing input files for the QM program that contain the current QM region atom types and coordinates and the MM point charges and coordinates,

2. executing the QM program via a system call, and

3. parsing the output files of the QM program to retrieve the energy and forces and, if requested, the dipole moment and atomic partial charges of the QM region atoms

If requested by the user the interface will store the dipole moment and atomic partial charges of the QM region along an MD trajectory. The interface stores the in- and output files of the QM calculation for both the present and the last call which simplifies debugging in case of program crashes.

MPI-2 based client/server model A client/server model for data exchange based on version 2 of the message passing interface standard[14] (MPI-2) is also implemented. Since this requires corresponding changes to the electronic structure software, support is currently provided only by TeraChem.[15] In this case, the electronic structure code has to be started in server mode at the beginning of a simulation. The interface then connects as a client and all subsequent data exchange occurs via standard MPI calls as described below in section Section 2.3.

2.2 Fortran API

The Fortran 90 module for each supported electronic structure program exposes a driver subroutine for exchange of relevant data with the MD program: `get_adf_forces`

(for ADF), `get_gms_forces` (for GAMESS), `get_nw_forces` (for NWChem), `get_gau_forces` (for Gaussian), `get_orc_forces` (for Orca), `get_tc_forces` (for TeraChem), `get_genmpi_forces` (generic interface for MPI-2 client/server model). The calling convention for modules with restriction to mechanical embedding (ADF, GAMESS, NWChem) is (exemplified for ADF):

```
call get_adf_forces(do_gradient, nstep, ntpr, id,
            & nqm, qmcoords, qmtypes, energy, dxyzqm, qmcharge, qmspin)
```

The calling convention for modules that support electronic embedding (Gaussian, Orca, TeraChem, generic MPI-2) is (exemplified for Gaussian):

```
call get_gau_forces(do_gradient, nstep, ntpr, id,
            & nqm, qmcoords, qmtypes, ncl, clcoords,
            & energy, dxyzqm, dxyzcl, qmcharge, qmspin)
```

Data passed to the driver subroutines (*intent in*):

`do_gradient` *logical*, `.true.` if both energy and forces shall be calculated.

`nstep` *integer*, MD step number.

`ntpr` *integer*, frequency of printing (is used if dipole moment or atomic partial charges are requested to be otuput).

`id` *character(len=3)*, ID of the current thread. The QM calculation will be executed in a subdirectory with corresponding in- and output file names. Useful to run multiple QM calculations in parallel as for example in REMD and PIMD simulations.

`nqm` *integer*, number of atoms in the QM region, including link atoms.

`qmcoords` *double precision, dimension(3,nqm)*, Cartesian coordinates of atoms in the QM region in Angstrom.

qmtypes *integer, dimension(nqm)*, atom types of atoms in the QM region (nuclear charge in atomic units).

ncl *integer*, number of atoms in the MM region that are included as point charges for electronic embedding. Only for modules supporting electronic embedding.

clcoords *double precision, dimension(4,ncl)*, Cartesian coordinates (Angstrom) and charges q (atomic units) of point charges in order (x, y, z, q). Only for modules supporting electronic embedding.

qmcharge *integer*, charge of the QM region.

qmspin *integer*, spin multiplicity of the QM region.

Data returned from the driver routines (*intent out*):

energy *double precision*, QM contribution to the QM/MM energy in kcal/mol.

dxyzqm *double precision, dimension(3,nqm)*, QM contribution to the force acting on atoms in the QM region, including link atoms, in kcal/(mol*Å).

dxyzcl *double precision, dimension(3,ncl)*, QM contribution to the force acting on point charge atoms in the MM region that are included in the electronic QM/MM Hamiltonian, in kcal/(mol*Å). Only for modules supporting electronic embedding.

Control data for the QM runs (such as QM method or whether to use a template input file) is obtained from a Fortran namelist corresponding to each supported electronic structure program as described in the AMBER manual. The namelist is read from a formatted text file that is expected to be connected to Fortran unit 5 (as is the AMBER input file `mdin` in the AMBER implementation).

2.3 API for MPI-2 based client/server model

To perform a QM/MM MD simulation using the MPI interface, an MPI version of the electronic structure program has to be launched in server mode before the interface that is trying to connect to

it, that is, in the case of AMBER before the MD program SANDER is executed. An MPI version of the MD program can then be started (SANDER in the case of the AMBER implementation) in which the interface is executed in client mode. For TeraChem,[15] this can be controlled by setting the Fortran namelist variable `mpi` in its input namelist to 1, while all other control data for the QM runs (such as QM method or whether to use a template input file) is obtained in the same fashion from the Fortran namelist as for data exchange using files and system calls. The interface also implements a generic version of the MPI-2 client/server model for data exchange that is not specific to TeraChem and which can be used to communicate with other electronic structure software. The API is described below.

The QM program needs to open an MPI port and publish its name for connection by the interface. Upon execution in client mode, the interface uses `MPI_LOOKUP_NAME` to look for a port name that has been published under the service name `qc_program_port`. In the case of simulations with multiple replicas, such as PIMD and REMD in AMBER, an ID is appended, that is, the corresponding thread will look for published service names `qc_program_port.N`, where `N` runs from 1 to number of replicas.

```
MPI_LOOKUP_NAME(servicename, MPI_INFO_NULL, portname)
```

Next, the interface connects as a client to the published port and establishes a new MPI communicator via `MPI_COMM_CONNECT` that is used for all subsequent data exchange, which proceeds via standard MPI send and receive calls.

```
MPI_COMM_CONNECT(portname, MPI_INFO_NULL, 0,
                 MPI_COMM_SELF, newcomm)
```

Next, the interface sends all settings for the QM program using `MPI_SEND` to rank 0 with tag 1. The interface has obtained these settings either from the input file via the corresponding namelist or uses default settings as described below or has read them as character strings from a template input file (limited to 256 characters per line with a maximum of 128 lines). These settings are sent only during the first call of the interface.

```
MPI_SEND(settings, 128*256, MPI_CHARACTER, 0, 1, newcomm)
```

The advantage of sending the program settings as a character array is that no assumption about the data type (character, integer number, floating point number) transmitted has to be made. The electronic structure software can parse it as if it were a regular input file. Currently, the following keyword/value pairs are sent by default, each occupying 256 characters with trailing blanks:

- 'method BLYP'
- 'basis 6-31G*'
- 'jbasis none'
- 'cbasis none'
- 'scfconv 1E-08'
- 'scfiter 100'
- 'guess read'
- 'gradient true'
- 'grid none'

All subsequent data exchange with the electronic structure program takes place during the first and all subsequent calls of the interface. The interface sends the following data in order, using

```
MPI_SEND(data, size, MPI_DATATYPE, 0, 1, newcomm)
```

qmcharge *size=1, MPI_INTEGER*, charge of the QM region

qmspin *size=1, MPI_INTEGER*, spin multiplicity of the QM region

nqm *size=1, MPI_INTEGER*, number of QM atoms including link atoms

qmtypes *size=2*nqm, MPI_CHARACTER*, element names of QM atoms

qmcoords *size=3*nqm, MPI_DOUBLE_PRECISION*, Cartesian coordinates of QM atoms (Ångstrom)

ncl *size=1, MPI_INTEGER*, number of point charges

clcharges *size=ncl, MPI_DOUBLE_PRECISION*, point charge values (atomic units)

clcoords *size=3*ncl, MPI_DOUBLE_PRECISION*, Cartesian coordinates of point charges (Ångstrom)

The number of QM atoms and the charge and spin multiplicity of the QM region are sent during each MD step to allow for future implementations of flexible QM/MM approaches with changing QM regions.

The interface then receives the following data, in order, using

```
MPI_RECV(data, size, MPI_DATA_TYPE, MPI_ANY_SOURCE,
         MPI_ANY_TAG, newcomm, status)
```

energy *size=1, MPI_DOUBLE_PRECISION*, QM contribution to the QM/MM energy (atomic units)

charges *size=nqm, MPI_DOUBLE_PRECISION*, atomic partial charges from population analysis (atomic units)

dipole *size=4, MPI_DOUBLE_PRECISION*, QM dipole moment (x, y, z, total; atomic units)

dxyzqm *size=3*nqm, MPI_DOUBLE_PRECISION*, QM contribution to the force acting on atoms in the QM region, including link atoms (atomic units)

dxyzcl *size=3*ncl, MPI_DOUBLE_PRECISION*, QM contribution to the force acting on point charge atoms in the MM region that are included in the electronic QM/MM Hamiltonian (atomic units)

Upon its last call, the interface invokes an MPI send command with a 0 value tag (instead of sending the QM charge as laid out above) that should be interpreted by the electronic structure code to disconnect and quit:

```
MPI_SEND(empty, 1, MPI_DOUBLE_PRECISION, 0, 0, newcomm)
```

References

(1) Case, D. A.; Cheatham III, T. E.; Darden, T.; Gohlke, H.; Luo, R.; Jr., K. M. M.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J. *J. Comput. Chem.* **2005**, *26*, 1668–1688.

(2) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. *WIREs Comput. Mol. Sci.* **2012**, in press. DOI: 10.1002/wcms.1121.

(3) Case, D. A.; Darden, T. A.; Cheatham, T. E., III; Simmerling, C. L.; Wang, J.; Duke, R. E.; Luo, R.; Walker, R. C.; Zhang, W.; Merz, K. M.; Roberts, B. P.; Hayik, S.; Roitberg, A.; Seabra, G.; Swails, J.; Götz, A. W.; Kolossváry, I.; Wong, K. F.; Paesani, F.; Vanicek, J.; Wolf, R. M.; Liu, J.; Wu, X.; Brozell, S. R.; Steinbrecher, T.; Gohlke, H.; Cai, Q.; Ye, X.; Wang, J.; Hsieh, M.-J.; Cui, G.; Roe, D. R.; Mathews, D. H.; Seetin, M. G.; Sagui, C.; Babin, V.; Luchko, T.; Gusarov, S.; Kovalenko, A.; Kollman, P. A. *AMBER 12*, University of California, San Francisco, 2012.

(4) te Velde, G.; Bickelhaupt, F. M.; Baerends, E. J.; Guerra, C. F.; van Gisbergen, S. J. A.; Snijders, J. G.; Ziegler, T. *J. Comput. Chem.* **2001**, *22*, 931–967.

(5) Guerra, C. F.; Snijders, J.; te Velde, G.; Baerends, E. J. *Theoret. Chem. Acc.* **1998**, *99*, 391–403.

(6) *ADF2012, SCM, Theoretical Chemistry, Vrije Universiteit, Amsterdam, The Netherlands, http://www.scm.com*.

(7) 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.; Windus, T. L.; Dupuis, M.; John A. Montgomery, Jr, *J. Comput. Chem.* **1993**, *14*, 1347–1363.

(8) Gordon, M. S.; Schmidt, M. W. Advances in in electronic structure theory: GAMESS a decade later. In *Theory and Applications of Computational Chemistry, the first forty years*; Dykstra, C. E., Frenking, G., Kim, K. S., Scuseria, G. E., Eds.; Elsevier: Asterdam, 2005; Chapter 41, pp 1167–1189.

(9) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Dam, H. J. J. V.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L.; de Jong, W. A. *Comp. Phys. Comm.* **2010**, *181*, 1477–1489.

(10) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Mennucci, B.; Petersson, G. A.; Nakatsuji, H.; Caricato, M.; Li, X.; Hratchian, H. P.; Izmaylov, A. F.; Bloino, J.; Zheng, G.; Sonnenberg, J. L.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Montgomery, J. A., Jr.; Peralta, J. E.; Ogliaro, F.; Bearpark, M.; Heyd, J. J.; Brothers, E.; Kudin, K. N.; Staroverov, V. N.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Rega, N.; Millam, J. M.; Klene, M.; Knox, J. E.; Cross, J. B.; Bakken, V.; Adamo, C.; Jaramillo, J.; Gomperts, R.; Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Zakrzewski, V. G.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Dapprich, S.; Daniels, A. D.; Farkas, O.; Foresman, J. B.; Ortiz, J. V.; Cioslowski, J.; Fox, D. J. *Gaussian 09, Revision C.01*, Gaussian, Inc., Wallingford CT, 2010.

(11) Neese, F. *WIREs Comput. Mol. Sci.* **2012**, *2*, 73–78.

(12) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 1004–1015.

(13) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 2619–2628.

(14) Message Passing Interface Forum, *MPI: A Message-passing Interface Standard, Version 2.2*; High-Performance Computing Center Stuttgart: University of Stuttgart, Nobeliusstr. 19, 70550 Stuttgart, Germany, 2009.

(15) Isborn, C.; Götz, A. W.; Clark, M. A.; Walker, R. C.; Martínez, T. M. *J. Chem. Theory Comput.* **2012**, *8*, 5092–5106.