Environ Documentation

Oliviero Andreussi

Oct 06, 2021

Contents:

Environ is a computational library aimed at introducing environment effects to atomistic first-principles simulations, in particular for applications in surface science and materials design. It is a Quantum-ESPRESSO module currently compatible with QE-6.3 and up.

CHAPTER 1

Installation instructions

The following sections cover downloading, configuring, and installing Quantum-ESPRESSO (QE) and Environ.

1.1 Preliminary steps

1.1.1 Obtaining source files

QE

1. clone the repository and checkout *qe-6.3* (or later)

```
git clone https://gitlab.com/QEF/q-e
git checkout qe-6.3
```

or

1. download a qe-6.3 archive (or later) from the QE releases page
2. unpack the archive

```
tar zxvf qe-X.Y.Z.tar.gz
```

or, if your **tar** doesn't recognize the *z* flag

```
gunzip -c qe-X.Y.Z.tar.gz | tar xvf -
```

Environ

1. clone the Environ repository into the QE root directory

```
git clone https://github.com/environ-developers/Environ.git
```

or

1. download an archive from the Environ releases page
2. unpack the archive inside the QE root directory

```
tar zxvf qe-X.Y.Z.tar.gz
```

or, if your **tar** doesn't recognize the *z* flag

```
gunzip -c qe-X.Y.Z.tar.gz | tar xvf -
```

1.1.2 Configuration

QE

Quantum-ESPRESSO uses a **configure** script to detect compilers and libraries necessary for compilation. To configure the environment, run the following command from the QE root directory

```
./configure
```

For additional flags for fine-tuning the configuration process, check out the QE docs.

Environ *v2.0* and up

The **configure** script was adopted in the recent Environ *v2.0* release. To configure Environ, run the same command from the Environ root directory

```
./configure
```

making sure to use the same compiler flags and libraries.

1.2 Pre-compilation

Note: Optional for Environ *v2.0* and up.

The installation process of Environ *v2.0* (or later) will automatically pre-compile QE and will abort if any issues arise. However, if using an older version of Environ, QE pre-compilation is required at this stage. You may do so from the QE root directory with

```
make <package>
```

where *package* is any of the packages supported by Environ: pw, cp, tddfpt, or xspectra. If issues arise, solutions may be found in the QE docs or on the QE forum.

Note: If QE is manually pre-compiled, Environ *v2.0* (or later) re-compilation of QE will proceed quickly.

1.3 Environ installation

The process has been modified greatly in the recent *v2.0* release, both for installing and uninstalling Environ. Please follow the links matching your working version.

1.3.1 Environ *v2.0* and up

Note: Run the following commands from the Environ root directory.

Installing

1. install Environ

```
make install
```

    1. select the QE package you wish to work with
    2. select the number of cores for parallel compilation (default = 1)

Note: If issues arise, try compiling in serial.

Uninstalling

To uninstall Environ, run

```
make uninstall
```

1. enter **y** to proceed
2. when prompted for, enter **y** to uninstall QE (optional)

1.3.2 Environ *v1.0* up to *v2.0*

Note: Run the following commands from the QE root directory.

Installing

1. run the script **addonpatch.sh** with the -patch option

```
./install/addsonpatch.sh Environ Environ/src Modules -patch
```

2. patch the necessary QE files with

```
./Environ/patches/environpatch.sh -patch
```

3. update QE dependencies

```
./install/makedeps.sh
```

4. re-compile the chosen QE package (pw, cp, tddfpt, or xspectra)

```
make <package>
```

Uninstalling

1. run the QE script addsonpatch.sh with the -revert option

```
./install/addsonpatch.sh Environ Environ/src Modules -revert
```

2. run the Environ installation script with the -revert option

```
./Environ/patches/environpatch.sh -revert
```

3. run the QE script to regenerate modules' dependencies

```
./install/makedeps.sh
```

4. remove objects, modules, and exectuables

```
make clean
```

Known Issues

In Environ *v1.0*, there is an issue with the installation procedure for codes different from pw.x. The problem seems to depend on the compiler, but it is present in the most common Intel Fortran compiler. The solution for this problem requires some work, which is described here, and will be fixed in future releases.

There is some circular dependencies between Environ modules and PW modules, but since the linkers only look for the dependencies written at the time of the Environ installation, they will not be able to function as intended.

To fix this, one can edit the Makefile manually, to add these PW dependencies, so that instead of:

```
PWOBJS = some-path/PW/src/libpw.a
QEMODS = some-path/Modules/libquemod.a some-more-libraries.a
```

we have:

```
PWOBJS = some-path/PW/src/libpw.a
QEMODS = some-path/Modules/libquemod.a some-more-libraries.a some-path/PW/src/libpw.a
```

This may be necessary for the following files:

```
PHonon/FD/Makefile
PHonon/Gamma/Makefile
PHonon/PH/Makefile
PWCOND/src/Makefile
TDDFPT/src/Makefile
XSpectra/src/Makefile
GWW/bse/Makefile
GWW/head/Makefile
GWW/pw4gww/Makefile
```

There is an exception, for the CPV Makefile, instead of this:

```
QEMODS=../../Modules/libqemod.a
```

do this:

```
QEMODS=../../Modules/libqemod.a libcp.a
```

CHAPTER 2

User Tutorial

This user tutorial will run through the examples that exist in the Environ module. These examples are set up to run via executing bash scripts. The context for these tests are given in the appropriate README file. The bash scripts contain explanations behind settings. A more verbose explanation, along with a guide on setting up jobs from scratch can be seen in this tutorial. It is assumed that the reader has some familiarity of Quantum Espresso and the setup of a pw input file. If this is not the case, one can refer to a brief overview: *QE PW Input File*.

2.1 Example: Solvation Energy (SCCS)

This first example will demonstrate using the pw.x executable with Environ to calculate the solvation energy and other related solvation quantities for a water molecule in a water solvent. A more in-depth explanation of the demonstration can be found in the README. The overall idea is as follows: solvation energy is the difference in energy between a solute in vacuum and in solvent. By default, the pw.x program calculates the energy of a system in vacuum. The Environ module allows us to account for additional effects or correction factors. It also allows us to add some non-vacuum environment, in this case, a water solvent. Hence by running pw.x twice with different Environ parameters, one can calculate the energy of the system in vacuum and in water, thus obtaining the solvation energy.

In general, execution of pw.x requires a correctly formatted input file with all the descriptive properties of the desired system. With the Environ addon, an additional input file is required that contains simulation parameters for the environment. Unlike the pw input file that can be arbitrarily named and then specified on execution of the program, the Environ file must be named 'environ.in'. To enable the environ addon on a pw calculation, the `--environ` modifier should be added. Hence the command, `./pw.x --environ < example.in` would feed in a pw input file named example.in into pw.x and run a serial calculation via whichever FORTRAN compiler Quantum ESPRESSO has been configured with (as with any program, call `pw.x` from its actual position, or add it to the PATH). Since the `--environ` modifier is added, `pw.x` now expects an environ.in file. Failure to do so will result in an error.

The bash script run_example.sh generates all the necessary files described above and executes multiple pw calculations successively. Due to the constraint of the environ.in name, a change in environment should require files to be renamed on the fly (something that is taken care of here in the bash script). Alternatively one can set up multiple runs to execute in separate directories thus avoiding this issue entirely.

The general format of the environ.in file can be seen starting line 154 of the bash script. Alternatively one can run the bash script and view the generated environ files in the results folder (these have the environ prefix and

the .in suffix). Refer to the input documentation (which can be viewed on the browser from the local html file, Doc/INPUT_Environ.html) for in-depth information on any of the variables as well as the expected structure. The general structure splits parameters into a number of keywords, &ENVIRON, &BOUNDARY, and &ELECTROSTATIC. This file has fortran-like formatting and as such, '!' can prepend comments.

```
! Some comment
&ENVIRON
   verbose = 0
```

Environ is designed to be modular and as a result, many of the parameters are interchangeable. Each parameter belongs to a corresponding keyword and therefore should be placed accordingly. The parameters used in the program are also helpfully described in the bash file. By default, the verbosity parameter (verbose) is set to zero, which limits output to the standard pw output location (which is to the terminal, and therefore is often piped into an output file).

Some helpful environment specific quantities can also be printed out. These are typically used on the development side for debugging. The option `verbose=1` (or any higher integer value) will output these in a file named environ.debug. The option `verbose=2` will additionally output gaussian cube files that contain the density objects of important quantities. Specifying this option will impact performance due to the heavy I/O operations, and for the majority of simulations, a verbosity of zero ought to be sufficient, which is the default.

```
&ENVIRON
   environ_thr = 1.d-1
```

The environ threshold specifies when to start environ contributions to the SCF iterative step. The default value is suitable primarily for small systems. Note that since these input files follow fortran convention, double precision is notated as above.

```
&ENVIRON
   environ_type = 'vacuum'
```

The environ type simplifies the requirement of some of the physical parameters (such as pressure, surface tension, and static permittivity), which arise depending on the interface model applied. One can manually add these, but these have been tuned to optimal values, and therefore it is generally preferable to use the preset values by setting this parameter accordingly (vacuum, water, water-anions, water-cations).

Note: This option is equivalent to

```
&ENVIRON
   environ_type = 'input'
   env_surface_tension = 0
   env_pressure = 0
   env_static_permittivity = 1
```

```
&ELECTROSTATIC
   pbc_correction = 'parabolic'
   pbc_dim = 0
```

By design, pw assumes a simulation cell with 3D periodicity. This can be overcome by setting a simulation cell with enough space and enabling the pbc_correction parameter (which in turn requires the env_electrostatic parameter to be set as true). This correction is an approximation but should be sufficient for most simulation runs. For more accurate results, one may want to refer to martyna-tuckerman (see Q-E input documentation under 'assume-isolated'), which does require a larger simulation cell (and therefore more physical memory). For simulations that require one to retain periodicity in 1 or 2 dimensions (for example, a diffuse-layer simulation), the pbc_dim parameter can be set appropriately. More details on this quadratic correction method can be found in the publication[1].

[1]

```
&ELECTROSTATIC
   tol = 1.d-11
   mix = 0.6
```

The electrostatic calculation has its own accuracy that, for the most part, should be set manually in the input file (since the value itself is reliant on the solver picked).

On running a pw calculation with Environ while reading the output, a few differences are apparent. Firstly before the SCF loop, Environ Module will be printed, followed by a brief description of the module, and some of the settings (as defined in the input file). On each step of the SCF iteration, there are additional energy corrections printed that are a result of the Environ module (the electrostatic embedding term and the correction to one-el term). These corrections should be non-zero once the threshold specified in Environ's input file is met. Finally one can inspect the computation cost of the environ processes at the end of the file.

```
&BOUNDARY
/
```

In this example, the boundary is left empty. By default, Environ will use the SCCS model to define the interface. This model uses the electrostatic density to define the separation between solute and solvent.

```
&ELECTROSTATIC
   solver = 'direct'
   auxiliary = 'none'
```

A number of different solvers can be employed to calculate for electrostatic potential. Typically these default to sensible values according to the setup, however, one can manually set these along with the auxiliary parameter (in the event of say, polarization charge).

For small investigations such as this example, where only 3 simulations are run, it is sufficient to process the result directly from the command line. When scaling up to a higher number of simulations, bash or python scripting can be exploited to automate the process of calculating the solvation energy. Suppose the user is in the directory containing the newly created PW output files (in the results folder), the output files of Quantum ESPRESSO are designed for convenient extraction of commonly used results via grep. Calling `grep ! *.out` will return the total energy values. Since this is a PW relax calculation, the energy is calculated multiple times, while updating the positions of the ions depending on the calculated forces in the system. One should expect to get a result like:

```
h2o_vacuum_direct.out:!    total energy              =     -34.26804813 Ry
h2o_vacuum_direct.out:!    total energy              =     -34.26815510 Ry
h2o_vacuum_direct.out:!    total energy              =     -34.26825783 Ry
h2o_vacuum_direct.out:!    total energy              =     -34.26826238 Ry
h2o_water_cg.out:!    total energy              =     -34.28845991 Ry
h2o_water_cg.out:!    total energy              =     -34.28851745 Ry
h2o_water_cg.out:!    total energy              =     -34.28858148 Ry
h2o_water_iterative.out:!    total energy              =     -34.28845478 Ry
h2o_water_iterative.out:!    total energy              =     -34.28851656 Ry
h2o_water_iterative.out:!    total energy              =     -34.28857886 Ry
```

Note that the values may not match exactly due to the compiler used. It is useful to bear in mind when analysing energies in Rydberg, differences of E-2 are typical for solvation energies, whereas differences of less than 3E-3 are less than chemical accuracy. We see that the final energy of the system depends on the solver used, but the difference is 3E-5, well below chemical accuracy and therefore not of much concern for regular applications. Hence the solvation energy can be calculated (as the difference between the solvation and vacuum energies) as -6.374kcal/mol (or 2.03E-2Ry) via the conjugative gradient solver, and -6.373kcal/mol (or 2.03E-2Ry) via the iterative solver. This compares closely to the experimental solvation energy of water (-6.3kcal/mol[2]).

I. Dabo et al., Phys. Rev. B 77, 115139 (2008)

2

If the user is just interested in the final energy in a relax calculation, a command like `grep Final *.out` will achieve this. This example conveniently automates the calculation of these values via bash (along with other helpful energy values), and prints them out into `results.txt` for reference.

2.2 Example: Solvation Energy (SSCS)

In the previous example, one detail that was not explained was the choice of interface model. As stated previously, Environ is designed to be as modular as is reasonable, which means the user is able to define settings that enable different models, solvers, and corrections to different subsystems. Previously the interface model, that defines the separation of the quantum mechanical system (set in the PW input file), with the solvent environment, was set to SCCS. The SCCS model sets a smooth boundary according to the electronic density at any particular point. This function is in fact a piece-wise function that is chosen to be easily differentiable[1].

An alternative model that Environ has implemented is the soft-sphere (SSCS) model[2]. This model sets a smooth boundary according to the ionic positions. Interlocking spheres are taken around each ion, whose radii are by default consistent with the UFF force fields[3] (note that in the paper, there is an exception for nitrogen, which is based off Bondi's radius for nitrogen[4], an option which is not a default in Environ) in a nature similar to PCM, only these are smooth functions (spherical error functions).

This example repeats the previous setup, instead with a soft-spheres implementation. Since this is a boundary parameter, one can see on execution of the bash script that the environ input files here differ, now possessing parameters related to the &BOUNDARY keyword.

```
&BOUNDARY
  solvent_mode='ionic'
```

Simply by setting the solvent_mode parameter one can switch from SCCS to soft-sphere model, environ_type presets can take care of the required (tuned) parameters associated with the model. Note that the solver (and auxiliary) parameters are not required for the SSCS.

Calculating the solvation energy is exactly the same procedure as before, and results in a value of -5.786kcal/mol, within chemical accuracy of the experimental results, as seen in the first example.

2.3 Example: Isolated Systems

This example shows how to use pw.x to simulate isolated charge systems in vacuum or immersed in a continuum dielectric constant. Note that this can be thought of as an alternative to Environ's own periodic correction as seen in Example 1. Since this setting resides in the pw input file, both input files should be checked for consistence. In the first example we removed all periodicity from the system with a correction that results in a fairly good approximation.

A. V. Marenich et al., Minnesota Solvation Database - version 2012; University of Minnesota: Minneapolis

1

O. Andreussi, I. Dabo, N. Marzari, J. Chem. Phys. 136, 064102 (2012)

2

G. Fisicaro et al., J. Chem. Comput. 2017, 13, 8, 3829-3845

3

A. K. Rappe et al., J. Am. Chem. Soc. (1992), 114 (25) 114 10024-35

4

A. Bondi, J. Phys. Chem. (1964), 68 (3) 68 441-51

```
&ELECTROSTATIC
   pbc_correction = 'parabolic'
   pbc_dim = 0
```

In this example, we demonstrate a system without periodicity, thus removing the parameters stated before and keeping to the defaults. We also demonstrate an alternative to the pbc correction, which is the martyna-tuckerman[1] correction. This correction is seen as a more accurate method, at the cost of a larger required cellsize. For reliable results, a cell length of twice the molecule length is recommended, in any direction. This can be added into the pw input file, under the SYSTEM keyword.

```
&SYSTEM
   assume_isolated = 'martyna-tuckerman'
```

The tolerance is different from previous examples. This value is chosen to facilitate the self-consistent process and will directly affect the speed and accuracy of the convergence. One should set this according to the type of simulation being run, these values in the examples serving as guidelines for that decision.

Clearly for larger systems, the parabolic correction is a more feasible choice for imposing isolation. This is a real-space quadratic correction of the electrostatic potential[2], which has been shown to provide energy accuracy improvements of almost 1 order of magnitude in comparison with the alternative PCC and GCC schemes for large cell sizes. This correction scheme implemented in Environ is limited to 0D and 2D systems (which account for the majority of systems that Environ typically works with).

2.4 Example: Two-Dimensional Systems

This example shows how to use pw.x to model two-dimensional periodic systems in contact with a continuum solvent. This is particurly useful for diffuse layer systems and other surface investigations. For this calculation, a Pt (111) slab is chosen with a CO molecule adsorbed on the surface. By running the bash file and inspecting the environ input files, one will notice that most of the input remains unchanged. The SCCS (self-consistent charge solvation) model is chosen for the solvent by default and the solvent type is set manually, for example for vacuum,

```
&ENVIRON
   environ_type = 'input'
   env_static_permittivity = 1
   env_surface_tension = 0.D0
   env_pressure = 0.D0
```

The change is in the pbc_dim parameter. This needs to be set accordingly and the axis should be subsequently chosen. This axis parameter determines the direction of the periodicity (that is the plane normal to the axis will be parallel to the surface or slab). This parameter expects an integer value, 1 for x, 2 for y, and 3 for z.

```
&ELECTROSTATIC
   pbc_dim = 2
   pbc_axis = 3
```

The tolerance is also different from previous examples. This parameter is typically set in order to optimize the ability for the self-consistent process to converge, and should be picked appropriately depending on the system. These examples serve as good guidelines for each type of simulation.

Along with the electrostatic solvation energy, which calculated similarly to Example 1, where one takes the difference between the total energy in vacuum with total energy in electrolyte solvent, this example calculates the Fermi energy

[1] G. J. Martyna and M. E. Tuckerman, J. Chem. Phys. 110, 2810 (1999)

[2] I. Dabo et al. Phys. Rev. B 77, 115139 (2008)

both in vacuum and in solution. Unlike other energy terms, the printed total Fermi energy is decomposed into terms, one from QE and a correction (delta energy) from Environ. Hence, the user should sum these terms during post-processing in order to get the Fermi energy of the system (this is done for the user in this example by the bash script).

2.5 Example: Charge Distributions

This example follows the previous, now adding a fixed, planar and smooth distribution of charge (specifically a helmholtz plane). The environ input file for this kind of system is a little more complicated.

First is the use of the keyword solvent_mode = 'full'. For this calculation, this choice is mandatory due to the Pt valence density, which results in a hole close to the ionic position, due to the missing core electrons. This option adds an additional charge density (gaussians) at the location of the ions, that represent the core electrons and the nuclei. This results in a better description of the interface between vacuum and solvent. This choice is useful for particular types of atoms where this effect can occur, for example halogens and transition metals.

```
&BOUNDARY
   solvent_mode = 'full'
```

To represent the helmholtz plane, it is necessary to specify the number of external charges in the ENVIRON keyword and then specify the the charges themselves in a block labelled EXTERNAL_CHARGES (at the end of the input file). Notice how this structure is similar to how the pw input files handle atomic positions.

```
EXTERNAL_CHARGES (bohr)
-0.5 0. 0. 25.697 1.0 2 3
-0.5 0. 0. -10.303 1.0 2 3
```

Each line specifies a charge, and possesses 7 values separated by spaces. In order these describe the charge (in units of electron charge), x position, y position, z position, spread, dimensionality, and axis. In this example we wish to define charge planes parallel to the Pt slab. Since we are defining two-dimensional objects that lie perpendicular to the z-axis, the x and y positions are irrelevent and are thus set to 0.0. The charge functions are represented by gaussians, and thus the parameters are simply position and spread. Finally the charges can represent basic shapes depending on the dimensionality defined. For a slab, we specify a dimensionality of 2. For the direction of the slab, we specify 3, which represents the z-axis. This convention is consistent with the boundary correction options (explained in the previous example) that define the system to be periodic in two-dimensions as opposed to the three by default.

We set this system up so that there exists two countercharge planes on either side of the Pt slab. Since the system is periodic, imposing symmetry in this manner allows the simulation to perform better and is generally advised. Since there are now two planes of half charge, the resultant electric field is half what it would have otherwise been. For detailed information on these planar charge setups, see[1],[2], or *Explicit Charge*.

2.6 Example: Electrolyte Solutions

This example, like the previous two, models a 2-D metallic structure, this time, an Ag (100) slab, in water solution. The slab is charged and an electrolyte solution is added. This solution can be thought of as a solvent (whose characteristics we have now implemented a number of times), and electrolyte ions, that are defined based off a set of parameters. The parameters describing the ions are placed in the ENVIRON keyword

[1] C. L. Fu and K. M. Ho, Phys. Rev. Lett. 63, 1617 (1989)

[2] F. Nattino et al., J. Chem. Phys. 041722 (2019)

```
&ENVIRON
   env_electrolyte_ntyp = 2
   zion(1) = 1
   zion(2) = -1
   cion(1) = 1.0
   cion(2) = 1.0
   cionmax = 10.0
```

The env_electrolyte_ntyp parameter sets the number of ionic species that are to be defined, and for each species the charge (zion) and the concentration (cion) are to be specified. Finally the maximum concentration can be set (respresenting the maximum concentration of ions at any point).

A number of electrolyte models can be implemented in Environ. This example iterates through the models based the numerical method, which utilizes the Poisson-Boltzmann equation or some variant. It also includes some analytical analogues that, rather than solve the Poisson-Boltzmann equation, add a correction to the potential. By default, the Poisson-Boltzmann model is set. By specifying the parameter

```
&ENVIRON
   electrolyte_linearized = .true.
```

the model is changed to the linear equivalent. Refer to the publications[1],[2] for more details on these models. Some pbc correction (as explained in example 1) is necessary here, since calculations involving the electrolyte require open boundary conditions. In the example, this is set to parabolic, referring to the numerical approach, whereas a change to the gcs correction,

```
&ELECTROSTATIC
   pbc_correction = 'gcs'
```

will switch to the analytic approach to solving the relevant Poisson-Boltzmann equation. Note that this does not include the size-modified equation. The page on diffuse-layer models will describe each of the models and how they differ in more detail. In particular, the required parameters for each models do vary slightly. For the linearized Poisson-Boltzmann model and the Poisson-Boltzmann model, the charge and concentrations need to be specified. The additional max concentration (cionmax) parameter should be included when choosing the modified Poisson-Boltzmann model. Notice that the model need not be specified explicitly and is instead inferred by the parameters supplied to the input. The physical models all require at least the concentrations and the charges of the ions. By setting the maximum concentration to be non-zero, Environ will assume the user wishes to use the modified Poisson-Boltzmann model.

Note: The gcs correction (Gouy-Chapman Stern) is a 1-D analytical solution to the Poisson-Boltzmann equation and thus is only valid for 2 dimensional systems. The correction has been shown to produce close results to the numerical equivalents and should therefore be considered if the user is looking to save computational time. The parabolic correction (as seen in previous examples) is a general correction that works in any number of dimensions, however in order for the gcs correction to be valid, there should be no parabolic correction.

As in previous examples, the solvent_mode parameter is set to 'full', due to a hole close to the Ag ions. A solvent-accessible but ion-free region in the proximity of the surface (Stern layer) is modeled by introducing an ion-exclusion function as described here[3].

[1] F. Nattino et al., J. Chem. Phys. 150, 041722 (2019)

[2] G. Fisicaro et al., J. Chem. Phys. 144, 014103 (2016)

[3] I. Dabo et al., arXiv 0901.0096 (2008)

The interface function respresenting the boundary between the quantum mechanical region and the electrolyte embedding environment is set in the BOUNDARY keyword

```
&BOUNDARY
   electrolyte_mode = 'system'
   electrolyte_distance = 10.0
   electrolyte_spread = 0.5
```

Here, the electrolyte_mode parameter is set to system, corresponding to a simple analytical function.

CHAPTER 3

Overview

As opposed to the more commonly used atomistic description of solvents in chemistry simulations, Environ implements a set of continuum models, with the goal of providing a method of describing systems in environments with the primary goal of maximizing the focus on the quantum mechanical system. The result is a relatively succinct description of the environment unsuitable for capturing certain effects within this regime, yet the advantages of reduced computational cost make this approach invaluable for many applications.

These models belong to a family of hierarchical approaches that separate the treatment of the quantum mechanical system from the environment. Varying choices for representing this environment result in varying groups of methods, each with their own ideal applications. For example Quantum Mechanics has been coupled with Molecular Mechanics (QM/MM) thus retaining atomistic details of the embedding system while still treating it with a less computationally intensive scheme. The drawback of such an approach is perhaps not going far enough in terms of cost reduction, since a statistical sampling is still required for the environment configurations. Additionally the accuracy of such an approach is strictly dependent on the quality of the classical parameterization used to describe the components within the environment.

Continuum models go one step further, now removing atomistic details in the environment under the assumption that a statistical average of the environment results in a homogenous continuum. This can be justified for many solvent systems (with the exception of perhaps solvent atoms close to the QM system, in certain scenarios). One popular continuum model within the chemistry community is the Polarizable Continuum Model (PCM), which has been progressively extended and optimized for simulation stability and speed over the years. The breakthrough in the field of condensed matter was the Fattebert and Gygi (FG) model, which kickstarted substantial interest and many publications in the following years. The continuum models implemented in Environ are in some ways derived from the ideas presented in the PCM and the FG model. These have been successfully applied to neutral liquids and electrolyte solutions, with periodic crystal structures and small isolated molecules.

More recent developments have been made to improve these methods, especially towards better transferability, and there has been work in proposed multi-scale approaches that combine multiple different embedding schemes, with the same goals of effectively distributing computational time towards more important areas.

CHAPTER 4

Continuum

Continuum models were originally designed to handle liquid solution environments where the statistical averaging assumption in the relevant regions was viable. Typically these environments would have neutral net charge and a temperature where specific interactions like hydrogen bonding did not compromise the homogeneity of the system. These models have evolved to handle a wider variety of systems, such as ionic liquids and complex multiscale dielectric environments, such as metal nanoparticles and plasmonic devices on nearby QM systems.

CHAPTER 5

Interface Models

This page presents a more in-depth description of the interface models implemented in Environ, along with all revelant input parameters. Much of this text is structured to mirror a tutorial review on Continuum Embeddings,[1].

In general the interface is a 3D continuous, differentiable scalar field with a range from 0 to 1. A value of 1 (or within some tolerance) refers to the system, and a value of 0 (or within some tolerance) refers to the environment. The boundary is a function of the ensemble of the positions of the environment molecule. The strategy taken by continuum embedding models is to define this boundary from the system information only. This avoids the need for statistical sampling and/or other computationally expensive approaches.

The tutorial review summarizes the reasons for choosing a smooth boundary over a sharp one. Environ implements smooth boundaries since the QM simulation of the embedded system works in 3D (and reciprocal) space. The choice to retain 3D when working with the interface bypasses the possible issues encountered in discretization, where singularities and discontinuities are possible. We argue that from a computational standpoint, it is more reasonable to work with a set of vector and scalar fields that share the same numerical domain.

5.1 Self-Consistent (SCCS)

The SCCS model is described comprehensively in the 2012 publication[2], we present here a summary of the theory and methodology behind the model.

This model chooses to base the definition of the interface on the electronic density, which is a scalar field that varies from a higher magnitude close to the ions, to a lower magnitude as we move into the environment. This approach is

[1] O. Andreussi and G. Fisicaro, Wiley DOI: 10.1002/qua.25725 (2018)

[2] O. Andreussi, I. Dabo, and N. Marzari, J. Chem. Phys. 136, 064102 (2012)

based on the original model proposed by Fattebert and Gygi[3].

$$s(\mathbf{r}) = \frac{1}{2}\left(1 - \frac{1-(\rho^{\mathrm{el}}(\mathbf{r})/\rho_0)^{2\beta}}{1+(\rho^{\mathrm{el}}(\mathbf{r})/\rho_0)^{2\beta}}\right)$$

which has two parameters, ρ_0 and β. Instead it uses a piece-wise definition,

$$s(\mathbf{r}) = t(\ln(\rho^{\mathrm{el}}(\mathbf{r})))$$

where $t(x)$ is a smooth function, and the two parameters are ρ_{min} and ρ_{max} that bound the above function, the function returns 1 with an input above the max density bound and 0 with an input below the min density bound.

5.2 Soft-Sphere (SSCS)

The SSCS model is described comprehensively in the 2017 publication[4], again, we present here a summary of the theory and methodology behind the model.

This model choose to base the definition of the interface on the ionic positions, and takes a similar approach to the PCM model[5]. In the same vein as the SCCS model, rather than sticking with a 2D definition of the boundary, the model defines its smooth 3D interface function on interlocking smooth spheres centered on the ionic positions, and scaled depending on the atom, with an additional global scaling for parameterization.

5.3 Non-local interfaces

The SCCS model can be thought of as a local interface, the value of the interface function is solely dependent on the electronic density at that position. Likewise, the interface function of the soft-sphere is agnostic of the 'bigger picture'. There are a number of limitations of this assumption. Firstly, it is easy to imagine that more complex system with artifacts such as cavities devoid of molecule where a solvent should not access, will be misrepresented by a local model, which will by design fill any cavities without regard for the physical implications. Secondly, in the SCCS, the net system charge will influence the size of the QM cavity. Physically this is desirable since for charged systems, a solvent such as water would interact more closely with the system, but only for positive charges, where the SCCS will shrink the QM cavity. For negative charges, the model scales the system in the wrong direction and therefore produces poor results without a reparameterization.

There have been a number of recent techniques proposed in the literature that have been implemented (or still under development) in Environ. First is the solvent-aware model, which can be toggled in conjunction with either implemented interface model.

[3] J. L. Fattebert, F. Gygi, J. Comput. Chem., 2002, 23(6), 662

[4] G. Fisicaro et al., J. Comput. Chem., 2017 13(8), 3829

[5] S. Miertus, E. Scrocco, J. Tomasi, J. Chem. Phys. 1981 2;55(1):117-129

CHAPTER 6

Interactions

The following pages cover the interactions between the embedded system and the environment.

6.1 Dielectric Medium

Electrostatic solvation effects can be described in terms of a reaction field, which provides a stabilization effect to the system. In classical electrostatics, this effect is represented in terms of a dielectric medium with some boundary and a dielectric permittivity, which has different representations depending on the application. Here we consider the permittivity as a function in space, thus making the definition of a sharp boundary surface redundant, which itself is consistent with our definition of the interface function.

Fattebert and Gygi presented an electrostatic picture in terms of 3D fields, with the total electrostatic free energy of the embedded system as

$$F[\rho^{\text{el}}, \{\mathbf{R}_i\}] = \int \rho^{\text{sys}}(\mathbf{r})\phi(\mathbf{r})d^3\mathbf{r} - \int \frac{1}{8\pi}\epsilon(\mathbf{r})|\nabla\phi(\mathbf{r})|^2 d^3\mathbf{r}$$

where the electrostatic density term can be decomposed into electronic and ionic terms,

$$\rho^{\text{sys}}(\mathbf{r}) = \left(\rho^{\text{el}}(\mathbf{r}) + \sum_i z_i \delta(|\mathbf{r} - \mathbf{R}_i|)\right),$$

z_i are the ionic charges, $\phi(\mathbf{r})$ is the electrostatic potential, and $\epsilon(\mathbf{r})$ is the dielectric permittivity, which reflects the system/continuum separation and is thus a function of the continuum interface, for example,

$$\epsilon(\mathbf{r}) \equiv \epsilon(s(\mathbf{r})) = 1 + (\epsilon_0 - 1)(1 - s(\mathbf{r})),$$

where ϵ_0 is the bulk dielectric permittivity of the environment.

Note: This dielectric permittivity is the constant set in the Environ input file. Hence Environ takes this value, and calculates an interface function that scales from 0 (environment) to 1 (system). As one can see from the equation, the

result is a smoothly scaling dielectric permittivity function that transitions at the computed interface, from 1 (system) representing vacuum, to the permittivity specified by the environment, say 80 for water.

The functional derivative of the total electrostatic free energy functional defined above with respect to the electrostatic potential is simply the generalized Poisson equation (GPE),

$$\nabla \cdot \epsilon(\mathbf{r}) \nabla \phi(\mathbf{r}) = -4\pi \rho^{\text{sys}}(\mathbf{r}),$$

which is similar to the more popularly known Poisson equation in vacuum,

$$\nabla^2 \phi^0(\mathbf{r}) = -4\pi \rho^{\text{sys}}(\mathbf{r}),$$

following the convention that the dielectric permittivity of vacuum is simply 1. The polarization potential is simply the difference between the vacuum potential and the generalized potential. The generalized Poisson equation is non-trivial to solve, and an alternative apporach is to define an auxiliary polarization density, spread at the continuum interface, and expressed as

$$\rho^{\text{pol}}(\mathbf{r}) = \frac{1}{4\pi} \nabla \ln \epsilon(\mathbf{r}) \cdot \nabla \phi(\mathbf{r}) - \frac{\epsilon(\mathbf{r}) - 1}{\epsilon(\mathbf{r})} \rho^{\text{sys}}(\mathbf{r}).$$

This depends on the gradient of the logarithm of the dielectric function, and thus Andreussi et al. proposed an alternative formulation of the dielectric in terms of the continuum interface,

$$\epsilon(\mathbf{r}) = \exp(\log(\epsilon_0[1 - s(\mathbf{r})])).$$

When optimizing the embedded system degrees of freedom, the functional derivatives of the energy with respect to the electronic density or atomic positions need to be computed. It is possible to break down these derivatives into more transferable forms,

$$V_{\text{KS}}^{\text{interface}}(\mathbf{r}) = \int \frac{\delta s(\mathbf{r}')}{\delta \rho^{\text{el}}(\mathbf{r})} \frac{\delta F[s]}{\delta s(\mathbf{r}')} d^3(\mathbf{r})',$$

and

$$f_i^{\text{interface}} = -\int \frac{\delta s(\mathbf{r})}{\delta \mathbf{R}_a} \frac{\delta F[s]}{\delta s(\mathbf{r})} d^3(\mathbf{r}).$$

6.2 Diffuse Layer Models

This page presents a more in-depth description of the diffuse layer models implemented in Environ, along with all revelant input parameters.

6.2.1 Explicit Charge

Environ has the ability to represent external charges of various types. As with most of the examples on this page, we will assume a setup consisting of some noble metal slab (i.e. a two-dimensional system) that can possess some surface charge. To compensate we then consider the addition of countercharge, either explicitly or via an electrolyte solution. Explicit charges come in a variety of implemented forms depending on the specified dimensionality. Suppose we want a point charge electron at some arbitrary position. One could enter the following into the Environ input file,

```
EXTERNAL_CHARGES (bohr)
-1 0. 0. 0. 1.0 0 0
```

Remember to update the env_external_charges parameter appropriately (here we could have something like

```
&ENVIRON
   env_external_charges = 1
```

This places a point-charge (since the dimensionality is set to 0), at the origin. The charge is given a negative value whose magnitude equals that of an electron. The spread is set to 1.0. The dimensionality is given as 0, and the axis setting is irrelevent but set to 0 anyway. The definition of the spread is based off the following 1D definition for the gaussian,

$$ae^{\frac{(x-b)^2}{c^2}}$$

Note the lack of the factor of two that some definitions use. In three dimensions, the equation is effectively the same, only radial around the 3-dimensional point specified in the input. The gaussian is normalized to the specified charge. This convention has a lot of transferability, and by simply changing the dimensionality of the input, can be extended to line charges, and planar charges. For these objects, the axis is relevant. For a line charge, the chosen axis corresponds to a the axis parallel to the line charge, and for a plane, it corresponds to the axis normal to the plane. Note that this definition clearly does not account for all possible definitions of line and plane charge distributions, but this can be achieved instead by changing the atomic positions of the slab.

For an example of plane countercharges, see *Example: Charge Distributions*.

For models with charge distribution, the free energy functional can be written as

$$F^{\text{PC}}[\rho(\mathbf{r}),\phi(\mathbf{r}]=\int\left[-\frac{\epsilon(\mathbf{r})}{8\pi}|\nabla\phi(\mathbf{r})|^2+\rho(\mathbf{r})\phi(\mathbf{r})+\rho^{\text{ions}}(\mathbf{r})\phi(\mathbf{r})\right]d\mathbf{r}$$

where $\rho(\mathbf{r})$ is the total charge density of the solute, $\phi(\mathbf{r})$ is the electrostatic potential, and $\rho^{\text{ions}}(\mathbf{r})$ is the external charge density that mimics the counterion accumulation.

6.2.2 Poisson-Boltzmann model

The Poisson-Boltzmann model accounts for the chemical potential and the entropy of the ions in solution. Adding these consequently explicit concentration-dependent terms results in a different free energy functional than before,

$$\begin{aligned}F^{\text{PC}}[\rho(\mathbf{r}),\phi(\mathbf{r},\{c_i(\mathbf{r}\}]=\int\Bigg[&-\frac{\epsilon(\mathbf{r})}{8\pi}|\nabla\phi(\mathbf{r})|^2+\rho(\mathbf{r})\phi(\mathbf{r})+\rho^{\text{ions}}(\mathbf{r})\phi(\mathbf{r})\\&-\sum_{i=1}^{\text{p}}\mu_i(c_i(\mathbf{r})-c_i^0)-T(s[\{c_i(\mathbf{r})\}]-s[\{c_i^0\}])\Bigg]d\mathbf{r}.\end{aligned}$$

Here, μ_i is the chemical potential of the ith electrolyte species and T is the temperature, that is included as part of the entropy term.

We assume that these electrolytes have point-charge, and there is ideal mixing, leading us to the following entropy expression,

$$s[\{c_i\}]=-k_B\sum_{i=1}^{\text{p}}c_i(\mathbf{r})\ln\frac{c_i(\mathbf{r})}{\gamma(\mathbf{r})}$$

where $\gamma(\mathbf{r})$ is the exclusion function and sets the boundary between the electrolyte and solute regions.

The functional above is minimized with respect to the concentrations in order to find the equilibrium for electrolyte concentrations. This leads to the well-known Poisson-Boltzmann equation, which can be numerically solved using Newton's iterative algorithm and a preconditioned conjugate gradient algorithm, which works with linear equations.

There are times when it is beneficial to approximate the Poisson-Boltzmann equation, which is non-linear and thus non-trivial to solve. For $z_i\phi(\mathbf{r})\ll k_BT$, that is, when the electrostatic potential is low, or in the high-temperature limit,

one can approximate the Poisson-Boltzmann by its linear form, which is derived by Taylor expanding the appropriate terms in the full Poisson-Boltzmann equation. The result is a less computationally expensive approach to representing the electrolyte, while still maintaining the same parameter set.

In the case where we deal with linear slabs with two-dimensional periodicity, a different approximation can be used. The idea here is that the one-dimensional Poisson-Boltzmann equation can be analytically solved and thus this result can be applied as a PBC (periodic boundary condition) correction. This approach is also known as the Gouy-Chapman Stern model for an electrolyte, and compares well to the numerical approach for our examples.

6.2.3 Modified Poisson-Boltzmann model

The Poisson-Boltzmann can be improved by dropping the assumpion of point-like ions. This assumption leads to an overestimate of the electrolyte countercharge accumulation at electrode surfaces, and thus, by accounting for the steric repulsion between ions, which opposes the electrostatic attraction between the counterions and the electrode surface, one can improve on the full Poisson-Boltzmann model. The size-modified Poisson-Boltzmann (MPB) can be derived from the same free energy functional as before, only with a modified entropy expression

$$s[\{c_i\}] = -k_B \sum_{i=1}^{\mathrm{p}} c_i(\mathbf{r}) \ln \frac{c_i(\mathbf{r})}{c_{\max}\gamma(\mathbf{r})}$$
$$-k_B \left(c_{\max}\gamma(\mathbf{r} - \sum_{i=1}^{\mathrm{p}} c_i(\mathbf{r}) \right) \ln \left(1 - \sum_{i=1}^{\mathrm{p}} \frac{c_i(\mathbf{r})}{c_{\max}\gamma(\mathbf{r})} \right).$$

The idea is to essentially impose a space dependent maximum ionic concentration, and the result is a better representation of the electrolyte, verified by a comparison to experimental differential capacitance.

6.2.4 Additional Interactions

6.2.5 Environ

Environ has implemented all of the above models in a modular way that allows one to mix and match models and correction methods where reasonable.

6.3 Pressure

There is a pressure field caused by the embedding environment. We represent this pressure field in a continuum fashion, following the concept of electronic enthalpy introduced by Cococcioni et al[1]. The approach exploits the notion of quantum volume, which can be straightforwardly defined in terms of a continuum interface function as

$$V \equiv V[s] = \int s(\mathbf{r}) d^3\mathbf{r}.$$

The contribution to the enthalpy of the system is expressed as

$$G^{\mathrm{PV}}[s] = P^{\mathrm{ext}} V[s],$$

with P^{ext} the external pressure of the environment. This approach is based on an electronic interface, but equivalent arguments may be adopted for ionic or mixed interfaces. The key ingredient for the derivation of the enthalpy contribution to the Kohn-Sham potential or the inter-atomic forces is the functional derivative of the additional energetic term with respect to the interface function, which for the equation defining the continuum interface function above, is simply

$$\frac{\delta F^{\mathrm{PV}}[s]}{\delta s}(\mathbf{r}) = 1.$$

[1] Cococcioni M, Mauri F, Ceder G, Marzari N, Phys. Rev. Lett. 2005 4;94(14):145501

6.4 Tension

Cococcioni et al.[1] proposed an environment effect related to the quantum surface of the embedded system. The macroscopic physical picture is the one of surface tension, which controls the size of the boundary between the system and environment. An embedding interaction based on the quantum surface was proposed by Scherlis et al.[2] to estimate the free energy penalty of creating a void into the continuum liquid solution, that is, the cavitation free energy,

$$F^{\text{cav}}[s] = \gamma S[s],$$

where the surface is defined in terms of the interface function,

$$S[s] = \int |\nabla s(\mathbf{r})| d^3\mathbf{r}.$$

Various publications have generalized and revised this cavitation free energy function[3],[4], the implementation of the surface tension being a result of this work. The functional derivative of the energy contribution with respect to the interface function is

$$\frac{\delta F^{\text{cav}}[s]}{\delta s}(\mathbf{r}) = -\nabla \cdot \left(\frac{\nabla s(\mathbf{r})}{|\nabla s(\mathbf{r}|} \right),$$

which is used in the calculation of the contribution to the Kohn-Sham potential or the interatomic forces.

[1] Cococcioni M, Mauri F, Ceder G, Marzari N, Phys. Rev. Lett. 2005 4;94(14):145501

[2] D. A. Scherlis, J. L. Fattebert, F. Gygi, M. Cococcioni, N. Marzari, J. Chem. Phys. 2006, 124(7), 74103

[3] Andreussi O, Dabo I, Marzari N, J. Chem. Phys. 2012 2;136(6):064102

[4] G. Fisicaro, L. Genovese, O. Andreussi, S. Mandal, N. N. Nair, N. Marzari, et al., J. Chem. Theory Comput. 2017, 13(8), 3829

CHAPTER 7

Environ Input

CHAPTER 8

QE PW Input File

This page explains the PW input files in more detail. It refers to each of the examples and the input files auto-generated by each of the scripts.

A standard PW file contains options for the locations of associated files, and any output files, as well any information about the system, including the atoms and their positions, and the dimensions of the cell. For users new to Quantum Espresso, it is often easier to copy working PW files and editing individual parameters rather than building these from scratch. For those that are interested, there are also tools that generate PW input files, such as ASE[1]. Alternatively a number of users have written their own helper scripts to expedite the tedious procedure of creating large batches of input files for larger investigations.

An essential reference page is the PW input file description page[2], which contains a brief explanation of every possible choice of parameter and their options.

8.1 Example 1

The input file generated for the first example is for an isolated water molecule chosen to be the system. Isolation is imposed by Environ, so although there are options on the Quantum-Espresso side (see Example 3), these are not necessary here. We start at the bottom of the file with the definition of the water molecule. Quantum Espresso expects the initial positions of the water in some consistent reasonable choice of units.

```
ATOMIC_SPECIES
 H   1  H.pbe-rrkjus.UPF
 O  16  O.pbe-rrkjus.UPF
ATOMIC_POSITIONS (bohr)
O   11.79  12.05  11.50
H   13.45  11.22  11.50
H   10.56  10.66  11.50
```

The atomic species keyword introduces each individual atom present in the system, along with their atomic mass and the name of the pseudopotential file they should refer to. Note that Quantum Espresso can have multiple entries of

[1] https://wiki.fysik.dtu.dk/ase/index.html
[2] https://www.quantum-espresso.org/Doc/INPUT_PW.html

the same atomic species (which should be labelled differently). The pseudopotential files should be placed in a folder referenced by the pseudopotential directory choice (whose default can be set in the environment variables for Quantum Espresso, or manually set as we will show here in the input file).

CHAPTER 9

Output Files

Unlike Quantum-ESPRESSO, which parses the entire output into an XML file, Environ currently relies solely on the output file and an associated environ-debug file (if the appropriate verbosity is set in the input). The reason for this is that Environ is currently a plugin and does not overwrite or modify any of the base Quantum-ESPRESSO files. Hence any additional output is only currently printed out into the terminal (or piped into an output file), or in environ specific output files. This may be dealt with more elegantly in the future.

9.1 Reading and Automation

When using Environ, it is more appropriate to rely on the direct output of the program. One can feed this into a file and parse that file, typically using grep on certain keywords, or using scripting. Users have developed post-processing tools for extracting information for personal use. These can be particularly attractive when running many simulations for a particular investigation and as such, it is likely that we will release some helpful post-processing tools along with future releases.

CHAPTER 10

FAQ

This section attempts to address Frequently Asked Questions regarding Environ. If your query does not relate to any of these sections, check out the Q&A section on the website, which has instructions on joining the mailing group where you can chat with one of the developers.

10.1 Converge Issues

There are physical and/or numerical reasons for why a calculation may explode. The latter is easier to solve, by tuning some of the parameters in the input file. The former is a little more tricky to handle.

Roughly speaking, in a calculation with Environ, one needs to include some extra terms to the Kohn-Sham potential of the system in vacuum. These terms depend on the electronic density itself, in particular due to the fact that the interface between the system and the environment is defined in terms of the electronic density. Moreover, to compute one of these terms (e.g. the electrostatic interaction with the dielectric continuum), Environ uses an iterative approach, so at every SCF step, there is an additional iterative procedure performed by Environ, let's call this the polarization iteration, in contrast to the SCF iteration.

There are two Environ parameters which are crucial to help convergence of the total SCF calculation, `tol`, which controls the convergence of the polarization iteration, and `environ_thr`, which determines when the polarization iteration starts.

When you decrease `tol`, the extra term coming from the interaction with the dielectric will be computed more accurately, but it will require more time to be computed. On the other hand, if the parameter is not set to be sufficiently accurate, the SCF procedure may not convergence as quickly, thus slowing the simulation down noticeably. Furthermore, a low convergence speed in the SCF may leads to oscillation around the SCF threshold, leading to convergence failure. Due to the fact that in general, the polarization iteration is less expensive than the SCF iteration, it is recommended that the user opts for a decreased `tol` when encountering these issues. Hence if in doubt, start with a generic value for the tolerance, be it the default or some value taken from the examples (depending on what you're trying to simulate), and decrease if necessary.

The existence of `environ_thr` allows for better convergence in general, since it is advisable not to add polarization correction terms in the first few SCF steps, due to the fact the accuracy of these energy terms will still be poor (although if one is starting from a good set of input parameters, it is probably fine to enable the polarization correction

from the start of the iteration, in which case the `environ_restart = .true.` will enable this). On the other hand, waiting too long to start adding Environ corrections can lead to poor convergence due to the fact that before the polarization iteration corrections kick in, the system is converging for vacuum, not a solution. Hence one can analyze the output file and see when the correction terms begin, in order to judge whether `environ_thr` is good or not. If the SCF accuracy explodes or converges poorly after the polarization iterations begin, it is recommended that one increases `environ_thr`. One rule of thumb to go by would be to make sure that Environ corrections skip only around 3-5 SCF steps.

A further source of errors may arise due to the use of pseudopotentials for handling core electrons. This problem is more common in transition metals of halogens, and it comes from the neglect of these core electrons in the charge density used to build the interface with the environment. In some cases there appears to be a hole in the charge density on the nuclei, which is then filled with the environment (in fact, a couple of the examples in the tutorial address this fact and account for it in the input file). The solution is to use `solvent_mode = 'full'`. This option adds additional gaussians centered on each atom, effectively describing the core electrons and thus not allowing the environment to permeate inside the atoms, while not compromising the definition of the interface.

Another suggestion is to simplify the physics in the problem. In particular, since in most applications the most important effects of the environment are the electrostatic ones, you may want to switch off all the other non-electrostatic terms. This is possible by setting `environ_type = 'input'` and manually setting the parameters that you want. By only setting `env_static_permittivity = 78.3`, one only retains the dielectric continuum in the calculation, since all other contributions are off by default.

When simulating a 2D system, convergence of the polarization potential is made more difficult by the artificial finite electric field coming from periodic boundary conditions. To avoid this, it would be better to use a PBC correction, in particular for slabs in Environ, a parabolic correction is implemented, and alternatively, one can utilize the Quantum ESPRESSO isolated options. See example 3 for a demonstration on isolated systems. Increasing the cellsize perpendicular to the slab will decrease periodic effects but increase convergence times due to the larger cell.

One reason of poor convergence is simply a complex shape of the interface. New releases implement models designed to deal with some of these complex shapes.

10.2 Harris-Foulkes Estimate

The Harris-Foulkes estimate does not take into account the extra terms coming from Environ, as it would cost more than necessary for a quantity that is only used to have an estimate. As a matter of fact, also the SCF accuracy does not take into full account of the environment, but again it would need a secondary correction that would cost more than its utility.

10.3 Environ Total-Energy Sum

The energy output in an Environ calculation may be confusing in that the total energy is not the sum of the reported terms and it is not immediately clear what solvation energy means, but before going into the details of why, one point should be addressed. `dGsol`, and in particular `dGel`, are not quantities that you get from a single calculation in solution, but rather are obtained as the difference in energy from a calculation fully optimized (nuclei and electrons) in solution minus a calculation fully optimized (nuclei and electrons) in vacuum. This is explained in Example 1 of the Tutorial. In general, for these values, one needs two geometry optimization (relax) calculations for these energy values that are typically reported in our publications. Hence solvation energy requires two calculations.

The different terms do not add to the same energy due to the fact that there is a spurious extra term which is subtracted from the one-electron contribution, but is not reported in the output. The reason is that total energy is computed in pw making use of the secular expression (see for example W. E. Pickett, Computer Physics Reports, 9(3), 115-198, 1989, equations 5.21-5.22 etc.), thus from the sum of occupied states eigenvalues (so called band structure energy, “eband” in pw.x), in which some contributions are included correctly (the kinetic and the electron-nuclei interaction), some are

double counted (hartree), some need to be removed and added back with the correct expression (xc term), and some are missing (nuclei-nuclei interaction via ewald sum). What PW reports in the output as the one-electron contribution is in fact ebands minus the double counted hartree and minus the wrong xc term (these spurious terms are summed up in pw.x into a term named "deband").

Similarly, when performing the solvated calculation, there are some terms which are double counted (the electronic part of the solvation energy), some that are missing (the ionic part of the solvation energy), some that have the wrong expression (pressure and cavitation) and some that need to be removed (for example a term coming from the rho-dependence of the dielectric constant). All these spurious terms are collected in a term named "deenviron", which has the same meaning as "deband" seen above. To get to the point, what is wrong in the output is the one-electron contribution, which is reported including the spurious deenviron term. As the Environ module was designed not to affect the standard printout of QE, this output was not modified, and consequently, the terms do not sum to the correct total energy. The correction deenviron and deband terms were not reported, due to their insignifance in anything alone. However, this may well change in future Environ releases.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search