Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Physics Communications 155 (2003) 244–259
Computer Physics Communications
www.elsevier.com/locate/cpc

ASEP/MD: A program for the calculation of solvent effects combining QM/MM methods and the mean field approximation ☆

I. Fdez Galván, M.L. Sánchez, M.E. Martín, F.J. Olivares del Valle, M.A. Aguilar *

Dpto Química Física, Univ. de Extremadura, Avda de Elvas s/n, 06071 Badajoz, Spain

Received 28 March 2003

Abstract

ASEP/MD is a computer program designed to implement the Averaged Solvent Electrostatic Potential/Molecular Dynamics (ASEP/MD) method developed by our group. It can be used for the study of solvent effects and properties of molecules in their liquid state or in solution. It is written in the FORTRAN90 programming language, and should be easy to follow, understand, maintain and modify. Given the nature of the ASEP/MD method, external programs are needed for the quantum calculations and molecular dynamics simulations. The present version of ASEP/MD includes interface routines for the GAUSSIAN package, HONDO, and MOLDY, but adding support for other programs is straightforward. This article describes the program and its usage.

Program summary

Title of program: ASEP/MD
Catalogue identifier: ADSF
Program Summary URL: http://cpc.cs.qub.ac.uk/summaries/ADSF
Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland
Computer for which the program is designed: it has been tested on Intel-based PC and Sun
Operating systems under which the program has been tested: Red Hat Linux 7.2 and SunOS 5.6
Programming language used: FORTRAN90
Memory required to execute with typical data: greatly depends on the system
No. of processors used: 1
Has the code been vectorized or parallelized?: no
No. of bytes in distributed program, including test data, etc.: 44 544
Distribution format: tar gzip file
Keywords: Solvent effects, QM/MM methods, mean field approximation, geometry optimization
Nature of physical problem: The study of molecules in solution with quantum methods is a difficult task because of the large number of molecules and configurations that must be taken into account. The quantum mechanics/molecular mechanics methods proposed to date either require massive computational power or oversimplify the solute quantum description.

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: maguilar@unex.es (M.A. Aguilar).

0010-4655/$ – see front matter
doi:10.1016/S0010-4655(03)00351-5

Method of solution: A non-traditional QM/MM method based on the mean field approximation was developed where a classical molecular dynamics simulation is coupled with a quantum calculation. The average electrostatic potential generated by the solvent over the solute is calculated from the simulation and introduced into the quantum calculation as an external field. This process can be performed iteratively. Standard external programs are used for the molecular dynamics simulations and for the quantum calculations. The present program acts as an interface and controls the flow of the calculation.
Restrictions on the complexity of the problem: At present, only pure liquids and binary dilute solutions (a single solute molecule) can be studied. For the molecular dynamics only MOLDY is implemented, while GAUSSIAN and HONDO are available for the quantum calculations. Restrictions of the aforementioned programs apply.
Typical running time: Running time depends on the nature of the chemical system and the options passed to the external programs, which are usually by far the longest part of the calculations.
Unusual features of the program: Uses SYSTEM and GETARG calls.

1. Introduction

Quantum Mechanics/Molecular Mechanics (QM/MM) methods [1] are now widely used in the study of molecules in solution. The main advantage of these methods is that they combine a quantum description of the solute, allowing chemical processes to be studied, with a detailed description of the solvent, obtained from simulation techniques. In most QM/MM methods the solute Schrödinger equation has to be solved for each solvent configuration, which implies several thousand quantum calculations. This imposes a limitation on the quality of the quantum description of the solvent (basis set and calculation level) and on the significance of the results (number of configurations considered).

In previous papers [2] we have developed a non-traditional QM/MM method that makes use of the mean field approximation [3] (MFA). In this approximation, the average value of the energies of the different solute–solvent configurations is replaced by the energy of the average configuration. Our method is based on the calculation of the Average Solvent Electrostatic Potential (ASEP) from Molecular Dynamics data (MD); this average potential is introduced into the molecular Hamiltonian of the solute and the Schrödinger equation is solved. This approximation, named ASEP/MD, reduces drastically the number of quantum calculations from several thousands to half a dozen, and introduces no significant inaccuracies [3] in the solute–solvent interaction energy or the solute dipole moment. This reduced number of quantum calculations permits one to use higher quality methods, while having an adequate sampling of the solvent configurations through the molecular dynamics simulations.

This paper presents of the computer program developed to implement our method. The quantum calculations and the molecular dynamics simulations are performed by external programs, intentionally left out of the present implementation. This dependence on external programs allows the users to employ whatever program they are accustomed to or that is best suited to their needs, although specific subroutines for the purpose may have to be written.

2. Method

The main characteristics of the ASEP/MD method have been discussed elsewhere [2]. The following is a short description. As in traditional QM/MM methods, in ASEP/MD the energy and wavefunction of the solvated solute molecule are obtained by solving the effective Schrödinger equation

$$\big(\widehat{H}_{\mathrm{QM}} + \widehat{H}_{\mathrm{QM/MM}}\big)|\Psi\rangle = E|\Psi\rangle. \tag{1}$$

The interaction term, $\widehat{H}_{\mathrm{QM/MM}}$, takes the following form:

$$\widehat{H}_{\mathrm{QM/MM}} = \widehat{H}^{\mathrm{elect}}_{\mathrm{QM/MM}} + \widehat{H}^{\mathrm{vdw}}_{\mathrm{QM/MM}}, \tag{2}$$

$$\widehat{H}^{\mathrm{elect}}_{\mathrm{QM/MM}} = \int \rho\big\langle V_s(r;\rho)\big\rangle \,\mathrm{d}r, \tag{3}$$

where ρ is the solute charge density, obtained from a quantum calculation, and the term $\langle V_s(r;\rho)\rangle$ is the average electrostatic potential generated by the solvent (ASEP) at the position r, and is obtained from MD calculations where the solute molecule has fixed geometry and charge distribution ρ—the angle brackets denote a statistical average. The term $\widehat{H}^{\mathrm{vdw}}_{\mathrm{QM/MM}}$ is the Hamiltonian for the van der Waals interaction, usually represented by a Lennard-Jones potential. Since the solvent structure, and hence the ASEP, is a function of the solute charge density, Eqs. (1) and (3) have to be solved iteratively. In general, only a few cycles of quantum calculation/molecular dynamics simulations are needed for convergence.

In order to facilitate its implementation in a standard quantum program package, the ASEP is represented be means of a set of point charges. The method used to obtain the ASEP and the point charges that approximate it is as follows. First a three-dimensional grid of points is created inside the volume occupied by the solute. Then, for every solvent configuration considered, the electrostatic potential created by the solvent at these points is calculated and averaged over all the configurations. The solvent charges inside a certain cut-off radius around the solute are explicitly kept (their values divided by the total number of configurations considered) and, to avoid having too many charges, they are grouped together if the distance between them is less than a certain value. The electrostatic potential created by these explicit charges is subtracted from the one calculated before. In this way one obtains the contribution of the solvent molecules placed outside the cut-off radius. We named this contribution ASEP1. Finally, an external set of point charges, arranged in two spherical shells, is obtained such that the potential generated by them is the closest to the calculated ASEP1. The positions of these charges are set previously and their values are obtained through a least squares fit, so that

$$\sum_i \left(V_i' - V_i\right)^2, \tag{4}$$

$$V_i' = \sum_a \frac{q_a}{r_{ai}} \tag{5}$$

is minimized, where V_i is the ASEP1 calculated at point i and V_i' is the potential generated by the fitted charges at the same point, q_a is the value of the charge a and r_{ai} is the distance between charge a and point i. These point charges are then introduced into the solute Hamiltonian. In this way, the solvent is represented by two sets of point charges, one closer to the solute and keeping some information about the solvent structure and a second, farther from the solute, completing the fit to the ASEP. In the current implementation the molecular dynamics is performed with the MOLDY program [4] and the quantum calculation with the GAUSSIAN [5] or HONDO [6] packages.

In some cases it is desirable to make a calculation with polarizable solvent, for example to study electron transitions in the solute, which are too rapid to allow a reorganization of the solvent nuclei but permit the polarization of the electron clouds of the solvent molecules. In this case, instead of performing a full polarizable molecular dynamics—which would be too expensive—the polarization is calculated after the simulation is done, using the obtained solute–solvent configurations. In each configuration, the induced dipole moment is calculated for every solvent molecule,

$$\mu_i' = \alpha_i E_i, \tag{6}$$

where μ_i' is the induced dipole moment of molecule i, α_i its polarizability tensor, and E_i the *total* electric field in the center of mass of the molecule i, including contributions from the solute and the charges and induced dipoles of the other solvent molecules. Since the dipole moment on each molecule depends on the dipole moments on the rest of molecules, the calculation is done iteratively until convergence. Once the dipole moments have been obtained, the ASEP is calculated as above.

To perform a geometry optimization of the solute with the ASEP/MD method [7] we use the free energy gradient (FEG) method [8], in which the forces on the nuclei are

$$F(x) = -\frac{\partial G(x)}{\partial x} = -\left\langle \frac{\partial V(x)}{\partial x} \right\rangle \tag{7}$$

and the Hessian

$$H(x) = \frac{\partial F(x)}{\partial x} = \left\langle \frac{\partial^2 V(x)}{\partial x^2} \right\rangle - \beta\left(\left\langle F(x)^2 \right\rangle - \left\langle F(x) \right\rangle^2\right), \tag{8}$$

where $G(x)$ is the free energy associated with the solute geometry x, $V(r)$ the potential energy of the solute including the interaction with the solvent molecules, and $\beta = 1/RT$. The last term in Eq. (8) is related to the thermal fluctuation of the force and, since the Hessian is not essential for geometry optimization, we neglect it and assume

$$F(x) = -\frac{\partial \langle V(x) \rangle}{\partial x}, \tag{9}$$

$$H(x) = -\frac{\partial^2 \langle V(x) \rangle}{\partial x^2}, \tag{10}$$

where we replace the statistical average of the gradient by the gradient of the "average configuration" given by the ASEP, and similarly for the Hessian. Once the gradient and Hessian have been obtained, a standard geometry optimization (steepest descent, Newton-like methods, RFO) is performed in each cycle of the process.

3. Program description

The ASEP/MD program is written with readability in mind. Since most processor and memory consuming tasks are those performed by the external programs (quantum calculations and molecular dynamics simulations), code optimization was considered as only secondary. Once compiled and linked, the program is available as a single executable file. To launch ASEP/MD the user needs to provide a general input file and "template" files, which will be used as input for the external programs (obviously, these external programs must have been installed previously). The external programs currently supported are GAUSSIAN and HONDO for quantum calculations and MOLDY for molecular dynamics simulations.

Fig. 1 shows the simplified flow chart of the program. The most important steps are those in bold face: molecular dynamics simulation, calculation of the ASEP, and the quantum calculation.

In a typical run, the first thing the program does is to read the values of some variables from the input file supplied by the user. It then reads the system definition from the molecular dynamics template (nature and number of solvent and solute molecules, interaction potential, etc.). With this information, and if needed, the external quantum calculation program is called to perform a calculation of the isolated solute molecule, using the options given in the quantum calculation template (method, basis set, etc.). The output of this calculation is read to extract the energy, dipole moment and atomic charges of the solute *in vacuo*.

Now the main ASEP/MD cycle is carried out. The solute and solvent geometry and atomic charges are inserted into the molecular dynamics input file and the appropriate program is called to perform the simulation. Next, a set of solvent–solute configurations is read from the output of the molecular dynamics program and the average electrostatic potential created by the solvent over the solute (ASEP) is calculated from them, as well as several interaction energies. The ASEP is then approximated by a set of point charges. A new quantum calculation, this time with the external point charges representing the solvent, is performed. The output gives again the energy, dipole moment, and atomic charges of the solute. A check for convergence is made and, if it has not been reached, the solute charges are inserted again into the molecular dynamics input file and the main cycle starts anew.

So far, the program flow has been described when a basic calculation is requested. There are two other main types of calculations which require some modifications of this scheme: polarizable solvent and geometry optimization.

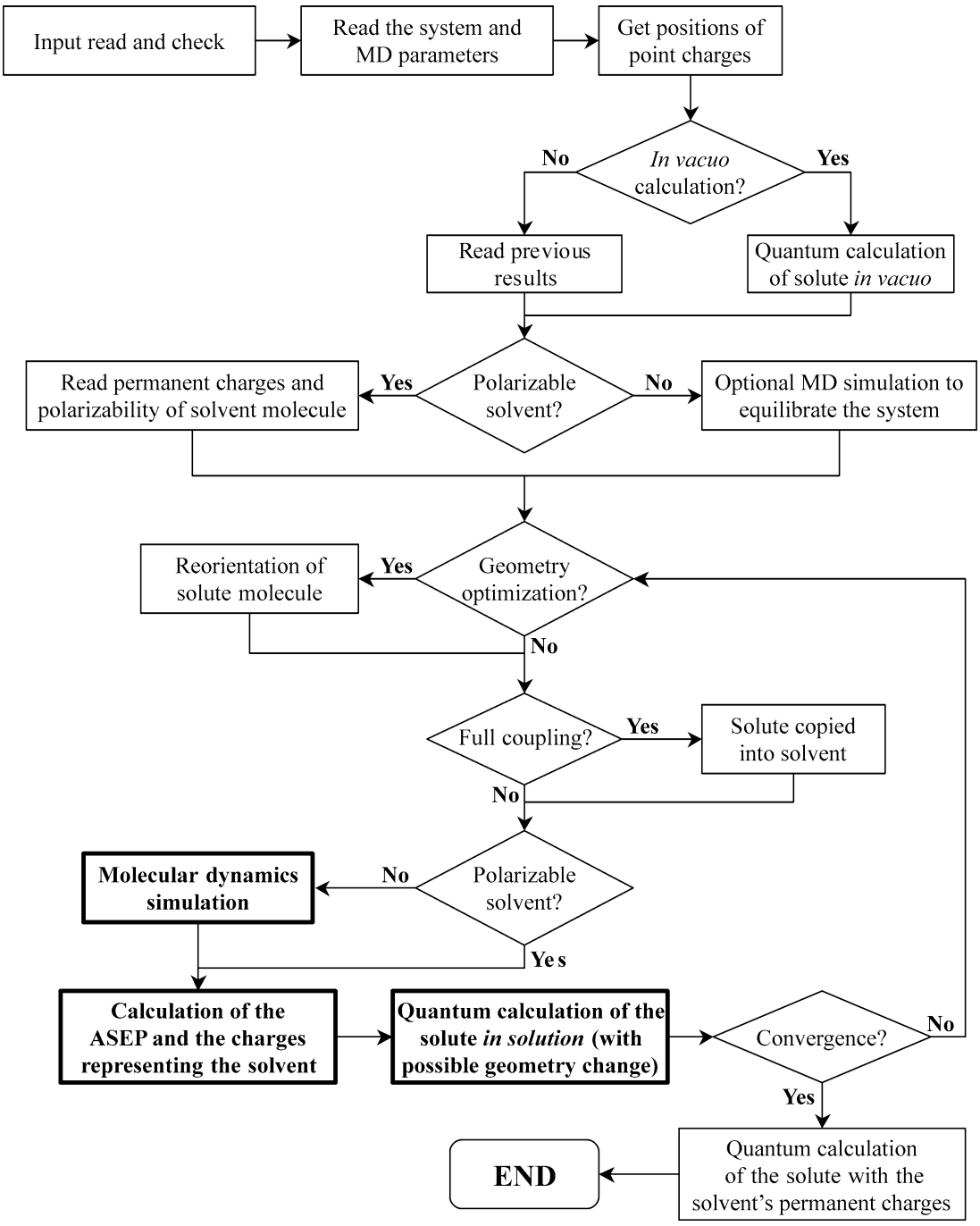

Fig. 1. Simplified flow chart of the ASEP/MD program.

3.1. Polarizable solvent

In the current implementation [2], the solvent polarization calculation is performed after an equilibrium sample of solvent–solute configurations has been obtained with a non-polarizable solvent. Once a normal run of the program, as described above, has finished, another calculation is carried out. This time no more molecular dynamics simulations are made, but the solvent configurations are read from the final simulation of the previous run. An additional input is given with the permanent *in vacuo* atomic charges of the solvent molecule and its polarizability tensor. Then, for each one of the configurations, all the solvent molecules are polarized by the solute charge distribution and by the other solvent molecules. The calculated ASEP now includes contributions from the permanent charges and the induced dipole moments of the solvent molecules. This potential is approximated by the two sets of point charges explained above and introduced into the quantum calculation. The charge distribution of the solute is used to obtain the solvent polarization again, and this process is repeated until convergence.

This whole calculation can performed for different solute electronic states, keeping always the same set of solvent configurations. In this way, it is possible to study the solvent effect on electronic transitions in the solute, which take place without appreciable reordering of the solvent nuclei, but with a major response of the solvent electrons.

3.2. Geometry optimization

The geometry optimization method implemented in this program is based on the Free Energy Gradient (FEG) method [8], in which the effective forces on the nuclei are taken as the average force over a set of equilibrium solvent–solute configurations. Once the molecular dynamics simulation has been made and the ASEP obtained, a geometry optimization calculation is performed using the rational function optimization (RFO) method [9]. In this way, a new solute geometry is obtained, which is introduced into the next molecular dynamics simulation. The process is repeated until convergence.

The gradient and Hessian, needed for the optimization process, are given by the quantum calculation program, but the appropriate van der Waals components have to be added. These components are calculated by the ASEP/MD program as an average over all the solute–solvent configurations considered.

There are a number of options affecting the optimization method used. In each cycle of the ASEP/MD process, the optimization may be complete (a stationary point is found and this geometry is used in the next cycle) or partial (a fixed number of optimization steps is performed, and while the geometry used in the next cycle is not a stationary point, it is closer to one). Other options are related to the algorithm of the optimization procedure.

4. Program organization

The ASEP/MD source code is divided into different files, each of them containing subroutines and definitions grouped according to the subject. The program uses modules to make variables and arrays globally available. The following is a short description of the contents of the different files:

Main.f90

This file contains the main program, global definitions, and subroutines related directly to the main program.

- Module definitions.
- `ASEP_MD`: This is the main program that controls the flow of the calculations.
- `LeerEntrada`: This subroutine reads the input provided by the user.
- `LeerContinuacion`: If a calculation is resumed, this subroutine reads the “continue” file.
- `EscribePrin`: Writes the output of the main program.

Execute.f90

The subroutines in this file are related to the external programs used by ASEP/MD to perform the quantum mechanics calculations and molecular dynamics simulations. Most of these subroutines are program-independent: they simply call the appropriate subroutine or process depending on the specific program used.

- `LeerSistema`: Calls another subroutine to read the system definition (solute, solvent, force field, etc.)
- `LeerAbinitio`: Calls another subroutine to read the output of a quantum calculation.
- `EjecutarAbInitio`: Runs the appropriate quantum calculation through a SYSTEM call.
- `EjecutarDinamica`: Runs the appropriate MD simulation through a SYSTEM call.
- `LeerConfig`: Extracts a solute–solvent configuration from the output of the MD program.

MoldySubroutines.f90

The subroutines included in this file deal with the input and output of the molecular dynamics program MOLDY.

- `LeerControlMoldy`: Reads the MOLDY control file for some necessary parameters.
- `LeerSistemaMoldy`: Reads the system definition from the MOLDY input file.
- `ModificarMoldy`: Modifies the MOLDY input files to run a new simulation.

GaussianSubroutines.f90

The subroutines included in this file deal with the input and output of the quantum calculation program GAUSSIAN.

- `ModificarGaussian`: Modifies the GAUSSIAN input file to perform a new quantum calculation.
- `LeerSalidaGaussian`: Reads the output of GAUSSIAN.

HondoSubroutines.f90

The subroutines included in this file deal with the input and output of the quantum calculation program HONDO.

- `ModificarHondo`: Modifies the HONDO input file to perform a new quantum calculation.
- `LeerSalidaHondo`: Reads the output of HONDO.

Calculations.f90

This file is the core of the program. It contains the subroutines that calculate the average solvent electrostatic potential and the point charges that reproduce it. It also contains some other auxiliary subroutines.

- `PosicionesCargas`: Gets the positions of the outer point charges representing the average solvent electrostatic potential.
- `CalcularCargas`: This subroutine calls other subroutines to calculate the values of the point charges that will represent the solvent in the quantum calculations.
- `PuntosPotencial`: Calculates a grid of points inside the solute in which the ASEP will be calculated.
- `CalcularPotencial`: Calculates the electrostatic potential and interaction energies for a given solute–solvent configuration.
- `ReducirCapa`: Reduces the number of point charges in the solvent representation.

- `CalcularCampo`: Calculates the fluctuations of the solvent electric field.
- `AjusteLagrange`: Fits the outer point charges by the Lagrange multipliers method.
- `ResolverLU`: Solves a system of linear equations by the LU factorization method.
- `GirarMolecula`: Rotates a molecule (solute or solvent) to orient it along its main inertia axes.

Optim.f90

This file contains all the subroutines needed to perform a geometry optimization.

- `OptimizarGeometria`: This is the main subroutine for geometry optimization. It controls the flow of the program in this section.
- `CalcularGradHessPot`: Calculates the contribution of the van der Waals potential to the gradient and Hessian.
- `BuscarMinimo`: Performs a linear search to find a minimum.
- `EnergPunto`: This is a function that returns the total energy for a given solute geometry.
- `ActualizarHessiana`: Updates the Hessian using an appropriate formula.
- `CalcularIncremento`: Calculates the increment to obtain the new solute geometry.
- `EscribeOptim`: Writes output related to the optimization process.

5. Usage

The compilation of this program is straightforward. Just feed the names of the different source files to the compiler. For example, with `pgf90` use (this is a single line):

```
[user@localhost]$ pgf90  Main.f90 Execute.f90  MoldySubroutines.f90
GaussianSubroutines.f90  HondoSubroutines.f90      Calculations.f90
Optim.f90 -o asepmd
```

No makefile or configure script is needed.

Once the program has been successfully compiled and linked, the resulting binary file can be executed. It accepts a single command-line option: the name of the global input file. This file contains the values of the different options affecting the calculation. The input is formatted as a namelist called Input, whose syntax may depend on the compiler and platform.

In addition to the global input file, the user must have access to suitable external programs for the quantum calculations and for the molecular dynamics simulations. For these external programs the user must supply template files from which ASEP/MD can build the appropriate input files as needed. Currently, there are some limitations on the options and units that can be used in the template files, but they can be easily circumvented by modifying the launching scripts or the parser subroutines. Future versions may eliminate these limitations.

As the calculation progresses, input and output files for the external programs will be created for each cycle of the process. All files will be created in the current working directory, so it is recommended to create a dedicated directory for each calculation and run the program from there.

5.1. Input

The following is a short description of the different options in the `Input` namelist. For each option, its type (string, integer, or float) is indicated:

- `DirDynamics`: String. Must point to the directory where the molecular dynamics binaries reside, without a trailing slash.
- `ProgAbInitio`: String. Location of a script or binary file that will run the quantum calculation program. It will be launched as '`ProgAbInitio input-file output-file`', so the user is advised to write a script that runs the program in this way. The value of this option should include the full path.
- `ProgDynamics`: String. Location of a script or executable that will run the molecular dynamics program. It will be launched as '`ProgDynamics input-file output-file`', so the user is advised to write a script that runs the program in this way. The value of this option should include the full path.
- `AbInitio`: String. This option specifies the program that will be used for quantum calculations. Valid values are 'GAUSS' for GAUSSIAN and 'HONDO' for HONDO.
- `Dynamics`: String. This option specifies the program that will be used for molecular dynamics simulations. Since the only molecular dynamics program supported in this version is MOLDY, the value must be 'MOLDY'
- `MainOutput`: String. Name of the file where the general output will be written in text format.
- `ContFile`: String. Name of the file where the resume information will be written to and/or read from.
- `AbInitioFile`: String. Name of the template file for the quantum calculation input. This file will be read and used as a template for each quantum calculation performed. The system definition (solute and external point charges) will be added by ASEP/MD as needed, so only such options as basis set and quantum method have to be specified here. This file should be in the format expected by the quantum calculation program used. It will not be overwritten.
- `AbInitioOutput`: String. This is the generic name for the quantum calculation output files. The name will be modified for each cycle by appending '.cic.#', where # is the cycle number.
- `DynamicsFile`: String. Name of the template file for the molecular dynamics input. This file will be read and used as a template for each molecular dynamics simulation performed. This file should be in the format expected by the molecular dynamics program used, it will not be overwritten.
- `DynamicsOutput`: String. This is the generic name for the molecular dynamics output files. The name will be modified for each cycle by appending '.cic.#', where # is the cycle number.
- `InitialDynamics`: String. Sometimes the user may want to perform an initial molecular dynamics calculation with different conditions (smaller time step, lower temperature, etc.) to equilibrate the system. In that case the name of the input file for this simulation is specified here. If this option is left blank, no such equilibration dynamics will be performed.
- `EnergyConv`: Float. Value of the convergence criterium for the energy. If the solute energy difference between two consecutive cycles is less than `EnergyConv`, the energy is considered as converged.
- `ChargesConv`: Float. Value of the convergence criterion for the solute atomic charges. If the maximum difference in the atomic charges of the solute between two consecutive cycles is less than `ChargesConv`, the solute atomic charges are considered to have converged.
- `MaxDiffCharges`: Float. Maximum allowed variation in the solute atomic charges from cycle to cycle. The charges' variation is damped if it is larger than `MaxDiffCharges`.
- `ChargesType`: String. Type of atomic charges calculated for the solute. Allowed values are: 'MUL' for Mulliken type, 'CHE' for CHELP calculated by the quantum program, 'POT' for charges fitted to the electrostatic potential by the ASEP/MD program.
- `MaxIter`: Integer. Maximum number of molecular dynamics + quantum calculation cycles. If this number is reached, the program terminates.
- `MaxIterPol`: Integer. Maximum number of solvent polarization cycles if a solvent polarization calculation is requested.
- `Start`: Integer. Initial cycle for the ASEP/MD calculation. If `Start` $= 0$, the program will start with a quantum calculation of the isolated solute. For other values, the file specified in `ContFile` should contain valid data for the previous cycles.

- `Coupling`: String. Type of coupling between solute and solvent. Allowed values are: 'NON' for no coupling, a single quantum calculation for the solute in solution will be performed; 'PAR' for partial coupling, several cycles are made until convergence is achieved or `MaxIter` is reached; 'TOT' for full coupling, the solute is copied into the solvent (this is only for pure liquids).
- `Polarization`: String. 'YES' if a solvent polarization calculation should be made; 'NO' otherwise.
- `DoDynamics`: String. The ASEP/MD cycles start with a molecular dynamics simulation, then a quantum calculation, and then another cycle begins. If a previous run is being resumed and the next molecular dynamics simulation has already been done, setting `DoDynamics` = 'NO' will skip the first molecular dynamics run and start with the following quantum calculation. This only skips the *first* molecular dynamics simulation needed. The following cycles will be made normally. If all simulations are to be performed, `DoDynamics` should be 'YES'.
- `NumConfig`: Integer. Number of solute–solvent configurations considered for obtaining the ASEP.
- `NumIniConfig`: Integer. First configuration to consider in obtaining the ASEP.
- `RadiiFactor`: Float. Factor by which the van der Waals radii of the solute atoms will be multiplied when getting the grid of points to calculate the ASEP.
- `GridDiv`: Integer. Number of points in each dimension where the ASEP will calculated. The maximum number of grid points is thus `GridDiv` [3], but it will typically be far less.
- `FirstShell`: String. If 'YES', the charges of the solvent molecules closest to the solute will be considered explicitly. If 'NO', they will not.
- `ShellRadius`: Float. Cut-off radius around the solute beyond which solvent molecules will not be considered explicitly.
- `Proximity`: Float. Minimum distance between point charges. If two charges are closer than this value, they will be added together.
- `PermCharges`: String. Name of the file with the solvent atomic charges and polarizability. This is only used when a solvent polarization calculation is performed.
- `IndDip`: String. Type of calculation for the induced dipole moments in the solvent molecules. 'NOR' triggers a one-pass calculation, 'CON' triggers a self-consistent polarization process until convergence.
- `MaxIterOpt`: Integer. Maximum number of optimization iterations within each ASEP/MD cycle. If `MaxIterOpt` = 0, optimization is turned off. If this number of iterations is reached, the optimization ends, but the ASEP/MD program continues normally.
- `GradientConv`: Float. Convergence criterion for the gradient in the optimization process. When the gradient's magnitude is less than this value, the optimization ends.
- `MinStep`: Float. Convergence criterion for the geometry variation in the optimization process. If the magnitude of the predicted change in geometry is less than this value, the optimization ends.
- `MaxStep`: Float. Maximum allowed change in the geometry between iterations of the optimization process. If the geometry change is larger, it will be damped accordingly.
- `CalcHessian`: Integer. Number of optimization iterations after which the Hessian will be calculated anew by the quantum calculation program; in other iterations it will just be updated by the appropriate formula. If `CalcHessian` = 1, the Hessian will be calculated in every iteration. If `CalcHessian` = 0, the Hessian will never be calculated; the identity matrix will be taken as the initial Hessian and updated in following iterations. If `CalcHessian` = 10, for example, the Hessian will be calculated again every 10 cycles.
- `LinearSearch`: String. 'YES' turns on the linear search algorithm to locate a minimum of the energy in the search direction. 'NO' turns it off.

The input file can be specified as an argument when running the program. If no extension is given, '.dat' will be assumed and appended to the name:

– `asepmd` will launch ASEP/MD and ask for an input file.
– `asepmd water` will launch ASEP/MD using `water.dat` as input.
– `asepmd water.input` will launch ASEP/MD using `water.input` as input.

5.2. Output

The output of this program, stored in `MainOutput`, will show, for each cycle of the ASEP/MD process, the values of the solute–solvent interaction energy, solvent dipole moment, and other useful energies. A short description of this output follows:

`MM ENERGIES`: Energies and properties calculated from the MD data only; the solute and the solvent are treated classically. The MM solute–solvent interaction energy is averaged over all the considered configurations and split into three components.

– `Charge(solute)-charge(solvent) interaction`: Interaction energy between the permanent charges of the solvent and the atomic charges of the solute.
– `Dipole(solvent)-charge(solvent) interaction`: Interaction energy between the induced dipoles of the solvent and the atomic charges of the remaining solvent molecules. This energy will only be non-zero if a polarizable solvent calculation has been made.
– `Van der Waals interaction`: Van der Waals interaction energy between solvent and solute, calculated by using the same potential (typically Lennard-Jones) given in the molecular dynamics input.
– `Electric field fluctuation`: Fluctuation of the electric field at the solute's center of mass, calculated as the difference between the mean squared field and the squared mean field.
– `Dipole-electric field interaction`: Solute dipole–electric field interaction energy. This is an oversimplification of the electrostatic solute–solvent interaction energy.

`QM ENERGIES`: Solute energies given by the quantum calculation program.

– `In vacuo energy`: Vacuum energy of the solute, this will be the same for all cycles, as it is calculated just once.
– `Total energy in solution`: Total solute energy in "solution", including the interaction energy between the solute and the solvent charges, but not including the self-energy of the external charges themselves.
– `Energy difference`: Energy difference between the two previous values.

`QM/MM ENERGIES`: Solute–solvent interaction energies where the solute is quantum and the solvent classical, as well as some energies derived from these. The solvent representation for these energies is the set of charges that was introduced into the quantum calculation.

– `Total interaction`: Total electrostatic solute–solvent interaction, calculated as the product of the solvent charges and the electrostatic potential generated by the solute over these charges.
– `Interaction with permanent charges`: Same as above, but considering only the permanent solvent charges (and not the contribution of the induced dipoles).
– `Interaction with induced charges`: Contribution of the solvent induced dipoles to the solute–solvent interaction energy, calculated as the difference between the two previous values.
– `Internal energy in solution`: Solute internal energy in solution, without the solute–solvent interaction energy.
– `Solute polarization energy`: Solute polarization or distortion energy: the difference between the internal energy in solution and the vacuum energy.

– `Electrostatic interaction energy`: Electrostatic interaction energy including the energy needed to polarize the solvent molecules.
– `Free energy variation`: Free energy variation between the previous cycle and the current one, calculated by the free energy perturbation method.

Some of this data, as well as short messages about what the program is doing, will be written to standard output, but this can be redirected to a file. For example:

`asepmd water` > `water.std` will launch ASEP/MD using `water.dat` as input and writing messages to `water.std`. The output file specified in `MainOutput` will still be generated.

Apart from errors in the input, the most common errors are those occurring in the external programs, so usually the first thing to do when the program stops unexpectedly is to check the output of the molecular dynamics or quantum calculation program. The last lines in `MainOutput` and in the standard output (or the file to which it has been redirected) can also help in finding the errors.

5.3. Example

This example shows a simple calculation for pure water. Input files are `water.dat`, `water.g` (input for GAUSSIAN), `water.ctr`, and `water.in` (input files for MOLDY). Output file, as specified by `MainOutput`, is `water.out`.

water.dat

```
&Input

DirDynamics = '/usr/local/MOLDY/'
ProgAbInitio = '/home/user/bin/gaussian'
ProgDynamics = '/home/user/bin/moldy'

AbInitio = 'GAUSSIAN'
Dynamics = 'MOLDY'
MainOutput = 'water.out'
ContFile = 'water.con'
AbInitioFile = 'water.g'
AbInitioOutput = 'water.ogau'
DynamicsFile = 'water.ctr'
DynamicsOutput = 'water.omol'

EnergyConv = 0.005
ChargesConv = 0.01
MaxDiffCharges = 10.0
ChargesType = 'CHELP'
MaxIter = 50
Start = 0
Coupling = 'TOTAL'
Polarization = 'NO'
DoDynamics = 'YES'

NumConfig = 500
NumIniConfig = 1
```

```
RadiiFactor = 0.7
GridDiv = 12
ShellRadius = 12.0
FirstShell = 'YES'
Proximity = 0.5

MaxIterOpt = 0
/
```

water.g

```
#P HF/6-311G

Water
 0 1
O  0.0000000 0.0000000  0.0000000
H  0.7569503 0.0000000 -0.5858822
H -0.7569503 0.0000000 -0.5858822
```

water.ctr (time-unit must be set to 4.8888213e–14)

```
title = Water
nsteps = 150000
step = 0.0005
sys-spec-file = water.in
lattice-start = 1
save-file = hf.save
text-mode-save = 1
density = 1.000
scale-interval = 1
const-temp = 1
temperature = 273
roll-interval = 1000
print-interval = 1000
begin-average= 50001
average-interval= 100000
begin-rdf = 50001
rdf-interval = 10
rdf-out = 100000
dump-file = water.dump
begin-dump = 50001
dump-interval = 100
dump-level = 3
ndumps = 500
time-unit = 4.8888213e-14
end
```

water.in

```
#Water

SOLVENT 214
1  0.0000000 0.0000000  0.0000000 16.0 -0.834   O(S)
```

```
2  0.7569503 0.0000000 -0.5858822 1.0   0.417   H(S)
2 -0.7569503 0.0000000 -0.5858822

SOLUTE 1
3  0.0000000 0.0000000  0.0000000 16.0 -0.834   O
4  0.7569503 0.0000000 -0.5858822  1.0  0.417   H
4 -0.7569503 0.0000000 -0.5858822

end

lennard-jones

1 1 0.60829 3.150656
1 3 0.60829 3.150656
3 3 0.60829 3.150656

end
```

water.out (last section)

```
Cycle 6

        Energy=   -76.18251789 Eh      Dipole moment= 3.8493 D

  Diff. energy=    -0.00165296 Eh Max. diff. charges= 0.00783200 e-

-------------------------------------------------------------------------------

| MM ENERGIES (Average of N configurations)                                   |

| **Charge(solute)-charge(solvent) interaction =      -0.20351731 Eh       |

|     U(q,s) = SUM[s*V(q)]                          -127.70905218 Kcal/mol |

| **Dipole(solvent)-charge(solvent) interaction =      0.00000000 Eh       |

|     U(p,q) = -SUM[p*E(q)]                            0.00000000 Kcal/mol |

| **Van der Waals interaction =                        0.04457549 Eh       |

|     U(vdw)                                          27.97154632 Kcal/mol |

| **Electric field fluctuation =        0.61581E-4, -.18949E-5, 0.83096E-5 |

| (tensor form, a.u.)                   -.18949E-5, 0.38376E-4, -.46923E-5 |

| <E\2> - <E>^2                         0.83096E-5, -.46923E-5, 0.97364E-4 |

| **Dipole-electric field interaction =                    -0.15982797 Eh |

|                                                  -100.29357736 Kcal/mol |
-------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------
|                            QM ENERGIES                                |
| **In vacuo energy =                          -76.00939257 Eh        |
|     E(0) = <Y(0)|H(0)|Y(0)>                -47696.61901494 Kcal/mol |
| **Total energy in solution =                 -76.18251789 Eh        |
|     E = <Y|H(0)+V|Y>                       -47805.25679947 Kcal/mol |
| **Energy difference =                         -0.17312531 Eh        |
|     D(E) = E - E(0)                         -108.63778453 Kcal/mol |
-------------------------------------------------------------------------

-------------------------------------------------------------------------
|                            QM/MM ENERGIES                             |
| **Total interaction =                          0.20160412 Eh        |
|     U(qt,r) = SUM[qt*V(r)]                   -126.50850884 Kcal/mol |
| **Interaction with permanent charges =         -0.20160412 Eh        |
|     U(q,r) = SUM[q*V(r)]                     -126.50850884 Kcal/mol |
| **Interaction with induced charges =           0.00000000 Eh        |
|     U(p,r) = SUM[(qt-q)*V(r)]                  0.00000000 Kcal/mol |
| **Internal energy in solution =              -75.98091377 Eh        |
|     E(int) = <Y|H(0)|Y> = E - U(qt,r)      -47678.74829063 Kcal/mol |
| **Solute polarization energy =                 0.02847881 Eh        |
|     E(pol)sol = E - E(0) - U(qt,r)           17.87072431 Kcal/mol |
| **Electrostatic interaction energy =           -0.20160412 Eh        |
|     U(elec) = U(qt,r) + 1/2*U(p,q) - 1/2*U(p,r)  -126.50850884 Kcal/mol |
| **Free energy variation =                      0.00171818 Eh        |
|     D(G), D(A)                                 1.07817397 Kcal/mol |
-------------------------------------------------------------------------

Convergence reached.
```

The scripts /home/user/bin/gaussian and /home/user/bin/moldy are used to give the required syntax. Their contents are:

/home/user/bin/gaussian

(the standard g98 script modified so that the last line reads):

```
g98 < $argv[1] > $argv[2]
```

/home/user/bin/moldy

(runs MOLDY in parallel using mpirun)

```
#!/bin/bash

/usr/bin/mpirun -v -np 3 -nolocal -stdin $1 /usr/local/MOLDY/moldy >& $2
```

Acknowledgements

This research was sponsored by the Dirección General de Investigación Científica y Técnica (BQU2000-0243) and by the Consejería de Educación y Juventud de la Junta de Extremadura (Project 2PR01A010).

References

[1] A. Warshel, M. Levitt, J. Mol. Biol. 103 (1976) 227;
U.C. Singh, P.A. Kollman, J. Comput. Chem. 7 (1986) 718;
M.J. Field, P.A. Bash, M. Karplus, J. Comput. Chem. 11 (1990) 700;
V. Luzhkov, A. Warshel, J. Comput. Chem. 13 (1992) 199;
J. Gao, in: K.B. Lipkowitz, D.B. Boyd (Eds.), Reviews in Computational Chemistry, Vol. 7, VCH Publishers, New York, 1996, p. 119.
[2] M.L. Sánchez, M.A. Aguilar, F.J. Olivares del Valle, J. Comput. Chem. 18 (1997) 313;
M.L. Sánchez, M.E. Martín, M.A. Aguilar, F.J. Olivares del Valle, Chem. Phys. Lett. 310 (1999) 195;
M.E. Martín, M.L. Sánchez, J. Olivares del Valle, M.A. Aguilar, J. Chem. Phys. 113 (2000) 6308;
M.E. Martín, M.L. Sánchez, M.A. Aguilar, F.J. Olivares del Valle, J. Mol. Structure (THEOCHEM) 537 (2001) 213;
M.L. Sánchez, M.E. Martín, M.A. Aguilar, F.J. Olivares del Valle, J. Comput. Chem. 21 (2000) 705;
M.E. Martín, M.L. Sánchez, F.J. Olivares del Valle, M.A. Aguilar, J. Chem. Phys. 116 (2002) 1613.
[3] M.L. Sánchez, M.E. Martín, I.F. Galván, F.J. Olivares del Valle, M.A. Aguilar, J. Phys. Chem. B 106 (2002) 4813.
[4] K. Refson, Comput. Phys. Comm. 126 (2000) 309.
[5] M.J. Frisch, G.W. Trucks, H.B. Schlegel, G.E. Scuseria, M.A. Robb, J.R. Cheeseman, V.G. Zakrzewski, J.A. Montgomery Jr., R.E. Stratmann, J.C. Burant, S. Dapprich, J.M. Millam, A.D. Daniels, K.N. Kudin, M.C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G.A. Petersson, P.Y. Ayala, Q. Cui, K. Morokuma, D.K. Malick, A.D. Rabuck, K. Raghavachari, J.B. Foresman, J. Cioslowski, J.V. Ortiz, A.G. Baboul, B.B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R.L. Martin, D.J. Fox, T. Keith, M.A. Al-Laham, C.Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P.M.W. Gill, B. Johnson, W. Chen, M.W. Wong, J.L. Andres, C. Gonzalez, M. Head-Gordon, E.S. Replogle, J.A. Pople, Gaussian 98, Gaussian, Inc., Pittsburgh, PA, 1998.
[6] M. Dupuis, A. Marquez, E.R. Davidson, in: HONDO 2000, based on HONDO 95.3; Quantum Chemistry Program Exchange (QCPE), Indiana University, Bloomington, 2000.
[7] I. Fdez Galván, M.L. Sánchez, M.E. Martín, F.J. Olivares del Valle, M.A. Aguilar, J. Chem. Phys. 118 (2003) 225.
[8] N. Okuyama-Yoshida, M. Nagaoka, T. Yamabe, Internat. J. Quantum Chem. 70 (1998) 95;
N. Okuyama-Yoshida, K. Kataoka, M. Nagaoka, T. Yamabe, J. Chem. Phys. 113 (2000) 3519;
H. Hirao, Y. Nagae, M. Nagaoka, Chem. Phys. Lett. 348 (2001) 350.
[9] A. Banerjee, N. Adams, J. Simons, R. Shepard, J. Phys. Chem. 89 (1985) 52;
J. Baker, J. Comput. Chem. 7 (1986) 385;
X. Prat-Resina, M. García-Viloca, G. Monard, A. González-Lafont, J.M. Lluch, J.M. Bofill, J.M. Anglada, Theoret. Chem. Accounts 107 (2002) 147.