Journal of Chemical Theory and Computation

The pDynamo Program for Molecular Simulations using Hybrid Quantum Chemical and Molecular Mechanical Potentials

Martin J. Field*

Laboratoire de Dynamique Moléculaire Institut de Biologie Structurale — Jean-Pierre Ebel (CEA/CNRS/UJF — UMR 9075), 41 Rue Jules Horowitz, F - 38027 Grenoble, Cedex 01, France

Received March 19, 2008

Abstract: The pDynamo program has been developed for the simulation of molecular systems using hybrid quantum chemical (QC) and molecular mechanical (MM) potentials. pDynamo is written in a mixture of the computer languages Python and C and is a successor to the previous version of Dynamo, now denoted fDynamo, that was written in Fortran 90 (*J. Comput. Chem.* **2000**, *21*, 1088). The current version of Dynamo has a similar range of functionality to the older one but extends it in some significant ways, including the addition of a density functional theory QC capability. This paper gives a general description of pDynamo and outlines some of the advantages and disadvantages that have been encountered in switching computer languages. Some technical aspects of the implementation of pDynamo's algorithms are also discussed and illustrated with the results of example calculations. pDynamo is available on the Web at the address http://www.pdynamo.org and is released under the CeCILL license which is equivalent to the GNU general public license but conforms to the principles of French law.

1. Introduction

Hybrid quantum chemical (QC) and molecular mechanical (MM) potentials, which were first introduced by Warshel and Levitt in 1976[1] in their study of the catalytic mechanism of the enzyme, lysozyme, have become standard tools for the investigation of reaction processes in condensed phase systems. As witness to their utility are the increasing number of modeling programs, both academic and commercial, that implement hybrid potentials and the large number of papers published that employ them (see refs 2 and 3 for recent reviews).

In spite, or because, of this success, hybrid potential techniques are the target of intense research which aims at improving their precision and speed as well as the range of problems to which they can be applied. Hybrid potentials also represent a well-studied and relatively well-understood example of a multiscale approach, in which methods that operate at different scales—in this case electrons and atoms—are coupled together and which are gaining in importance in many areas of computational science.

For many years, our laboratory has been using hybrid potentials to study the mechanisms of enzyme reactions. All our work in this area is done with a program, Dynamo, that we have written and which incorporates all the hybrid potential methods that we have developed and with which all our hybrid potential simulations are performed. Our policy has been to make Dynamo freely available, with source code, in the hope that it could prove useful to the wider scientific community.

The aim of this paper is to introduce a new version of our program, called pDynamo, that is written in Python[4] and C, and which succeeds the previous version, called fDynamo, that was written in Fortran 90.[5] The outline of this paper is as follows. Section 2 gives a general description of the pDynamo program, section 3 discusses, with examples, certain aspects of its implementation, and section 4 summarizes.

* Corresponding author phone: (33)-4-38-78-95-94; fax: (33)-4-38-78-54-94; e-mail: martin.field@ibs.fr.

2. The pDynamo Program

A recent monograph[6] that introduces some of the concepts of molecular simulation employs pDynamo for its example programs and explains many of its features. To avoid duplicating what was said there, we shall restrict ourselves in this paper to a brief overview of the pDynamo program and shall present some of the more technical aspects of the algorithms it implements. Interested readers seeking more should refer to ref 6.

The current version of pDynamo may be downloaded from the pDynamo Web site[7] and is released under the CeCILL license.[8] This is similar in many respects to the GNU general public license[9] but has been drafted to conform to the principles of French law.

2.1. Capabilities of the Program. pDynamo reproduces and extends the functionality of its predecessor fDynamo. Both programs permit the following general classes of calculation or operation:

• Calculation of the potential energy of a system using QC, MM, and hybrid QC/MM potentials. fDynamo employed the all-atom OPLS force field[10] as its MM potential and the semiempirical AM1, MNDO, and PM3[11–13] methods as its QC potentials.

• Geometry optimization of a system to locate both minima and saddle points.

• Reaction path determinations using a variety of methods, including the nudged-elastic-band method.[14]

•Normal mode analysis.

• Classical molecular dynamics simulations for aperiodic (vacuum) systems or for periodic systems in a variety of thermodynamical ensembles.

• Monte Carlo calculations in the canonical and isobaric−isothermal ensembles but limited to simulations of rigid molecules using MM potentials.

• Miscellaneous tools that enable the imposition of constraints, the manipulation of coordinates, and the analysis of simulation data.

In addition, pDynamo introduces some new features, including

• An easier and more powerful interface thanks to the use of the Python computer language.[4]

• A density functional theory (DFT) QC method which can be used alone or as part of a QC/MM hybrid potential.

• An increased range of semiempirical methods, including PDDG[15] and RM1.[16]

• The option of using the AMBER[17] and CHARMM[18] force fields in addition to OPLS.

• The possibility of performing geometry optimizations or molecular dynamics and Monte Carlo simulation in systems with arbitrary crystal symmetry and without the restriction of the minimum image convention.

• The ability to read, write, or otherwise transform molecular data between a larger range of standard molecular formats, including MDL MOL format[19] and SMILES.[20]

Two features currently missing from pDynamo that were present in fDynamo are the abilities to calculate analytical second derivatives for MM potentials and to perform path-integral molecular dynamics simulations.

2.2. General Structure of the Program. pDynamo is structured as a collection of modules and packages written in Python,[4] which is a dynamic, high-level, scripting language that is becoming increasingly popular in scientific computing due to its power and ease of use. pDynamo provides no interface language of its own, and so users perform pDynamo calculations by writing Python programs. Although users only ever see Python, many of the most computationally expensive parts of pDynamo's algorithms are implemented directly in C. The link between the Python and C portions of the code is effected with the glue language Pyrex.[21]

Currently pDynamo has about 140 Python modules made up of approximately 65000 lines of computer code (excluding third-party libraries) of which 35% are Python, 15% Pyrex, and 50% C. The modules are divided into four groups. Three of these are Python packages which in order of precedence are:

• **pCore** is the most fundamental package in the pDynamo program and contains modules that cope with basic programming and mathematical tasks, such as file handling, linear algebra, and the optimization of functions. The modules in this package have been designed to be independent of molecular simulation and could, if necessary, be employed in other Python programs.

• **pDynamo** has modules that deal with the definition of a molecular system and the calculation of its potential energy using either QC, MM, or hybrid potential approaches.

• **pBabel** defines the modules that handle the input, output, and transformation of molecular data between various standard representations or file formats.[22]

The fourth group of modules, pMoleculeScripts, does not comprise a package but consists of a series of miscellaneous Python scripts which carry out various common tasks in molecular simulation, such as geometry optimization, normal-mode analysis, and molecular dynamics simulation. Modules in pMoleculeScripts employ items from all of the packages described above.

Python is not a pure object-oriented language but it permits object-oriented programming, and it is this style which pDynamo employs. All items in pDynamo are organized into classes, and operations are performed by setting the attributes or invoking the methods of class instances. Although elegant, this approach can sometimes be cumbersome, and so it is convenient for certain classes to define "helper" functions that carry out particular operations without having to explicitly instantiate them.

To get a flavor of what this means and also of the flexibility that Python provides consider the pDynamo program shown in Figure 1. This calculates the energy and various other properties of a water molecule using the AM1, MNDO, and PM3 semiempirical QC methods. The salient points of this program are as follows, noting that all blank lines and lines starting with a hash character, #, are ignored:

• **Line 1** is a Python documentation line that gives a brief description of what the program does.

• **Lines 3−5** import items from the various pDynamo packages that are to be used later in the program.

```
1  """Example water calculation."""
2
3  from pBabel  import MOLFile_ToSystem
4  from pCore   import logfile
5  from pDynamo import QCModelMNDO
6
7  # . Define the energy models.
8  energymodels = [ QCModelMNDO ( "am1"  ), \
                    QCModelMNDO ( "mndo" ), \
                    QCModelMNDO ( "pm3"  )  ]
9
10 # . Loop over the energy models.
11 results = []
12 for model in energymodels:
13     molecule = MOLFile_ToSystem ( "water.mol" )
14     molecule.DefineQCModel ( model )
15     molecule.Summary ( )
16     energy  = molecule.Energy ( )
17     charges = molecule.AtomicCharges ( )
18     dipole  = molecule.DipoleMoment  ( )
19     results.append ( ( model.label, energy, charges, dipole.Norm2 ( ) ) )
20
21 # . Output the results.
22 table = logfile.GetTable ( columns = [ 10, 20, 20, 20, 20, 20 ] )
23 table.Start  ( )
24 table.Title  ( "Energy Model Results for Water" )
25 table.Heading ( "Model"  )
26 table.Heading ( "Energy" )
27 table.Heading ( "Charges", columnspan = 3 )
28 table.Heading ( "Dipole" )
29 for ( label, energy, charges, dipole ) in results:
30     table.Entry ( label )
31     table.Entry ( "%.1f" % ( energy, ) )
32     for charge in charges: table.Entry ( "%.3f" % ( charge, ) )
33     table.Entry ( "%.3f" % ( dipole, ) )
34 table.Stop ( )
```

Figure 1. A pDynamo Python program for calculating the energy of a water molecule using the AM1, MNDO, and PM3 semiempirical methods.

• **Line 8** defines a Python list containing the QC energy models which are to be employed in the calculation. Each item in the list is created by instantiating the class QCModelMNDO with, as argument, a string that indicates the appropriate MNDO Hamiltonian to use.

• **Lines 12−19** form a loop in which the different energy models are iterated over. Line 12 initiates the loop by extracting, at each iteration, a model from the list of energy models, whereas lines 13−19 form the loop's body. In contrast to other languages, Python employs indentation to indicate which lines are to be iterated over, a return to the original indentation level (in this case line 22) denoting an end to the loop's scope.

• **Line 13** reads a MOL file of a water molecule using the MOLFile_ToSystem helper function of the MOLFileReader class. This function instantiates the class, opens, parses, and closes the file, and then returns the file's data as an instance of the class System which, in this case, is assigned to the variable molecule. System is in many respects the most important pDynamo class as its instances represent molecular systems that are to be simulated.

• **Lines 14−15** invoke methods of molecule to define its QC energy model and to output a concise summary of its composition.

• **Lines 16−19** calculate the potential energy, the atomic charges (using a Löwdin analysis),[23] and the dipole moment vector for molecule. The results, along with the name of the energy model (model.label), are saved on line 19 in the list called results which was created before entry to the loop on line 11. Note that the entire dipole moment vector is not saved but only its magnitude which is determined by invoking the vector's Norm2 method.

• **Lines 22−34** output the results of the calculation to a table. Line 22 creates the table by invoking the appropriate method of the variable logfile which, roughly speaking, is pDynamo's equivalent of standard output. Lines 23−28 set up the table and its headers whereas lines 29−33 form a loop that unpacks the results for each energy model from results and then prints them.

2.3. Python/C versus Fortran. There were a number of complementary reasons why the choice was made to switch to Python from Fortran for the new version of Dynamo: (i) Python is a high-level, dynamic language, with features such as object-orientation and garbage collection, that greatly simplifies the writing of programs, and (ii) Python is free and open-source and has implementations for many different platforms and operating systems. In constrast, even after more than 15 years, there are still no fully reliable, open-source Fortran 90 compilers, which has complicated the use of fDynamo. (iii) Python has a very active development community and an extensive standard module library that performs a wide range of different tasks. This is an advantage when compared not only to Fortran but also to other scripting languages, such as Ruby.[24] (iv) Running a fDynamo program requires a preliminary compilation step, but this is unnecessary for pDynamo as running a Python program requires no (explicit) compilation step and pDynamo's C extension modules are compiled when the program is installed.

On the downside, Python does have some disadvantages compared to Fortran (or other compiled languages, such as C). One of these is that many errors that would be caught at compilation time in Fortran programs do not appear until run time in Python because it is a dynamically typed language. This behooves users to check thoroughly their scripts on short simulations before running longer ones.

A more serious problem is one of execution speed as Python programs are normally much slower than programs of compiled languages. This drawback can be alleviated by coding the computationally demanding parts of algorithms in a compiled language (C in the case of pDynamo), but the process of linking Python and C code is nontrivial. The use of Pyrex helps greatly in this procedure, but it is yet another language that has to be mastered.[21]

Given that one of the most important roles of pDynamo in our laboratory is as a testbed for trying out new algorithms, Python's flexibility has proved more important than the performance penalty that its use incurs. In general, though, for standard types of calculation we aim to have no more than a factor of 2 speed difference between pDynamo and an equivalent algorithm coded in a compiled language.

3. Specific Topics

In this section, we illustrate with examples some aspects of pDynamo's QC and hybrid QC/MM potentials that have not been presented elsewhere and, in particular, its DFT capability. As yet, we have completed no full enzyme reaction study with pDynamo, but such work is underway and will be reported in due course.

3.1. Density Functional Theory Implementation. Semiempirical QC methods have proved very useful in hybrid potentials, but, clearly, there are situations in which ab initio or DFT methods are necessary, either for validating the results of semiempirical calculations or for treating systems for which semiempirical methods are not very reliable, such as those that contain transition methods. Given the success and widespread use of DFT methods, we chose to incorporate one of these techniques into Dynamo. Initial work was done in fDynamo but was transferred to pDynamo as this version developed.

We considered two approaches for adding a DFT method to Dynamo. The first was to loosely couple Dynamo with an existing stand-alone DFT program and enable communication either through files or a minimal interface. The second was to embed the DFT code into Dynamo in the same way as the code of its existing QC methods. There are advantages with both approaches. The former is easier because changes to each of the coupled programs can be minimized, whereas the latter allows a greater freedom in the implementation of the DFT hybrid potential. In the end we adopted the second approach because, when we started this work, we were unable to find a reliable, open-source DFT program that was appropriate for our needs, and so we decided to write our own. Subsequently, we have also employed the first approach, but we shall leave a discussion of this until section 3.3.

For our initial DFT implementation, we selected a method based upon Gaussian basis functions, as opposed to plane-waves or other types of basis function, as we deem Gaussians more suitable for the types of hybrid potential that we use and the systems that we study. Currently, pDynamo's DFT module has the following features:

• Standard local and nonlocal (or generalized gradient approximation) functionals are supported but only those that do not include Hartree–Fock exchange.[25] This means, for example, that BLYP is present but not B3LYP.[26]

• A Coulomb-fitting (or RI) approximation is employed to calculate the electron–electron repulsion terms in the Kohn–Sham equations.[27,28] All other terms are evaluated directly with the orbital basis. The Coulomb-fitting basis, like the orbital basis, uses Gaussians.

• Mixed Cartesian and real spherical-harmonic Gaussian basis sets can be treated for both the orbital and Coulomb-fitting functions. At present, integrals and their first derivatives may be calculated if these involve functions of *g* angular momentum or less, although extension to functions of higher angular momentum would be straightforward. pDynamo is distributed with a number of standard orbital and Coulomb-fitting bases.

• All integrals are evaluated at the start of each energy calculation and stored in memory. This improves speed but limits the size of calculation that can be done on a machine with 2 Gb of memory to one with approximately 400 orbital and 1000 Coulomb-fitting functions, respectively. It is possible that this restriction will be removed in the future by going to a direct scheme, in which integrals are re-evaluated when they are needed, or to one that uses external files.

• The DFT correlation and exchange terms are evaluated using Euler-Maclaurin radial quadrature and Lebedev angular integration on atom-centered grids combined using Becke weights.[29,30] A variety of grids is provided which give integrations of different accuracy.

The DFT implementation is reasonably efficient and, most importantly, is done in such a way that whenever a semiempirical QC calculation is performed, a DFT calculation may be carried out instead. Thus, for example, in the example of Figure 1 it would have been possible to add an extra item to the energy models list, defined on line 8, that corresponded to a DFT QC energy model with a particular functional, integration grid and orbital and Coulomb-fitting basis sets.

3.2. The Hybrid Potential. pDynamo's hybrid potential mirrors, in most respects, that of fDynamo[5] and follows closely the one developed by Field et al.[31] In brief, the potential partitions the atoms in a system into two regions, one QC and the other MM. The total potential energy of the system, E, can then be written as a sum of three terms—one for the atoms in the QC region, one for those in the MM region, and a term which describes the interactions between the two—as follows:

$$E[\Psi|\rho] = E_{\mathrm{QC}}[\Psi|\rho] + E_{\mathrm{MM}} + E_{\mathrm{QC/MM}}[\Psi|\rho] \qquad (1)$$

The energy expression for the atoms in the QC region, E_{QC}, takes the form that is appropriate for the QC method being used and will be a function(al) of the system's wave function, Ψ, for a molecular orbital method, or its density, ρ, for a DFT method. Likewise, the energy of the MM region, E_{MM}, is evaluated in the way appropriate for the force field being used. This term does not depend on the QC electronic variables, Ψ or ρ, because all the force fields currently available in pDynamo are nonpolarizable.

The QC/MM interaction energy, $E_{\mathrm{QC/MM}}$, consists of the following two terms:

$$E_{\mathrm{QC/MM}}[\Psi|\rho] = E^{\mathrm{el}}_{\mathrm{QC/MM}}[\Psi|\rho] + E^{\mathrm{LJ}}_{\mathrm{QC/MM}} \qquad (2)$$

The first term, $E^{\mathrm{el}}_{\mathrm{QC/MM}}$, is for the electrostatic interactions between the atoms of the QC and MM regions and will be described in detail in the next section as it is here that some changes have been made. The second term, $E^{\mathrm{LJ}}_{\mathrm{QC/MM}}$, is the Lennard-Jones interaction energy between the atoms of the two regions and is calculated in the usual way for the MM method.

Once the energy expression of eq 1 has been defined, it can be used to set up the equations that must be solved to obtain the wave function or electron density of the atoms in the QC region in the standard way. This gives rise to the Roothaan-Hall and Kohn–Sham equations for molecular orbital and DFT methods, respectively.[6,25] In practice, pDynamo employs a single spin-restricted or spin-unrestricted determinant to represent the electronic wave function or density for the system and solves the Roothaan-Hall or Kohn–Sham equations using a self-consistent field (SCF) technique, iterated to convergence, every time the geometry of the system changes or, in other words, at each step of a geometry optimization or of a molecular dynamics or Monte Carlo calculation. It is important to emphasize that the optimized wave function or density will be polarized by the MM environment because the electrostatic QC/MM interactions are included in the SCF procedure.

In those cases where there are covalent bonds between QC and MM atoms, pDynamo employs a link-atom technique which is identical, except for the electrostatic interactions, to that of fDynamo.[5] In this scheme, all MM atoms that have covalent bonds to QC atoms are marked as "boundary" atoms. Each boundary atom has two components—the MM atom, with its associated parameters, which represents the boundary atom in the MM calculation, and a hydrogen link-atom which enters into the QC calculation. The link-atom does not have an independent position of its own because its coordinates are constructed as an analytic function of the coordinates of its parent MM and QC atoms each time an energy is calculated. This is done by placing the link-atom along the QC-MM bond at a suitable distance (~1 Å) from the QC atom. The gradients of the energy with respect to the link-atom coordinates, if these are calculated, are easily partitioned between the parent MM and QC atoms by a straightforward application of the chain rule. The identification and handling of boundary and link-atoms is handled automatically by pDynamo and requires no intervention by the user. Although link-atom methods have been criticized, they are robust and easy to implement and, in our experience, are not consistently less accurate than alternative approaches.

3.2.1. Electrostatic Interactions. The principle change in the hybrid potential between pDynamo and fDynamo lies in the treatment of the electrostatic QC/MM interactions. In fDynamo, the QC/MM electrostatic energy was calculated with the following formula

$$E^{\mathrm{el}}_{\mathrm{QC/MM}} = \sum_m Q_m \Bigg\{ \sum_q Z_q f_{\mathrm{nucleus}}(\boldsymbol{r}_q, \boldsymbol{r}_m) - \int d\boldsymbol{r}\rho(r) f_{\mathrm{electron}}(\boldsymbol{r}, \boldsymbol{r}_m) \Bigg\} \qquad (3)$$

where ρ is the electron density of the QC atoms expressed in terms of the QC density matrix and products of orbital basis functions, $\boldsymbol{r}$ is a position vector, Z is a nuclear charge, Q is a partial charge, and the subscripts q and m refer to QC and MM atoms, respectively. The functions f_{nucleus} and f_{electron} determine the distance-dependence of the electrostatic interaction. For a DFT method they are both equal to the inverse distance (i.e., $1/|\mathbf{r} - \mathbf{r}_m|$), whereas for the semiempirical methods they have more complicated forms.

This expression is, of course, the natural, exact equation for the interaction energy between a set of point charges and a charge distribution consisting of a mixture of point charges (nuclei) and continuous densities (electrons). However, when extending our hybrid potentials with a DFT method, the question arose as to whether this is necessarily the best approach. First, the method is very expensive if there are thousands of MM atoms as is typically the case for the systems we study. Second, there is a great disparity in the representations of the QC and MM atom charge distributions as it is very detailed for the QC atoms and much more primitive for the MM atoms. This being so, is it worth calculating the interaction to such a high precision? Finally, there is the 'charge-sloshing' effect. In a pure QC calculation, the charge density of an atom is restricted by interaction with the charge densities of the QC atoms around it. In a hybrid potential (at least at the QC/MM boundary), charge can leak from the QC region and accumulate unphysically in the MM region. This is not so much of a problem for semiempirical QC methods because their basis functions are very localized but becomes one for ab initio and DFT calculations with

Gaussian basis sets, especially as the size and diffusiveness of the basis increases.

This led us to consider the following simplified expression for the interaction energy which is appropriate for both semiempirical and DFT hybrid potentials:

$$E^{\mathrm{el}}_{\mathrm{QC/MM}} = \sum_m \sum_q Q_m \widetilde{Q}_q[\rho] f_{\mathrm{charge}}(\boldsymbol{r}_m, \boldsymbol{r}_q) \qquad (4)$$

In this equation f_{charge} is the function that determines the distance-dependence of the electrostatic interaction and $\tilde{Q}_q$ is an effective charge for the QC atom q derived from the atom's nuclear charge and a population analysis of the electron density, ρ. Note that the charges, $\tilde{Q}_q$, are redetermined at each iteration of an energy calculation's SCF procedure, so that they reflect the changes that are occurring in the electron density as it converges.

Equation 4 addresses the three concerns cited above as follows: (i) it is cheaper to calculate, because population analysis is normally much less expensive than integral calculation even if it has to be done at each step of an SCF calculation; (ii) the expression treats the charge distributions of the atoms in the QC and MM atoms in a much more symmetrical way; and (iii) "charge-sloshing" can no longer occur because the QC atom charges are localized on their nuclei. The adoption of eq 4 also has the important advantage that many of the efficient methods that exist for the treatment of long-range interactions in MM calculations, whether these be truncation or Ewald techniques, can be adapted straightforwardly to the hybrid potential case. It is our intention to do this once we have confirmed to ourselves the robustness of the point charge approach.

Clearly, the accuracy of eq 4 depends crucially upon the form of the function f_{charge} and what type of population analysis is used to determine the charges $\tilde{Q}_q$. We have examined a large number of different approaches, and our work indicates that eq 4 can give results of an accuracy comparable to that of eq 3. We have also looked at the best way of calculating the electrostatic interactions for link atoms with this representation. An article on this research is in preparation and should be submitted shortly.[32] It is worth noting that although the current version of pDynamo employs eq 4 to describe the QC/MM electrostatic interactions, it is not excluded that future releases will also incorporate methods based upon eq 3 if these are deemed necessary.

3.2.2. Example Calculations. To give an illustration of the capabilities of pDynamo's hybrid potentials, we describe in this section calculations on two different types of periodic system, the first a cubic box of 215 water molecules solvating a methane molecule and the second a series of three molecular, organic crystals with a mixture of space groups. In each case, the calculations were repeated with a pure MM potential and with both semiempirical and DFT QC/MM hybrid potentials. The OPLS-AA force field[10] with the TIP3P water model[33] was taken for the MM potential, the AM1 method[11] for the semiempirical potential and, for the DFT method, the BLYP functional[26] with the 3−21G orbital[34] and DeMon Coulomb-fitting[35] basis sets.

All nonbonding (electrostatic and Lennard-Jones) interactions between MM atoms and between QC and MM atoms were treated with the atom-based, force-switching (ABFS) method[36] with inner and outer cutoffs of 8 and 12 Å, respectively. This means that the f_{charge} function in eq 4 was the one appropriate for ABFS electrostatic interactions. Effective charges for the QC atoms in these interactions were derived using a Löwdin population analysis of the QC density matrix at each iteration of the SCF calculation. The derivatives of the energy expression of eq 4 with Löwdin charges are straightforward to determine using standard techniques, both with respect to density matrix elements (required for construction of the Fock and Kohn−Sham matrices) and atomic coordinates (needed for calculation of the atomic forces).[37]

As stated in section 2.1, fDynamo required that the minimum image convention held when treating periodic systems and, in addition, that the nonbonding interaction cutoff was sufficiently large that QC atoms in the central, periodic image did not interact with QC atoms in neighboring images. In pDynamo this is no longer the case, and so it is common (depending upon nonbonding cutoff and system size) to have QC atoms of different images interact. These situations are straightforwardly handled by modifying the energy expression of eq 4 to include interactions of the form $\tilde{Q}_q\tilde{Q}_{q'}\, f_{\mathrm{charge}}\,(\mathbf{r}_{q'}, \mathbf{r}_{q'})$ where the subscripts q and q' refer to QC atoms in different images. This modification also has the consequence that it is possible to perform QC/MM calculations in which all atoms in the central image are defined to be in the QC region and there are no MM atoms at all! In these cases, the only manifestation of the MM potential is in the determination of the Lennard-Jones interactions between the (QC) atoms of the central image and their (MM) copies in the surrounding images. We shall see examples of this type of calculation later.

As a first illustration, we simulated a system consisting of a methane molecule solvated by 215 waters in a cubic box of approximate dimension 18.8 Å. Calculations were done with a pure MM potential and with hybrid potentials in which the atoms of methane were treated quantum chemically. Molecular dynamics simulations were performed using the constant pressure and temperature leapfrog Verlet algorithm of Berendsen et al.[38] and consisted of a short equilibration run of 10 ps followed by a data collection phase of 100 ps in which coordinate data were saved every 50 fs. Simulations were carried out at 1 atm and 300 K with pressure and temperature coupling constants of 2000 ps atm and 0.1 ps, respectively. The integration time step was 1 fs meaning that the simulations with each potential needed approximately 1.1×10^5 energy and force calculations.

The radial distribution functions (RDFs) between the methane carbon and the water oxygens determined from these simulations are shown in Figure 2. The curves are very similar with the first peak being shortest and broadest for the MM potential and sharpest in the QC(AM1)/MM case. The QC(DFT)/MM curve is intermediate between the two. The number of waters in the first solvation shell of methane may be estimated by integrating the RDFs and choosing the values that correspond to the distances of their first troughs (~5.5 Å). This gives approximately 21 waters for each simulation.

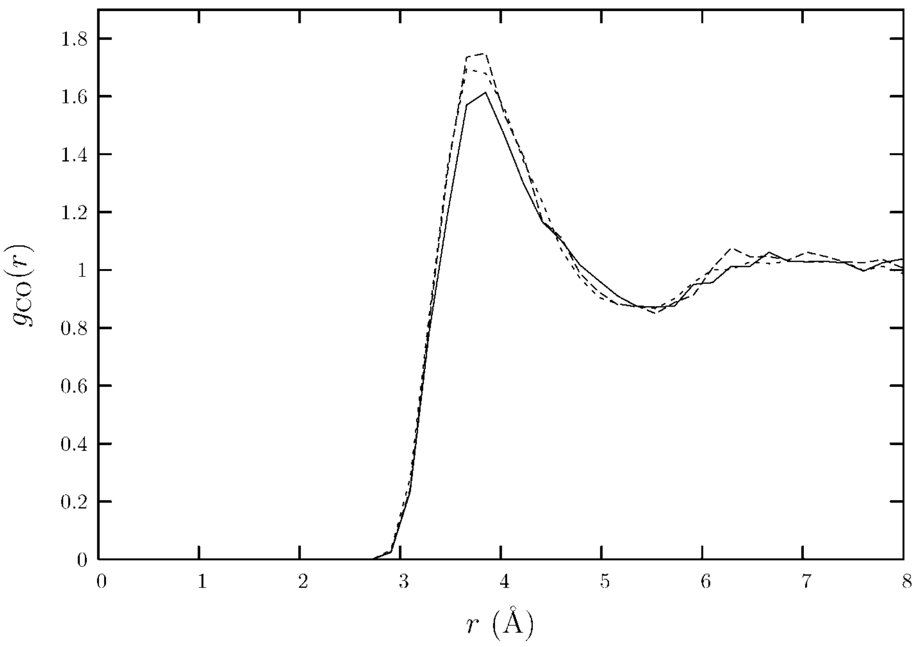

Figure 2. Radial distribution functions, $g_{CO}(r)$, determined from simulations of a molecule of methane solvated in a cubic box of 215 water molecules with the following potentials: MM — solid line; QC(AM1)/MM — long dashed line; QC(DFT)/MM — short dashed line. In the last two calculations the methane molecule is in the QC region and the waters are in the MM region.

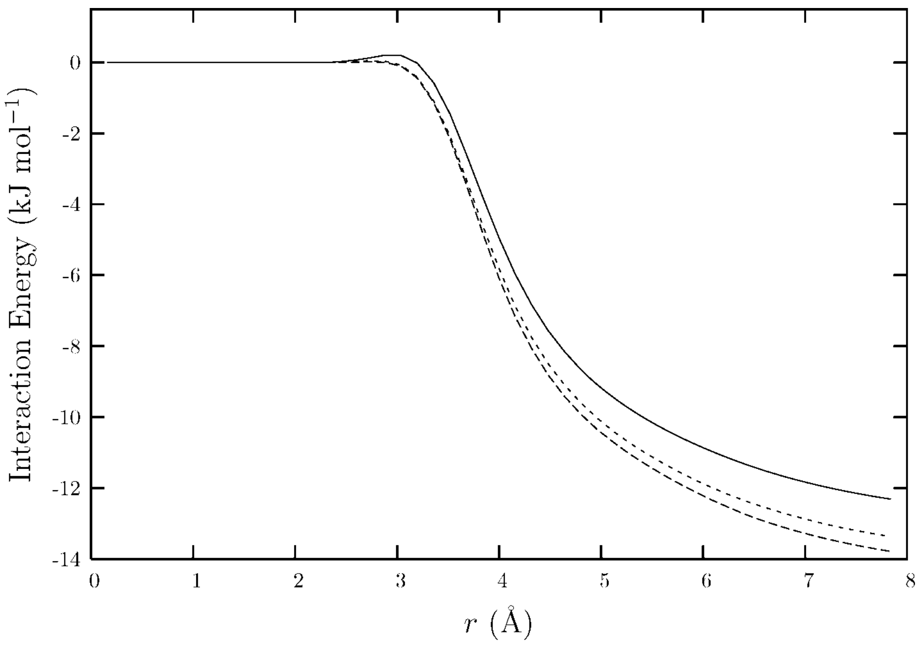

Figure 3. The average energy of interaction between methane and the surrounding water molecules as a function of cutoff distance determined from the simulations with the following potentials: MM — solid line; QC(AM1)/MM — long dashed line; QC(DFT)/MM — short dashed line.

Figure 3 shows the average energies of interaction between methane and the waters as a function of distance. The curves are again very similar although for all distances for which there is an interaction energy, the MM energies are the least stabilizing and the QC(AM1)/MM energies the most. At distances around 3 Å the MM energies are even slightly destabilizing, a feature which is absent from the other two calculations. Because there are 21 water molecules in the first solvation shell of methane in each simulation, methane's average energy of interaction with a water in that shell is approximately -0.5 kJ mol $^{-1}$.

As a final analysis, it is also possible to calculate methane's charge distribution from the hybrid potential simulations. The average Löwdin charges and their standard deviations on the carbon were -0.27 ± 0.02 and -0.37 ± 0.01 in the AM1 and DFT cases, respectively.

The second set of calculations involved a series of three molecular, organic crystals. These were obtained from the

Table 1. Results for the Crystal Geometry Optimization Calculations[a]

crystal	property	MM	AM1/MM	AM1	DFT/MM	DFT
HXACAN19	ΔE	−95.6	−175.8	−193.0	−152.0	−164.1
	RMSCD	0.27	0.18	0.06	0.18	0.13
	a	12.48 (0.39)	12.50 (0.37)	12.46 (0.41)	12.50 (0.37)	12.55 (0.32)
	b	9.17 (0.20)	9.14 (0.23)	9.73 (−0.36)	9.12 (0.25)	9.63 (−0.26)
	c	7.23 (−0.14)	7.25 (−0.17)	7.19 (−0.11)	7.26 (−0.17)	7.22 (−0.14)
	β	115.9 (−0.3)	115.9 (−0.2)	119.46 (−3.8)	115.7 (−0.1)	119.2 (−3.6)
LCDMPP10	ΔE	−172.6	−199.7	−302.2	−193.0	−268.9
	RMSCD	0.10	0.10	0.10	0.11	0.08
	a	7.62 (0.45)	7.62 (0.45)	7.66 (0.41)	7.66 (0.40)	7.61 (0.45)
	b	6.11 (−0.02)	6.10 (−0.01)	5.99 (0.09)	6.11 (−0.03)	6.04 (0.05)
	c	5.34 (−0.18)	5.34 (−0.19)	5.22 (−0.07)	5.33 (−0.18)	5.38 (−0.22)
	α	133.5 (−1.8)	133.6 (−1.9)	131.8 (−0.1)	133.7 (−2.0)	132.4 (−0.7)
	β	79.7 (2.7)	79.9 (2.5)	80.7 (1.7)	79.3 (3.1)	82.0 (0.4)
	γ	105.9 (0.7)	105.8 (0.8)	106.6 (0.0)	106.1 (0.4)	106.5 (0.1)
WABZOO	ΔE	−128.8	−134.9	−144.4	−105.7	−86.3
	RMSCD	0.37	0.20	0.23	0.25	0.34
	a	12.65 (−0.05)	12.64 (−0.05)	12.68 (−0.08)	12.59 (0.00)	12.59 (0.00)
	α	117.9 (0.1)	118.0 (−0.01)	117.9 (0.1)	117.9 (0.1)	117.9 (0.1)

[a] MM, AM1/MM, AM1, DFT/MM, and DFT refer to calculations with the following potentials: MM; QC(AM1)/MM with both QC and MM atoms in the central image; QC(AM1)/MM with no MM atoms in the central image; QC(DFT)/MM with both QC and MM atoms in the central image; and QC(DFT)/MM with no MM atoms in the central image. ΔE and RMSCD are the energy lowering (kJ mol^{-1}) and RMS coordinate difference (Å) between the experimental and geometry optimized structures. The RMSCDs were calculated after superimposing the optimized structure upon the experimental one. a, b, c, α, β, and γ refer to crystal unit cell distances (Å) and angles (°) for the optimized structures with the numbers in brackets giving the difference from the experimental values (experimental minus optimized).

Cambridge Structural Database[39] with codes HXACAN19, LCDMPP10, and WABZOO and correspond to the molecules paracetamol,[40] cyclo-L-alanyl-L-alanyl,[41] and (3aα, 4α,4α,7aα)-2,2-dimethyl-3a,4,4,7a-tetrahydro-1,3-benzodioxolane-4,7-diamine,[42] respectively. The numbers of atoms in the central image and the space groups are 20, 20, and 29 and $P21a$, $P1$, and $R3$, respectively.

Geometry optimizations of the crystals were done with a pure MM potential and with semiempirical and DFT QC/MM hybrid potentials in which both some and all of the atoms in the central image were included in the QC region. For those hybrid potential calculations in which there were both QC and MM atoms in the central image, the atoms in the QC region were the methyl group (HXACAN19), one of the two symmetry-equivalent methyls (LCDMPP10) and the five-membered ring along with its two methyl groups (WABZOO). These partitionings mean that there will be link atoms, one for each of HXACAN19 and LCDMPP10 and two for WABZOO. To avoid catastrophic electrostatic interaction between the partial charges of the link atom and its neighboring MM atom, we adopted the redistributed charge (RC) method of Lin and Truhlar[43] for these illustrative calculations. This involves an equal redistribution of the partial charge of the MM atom to the midpoints of the bonds between it and the MM atoms to which it is covalently bound. A full assessment of this and other approximations will be left to our future paper.[32]

Optimizations were carried out with a trust-radius, quasi-Newton algorithm with a starting second derivative matrix calculated with the MM potential at the experimental geometry. All degrees of freedom, molecular internal and crystal, were optimized with the restriction that the experimental space group was maintained. A summary of the results for the calculations is shown in Table 1. All optimizations converged and no significant difference was seen in the convergence behavior between the calculations with different potentials. Each of the potentials produced structures which remained close to the experimental one.

3.3. Coupling to External Programs. As an alternative to pDynamo's native DFT algorithms, it is also possible to employ third party QC programs. This can be advantageous when these programs are more efficient or provide complementary functionality, such as Hartree−Fock, post-Hartree −Fock, or excited-state methods. In this section, we illustrate what is possible by describing the coupling between pDynamo and the program ORCA.[44] This is a free, general purpose, quantum chemical package with a range of ab initio, DFT, and semiempirical QC algorithms that is available in binary form only.

Like many QC programs, ORCA requires a simple text input file in order to run and writes results to a text output file. Additional files are also produced in both text and binary formats that contain extra information about the calculation or quantities that could prove useful in future jobs (such as density matrices). This mode of functioning clearly suggests a model of coupling in which pDynamo writes an ORCA input file, launches the ORCA executable, waits for its termination, and then parses the output and other files for whatever information it requires.

In our current implementation, pDynamo controls all aspects of a simulation, and ORCA is only called when a QC energy and, optionally, its gradients are needed. All geometry optimization and other types of calculation, such as molecular dynamics integration and reaction-path finding, are done with pDynamo's own algorithms. The coupling between pDynamo and ORCA is transparent to users and means that ORCA can be invoked in exactly the same way as, and employed interchangeably with, pDynamo's other, in-built QC algorithms. Thus, in the example of Figure 1, to calculate the energy, atomic charges, and dipole moment of the water molecule with one of ORCA's QC methods, it suffices to instantiate the class corresponding to the ORCA

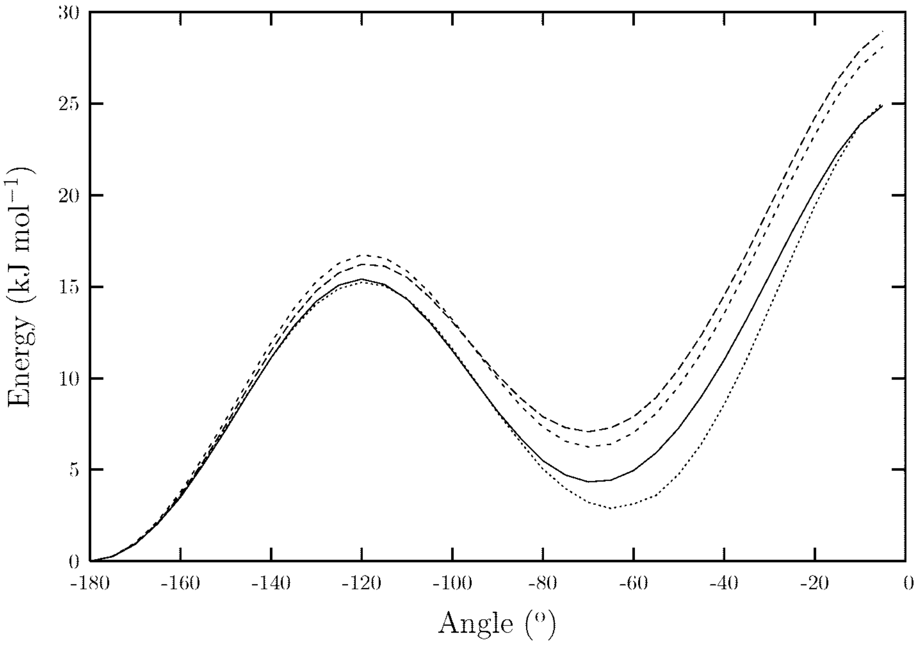

Figure 4. Energy profiles for rotation about the central C−C bond of butane calculated with the following potentials: MM − solid line; QC(AM1)/MM with half the molecule in the QC region − long dashed line; QC(MP2)/MM with half the molecule in the QC region − short dashed line; MP2 − dotted line. Only half the profile is shown as it is symmetrical.

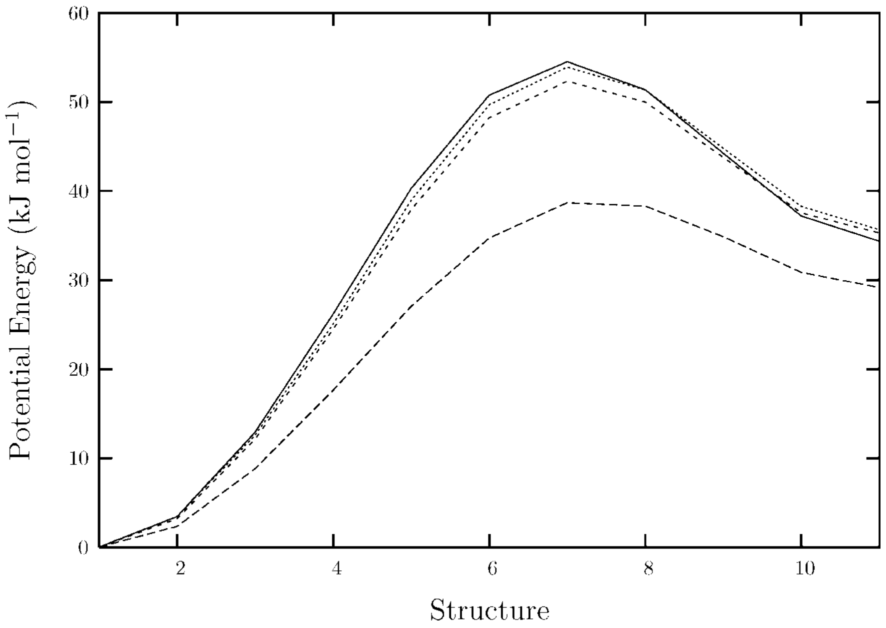

Figure 5. Energy profiles for the cyclohexane chair/twist-boat isomerization calculated with the following potentials: MM − solid line; QC(AM1)/MM − long dashed line; QC(HF)/MM − short dashed line; QC(MP2)/MM − dotted line.

QC model with the appropriate options and then add this to the energy models list.

The ORCA input file written by pDynamo when it invokes ORCA contains all the information necessary to perform an energy and gradient calculation. This includes a definition of the QC atoms and their coordinates along with various keywords that specify quantities such as the QC method, basis sets, and the system's charge and spin multiplicity. ORCA also permits QC calculations in the presence of non-QC point charges, and so, in a hybrid potential calculation, the input file contains the charges and the coordinates of the MM atoms as well.

Once the ORCA calculation is finished, pDynamo parses the appropriate files produced by ORCA during the run. These include the output file from which information, such as the atomic charges, is obtained and another file called the 'engrad' file. The latter contains the total energy of the system and the gradients with respect to the coordinates of the QC and, equally importantly, the non-QC (or MM) atoms. The fact that ORCA calculates these latter gradients is in marked contrast to most other QC programs and greatly simplifies its use as part of a hybrid potential.

The energy calculated by ORCA in the presence of point charges includes not only the interaction energy between the

QC atoms and the point charges but also the interaction energy between the point charges themselves. This means that the only electrostatic energy that pDynamo needs to calculate when using ORCA is that which would normally be excluded between MM atoms, i.e. that between MM atoms that are separated by one, two, and three bonds. This is rapid and straightforward to do and is subtracted from ORCA's total energy. All other energy terms, including the Lennard-Jones interactions involving QC and MM atoms, are handled by pDynamo in its normal fashion.

As a final point in the discussion of the technical aspects of the pDynamo/ORCA coupling, it is worth noting that hybrid potentials employing ORCA will be less flexible than those using pDynamo's in-built QC models because ORCA calculates the QC/MM electrostatic interactions following eq 3, not eq 4. This is time-consuming if there are large numbers of point charges; limits applications to 'vacuum' systems as periodic boundary conditions cannot be treated and may introduce artifactual boundary effects when simulating large systems due to the long-range nature of the electrostatic $1/r$ interaction.

We illustrate the coupling of pDynamo and ORCA with two sets of simple hybrid potential calculation, one that determines the energy profile for rotation about the central C−C bond of butane and the other for the chair/twist-boat isomerization in cyclohexane. In both sets of calculation we employed four potentials consisting of (i) a pure MM model using the OPLS-AA force field;[10] (ii) a QC(AM1)/MM hybrid potential using pDynamo's in-built AM1 method; (iii) a pDynamo/ORCA hybrid potential with a Hartree−Fock method and the 6−31G* basis set;[45] and (iv) a pDynamo/ORCA hybrid potential with an MP2 method and the 6−31G* basis set. To facilitate comparison between the different potentials all nonbonding interactions calculated by pDynamo were determined exactly (i.e., without truncation or the ABFS approximation), and the electrostatic interactions around link-atoms in ORCA were evaluated using the same RC approximation as described in section 3.2.2.

The energy profiles for rotation about the central C−C bond of butane were obtained by optimizing the butane structure with the appropriate potential while constraining the C−C−C−C torsion angle to have a specific value. Optimizations were done in 5^o increments in the range -180^o to 0^o (as the profiles are symmetric about 0^o). In the calculations with the three hybrid potentials, four different partitionings of the molecule were tried with increasing numbers of atoms in the QC region. These were a terminal methyl group, half the molecule, all but a terminal methyl group, and the complete molecule (i.e., equivalent to a pure QC calculation).

A selection of the profiles that were obtained is shown in Figure 4, whereas the full set of profiles are included as Supporting Information. From Figure 4 it can be seen that the pure MM and pure MP2 profiles are very similar although the energy of the MP2 s minimum (at -60^o) is slightly lower. Likewise the two hybrid potential profiles, one AM1 and one MP2, agree well although the energy barriers, and the energies of the second minimum are higher than for the MM and MP2 profiles.

As a second example, we calculated energy profiles for the isomerization of the chair to twist-boat forms of cyclohexane using the self-avoiding walk method of Elber et al.[46] In the cases where hybrid potentials were used, half the molecule was placed in the QC region (i.e., three consecutive CH_2 groups). Energy profiles were determined using 11 structures and are shown in Figure 5. It can be seen that the profiles obtained with the MM potential and with the two ab initio hybrid potentials (HF and MP2) agree very well, whereas the QC(AM1)/MM profile has a much lower barrier.

4. Summary

In this paper we have presented a new version of our Dynamo simulation program which is designed to study molecular systems with hybrid QC/MM potentials. The new version, pDynamo, is written in the languages Python and C and succeeds the previous version, fDynamo, which was written in Fortran 90. The switch to Python makes pDynamo much easier to use but has required that parts of the program be coded in C so that its computational efficiency approaches that of the Fortran version. pDynamo supports a similar range of molecular simulation algorithms as fDynamo but has some significant new capability, including the addition of a DFT QC method.

Acknowledgment. The author would like to thank Dr. Konrad Hinsen for introducing him to Python, Margherita Moreno and Dr. Guido Raos for help with the crystal calculations, and Dr. Troy Wymore and his colleagues at Pittsburgh Supercomputing Center for hosting the pDynamo web site. Acknowledgements also go to the Institut de Biologie Structurale − Jean-Pierre Ebel, the Commissariat à l'Energie Atomique, and the Centre National de la Recherche Scientifique for support of this work.

Supporting Information Available: Figures of all the rotational barriers calculated in section 3.3. This material is available free of charge via the Internet at http://pubs.acs.org.

References

(1) Warshel, A.; Levitt, M. *J. Mol. Biol.* **1976**, *103*, 227.

(2) Senn, H. M.; Thiel, W. *Top. Curr. Chem.* **2007**, *268*, 173.

(3) Lin, H.; Truhlar, D. G. *Theor. Chem. Acc.* **2007**, *117*, 185.

(4) Python Programming Language. http://www.python.org (accessed April 29, 2008).

(5) Field, M. J.; Albe, M.; Bret, C.; Proust-De Martin, F.; Thomas, A. *J. Comput. Chem.* **2000**, *21*, 1088.

(6) Field, M. J. *A Practical Introduction to the Simulation of Molecular Systems*, 2nd ed.; Cambridge University Press: Cambridge, U.K., 2007.

(7) Field, M. J. pDynamo Molecular Modeling Program. Hosted by Pittsburgh Supercomputing Center, Pittsburgh, U.S.A. http://www.pdynamo.org (accessed April 29, 2008).

(8) CeCILL, the French Free Software License. http://www.cecill.info (accessed April 29, 2008).

(9) The GNU General Public License. http://www.gnu.org/licenses/gpl.html. (accessed April 29, 2008).

(10) Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. *J. Am. Chem. Soc.* **1996**, *118*, 11225.

(11) Dewar, M. J. S.; Zoebisch, E. G.; Healy, E. F.; Stewart, J. J. P. *J. Am. Chem. Soc.* **1985**, *107*, 3902.

(12) Dewar, M. J. S.; Thiel, W. *J. Am. Chem. Soc.* **1977**, *99*, 4899.

(13) (a) Stewart, J. J. P. *J. Comput. Chem.* **1989**, *10*, 209. *ibid* **1989**, *10*, 221.

(14) Fernandez-Galván, I.; Field, M. J. *J. Comput. Chem.* **2008**, *29*, 139.

(15) Repasky, M. P.; Chandrasekhar, J.; Jorgensen, W. L. *J. Comput. Chem.* **2002**, *23*, 1601.

(16) Rocha, G. B.; Freire, R. O.; Simas, A. M.; Stewart, J. J. P. *J. Comput. Chem.* **2006**, *27*, 1101.

(17) Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R., Jr.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A. *J. Am. Chem. Soc.* **1995**, *117*, 5179.

(18) MacKerell, A. D.; Bashford, D.; Bellott, M., Jr.; Evanseck, J.; Field, M. J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; Joseph, 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. E.; Watanabe, M.; Wiórkiewicz-Kuczera, J.; Yin, D.; Karplus, M. *J. Phys. Chem. B* **1998**, *102*, 3586.

(19) Dalby, A.; Nourse, J. G.; Hounshell, W. D.; Gushurst, A. K. I.; Grier, D. L.; Leland, B. A.; Laufer, J. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 244.

(20) Weininger, D. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31.

(21) Pyrex — A Language for Writing Python Extension Modules. http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex (accessed April 29, 2008).

(22) The pBabel package is named after the well-known OpenBabel program but handles by no means as many different file formats, nor is it intended to. To find OpenBabel, see: Open Babel — The Open Source Chemistry ToolBox. http://openbabel.sourceforge.net (acccessed April 29, 2008).

(23) For MNDO semiempirical methods, such as AM1, Löwdin and Mulliken charge analyses are equivalent due to the fact that the overlap matrix is equal to the identity matrix.

(24) Ruby Programming Language. http://www.ruby-lang.org (accessed April 29, 2008).

(25) Koch, W.; Holthausen, M. C. *A Chemist's Guide to Density Functional Theory*; Wiley-VCH: New York, NY, 2000.

(26) (a) BeckeA. D. *Phys. Rev. A* **1988**, *38*, 3098. (b) Becke, A. D. *J. Chem. Phys.* **1993**, *98*, 5648. (c) Stephens, P. J.; Devlin, F. J.; Chabalowski, C. F.; Frisch, M. J. *J. Phys. Chem.* **1994**, *98*, 11623.

(27) Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. *J. Chem. Phys.* **1979**, *71*, 3396.

(28) Eichkorn, K.; Treutler, O.; Ohm, H.; Haser, M.; Ahlrichs, R. *Chem. Phys. Lett.* **1995**, *240*, 283.

(29) Murray, C. W.; Handy, N. C.; Laming, G. J. *Mol. Phys.* **1993**, *78*, 997.

(30) Becke, A. D. *J. Chem. Phys.* **1988**, *88*, 2547.

(31) Field, M. J.; Bash, P. A.; Karplus, M. *J. Comput. Chem.* **1990**, *11*, 700.

(32) Lelimousin, M.; Field, M. J. Manuscript in preparation.

(33) Jorgensen, W. L.; Chandrasekar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, *79*, 926.

(34) Binkley, J. S.; Pople, J. A.; Hehre, W. J. *J. Am. Chem. Soc.* **1980**, *102*, 939.

(35) Godbout, N.; Salahub, D. R.; Andzelm, J.; Wimmer, E. *Can. J. Chem.* **1992**, *70*, 560.

(36) Steinbach, P. J.; Brooks, B. R. *J. Comput. Chem.* **1994**, *15*, 667.

(37) See, for example:(a) Head-Gordon, M.; Pople, J. A. *J. Phys. Chem.* **1988**, *92*, 3063. (b) Field, M. J. *J. Phys. Chem.* **1991**, *95*, 5104.

(38) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; Di Nola, A.; Haak, J. R. *J. Chem. Phys.* **1984**, *81*, 3684.

(39) Allen, F. H. *Acta Crystallogr., Sect. B: Struct. Sci.* **2002**, *58*, 380.

(40) Wilson, C. C. *Z. Kristallogr.* **2000**, *215*, 693.

(41) Benedetti, E.; Corradini, P.; Pedone, C. *Biopolymers* **1969**, *7*, 751.

(42) Mackay, M. F.; Bannell, M. G.; Richards, S. L. *Acta Crystallogr., Sect. C: Cryst. Struct. Commun.* **1993**, *49*, 556.

(43) Lin, H.; Truhlar, D. G. *J. Phys. Chem. A* **2005**, *109*, 3991.

(44) Neese, F. ORCA Quantum Chemistry Program. http://www.thch.uni-bonn.de/tc/orca (accessed April 29, 2008).

(45) Hariharan, P. C.; Pople, J. A. *Theor. Chim. Acta* **1973**, *28*, 213.

(46) Czermiński, R.; Elber, R. *Int. J. Quantum Chem.: Quantum Chem. Symp.* **1990**, *24*, 167.

CT800092P