DinEMol - UFSC:
Dynamics of Electrons in Molecules

Explanatory Notes
(under preparation)

Summary of most recent updates

- early draft describing the general operation of the code

Contents

Chapter 1

Introduction

1.1 Downloading the code

The code has been developed under the project name is *"Charge-Transfer: tools for studying interfacial electronic charge transfer"*. For this project, a suite of routines has been created to manipulate and study charge transfer processes in large molecular nanostructures. The project is hosted on the **Google Code** website and the source files can be freely downloaded from there, by accessing
http://code.google.com/p/charge-transfer/
Details on how to use the Google Code website and its subversion repository commands (svn) can be found in the website. In particular, anonymously check out (downaload) the latest project source code of any of the suite of codes of the repository to any computer, the command-line **svn checkout** can be used. [1]

The open source files are under constant development and its updates are available at the Google Code website. Thus, some of the new features of the code may be incomplete and still in test phase. Apart from that, this documation does not intend to describe all the features of the program along with its most recent updates.

This manual is dedicated to describe the **DinEMol** code, located at the repository's directory
https://charge-transfer.googlecode.com/svn/branches/QMMM. But another useful suite of programs dedicated to manipulate molecular and extended system structures can be found at
https://charge-transfer.googlecode.com/svn/trunk/manipulate. The later can be used to:

- convert molecular structures for various chemical file formats,
- read and manipulate molecular frames,
- carry out copy, delete, paste, translate and rotate operations on the molecular structure,
- define residues, fragments, special atoms types, and more,
- to perform statistical analysis of molecular frames, such bond properties (length, angle, torsion), radial (and linear) density function distributions, and correlation analysis

[1] http://code.google.com/p/charge-transfer/source/checkout

Chapter 2

Installation

After downloading the source files, the installation of the code is carried out by a simple makefile utility, which comes along with the codes. Adjustments are usually necessary for using the make utility to compile the codes, to set the path to the fortran compiler and the necessary libraries. In most of the cases, after these minor modifications to the makefile, compilation is straightforward.

2.1 Makefile

Makefiles are special format files that, together with the make utility, compile all the source codes and link them amongst themselves and with the designated libraries to produce the executable file. The makefile is machine dependent, therefore it may need to be modified to run on different machines or different directory trees.

The source codes of the **DinEMol** suite are written in Fortran 95/2003, including also some routines in Fortran 77 and some features in Fortran 2008. The most recent implementation sets the standard for the entire code, therefore the compiler of choice must comply with Fortran 2008 feature. There are various compilers that support the Fortran 2008 standards (Absoft, Cray, g95, GNU, HP, IBM, Intel, NAG, Oracle, PathScale, PGI) [1] A non-commercial version of the Intel Fortran Composer XE 2013 for Linux can be downloaded from
http://software.intel.com/en-us/non-commercial-software-development

In summary, the makefile variables define:

- FC : the Fortran compiler with its path;
- FLAGS : compilation flags, which may include optimization, free format, parallelization, openmp, etc .;
- LIB : links to the necessary libraries. The code uses mkl (Intel Math Kernel Library), which is a combination of Basic Linear Algebra Subprograms (BLAS) and Linear Algebra Package (LAPACK), both optimized for Intel processors. It also makes use also of Open MP parallelization directives;
- SOURCE : a list of source files;
- compilation rules : The name of the executable is just **a**, but it can be changed as desired;

[1] The Fortran 2008 status can be found at http://fortranwiki.org/fortran/show/Fortran+2008+status.

- clean : cleaning rules.

If the compiler and library files are correctly determined in the make file the user just have to type **make**, the make utility of Linux will first compile the individual codes, then link them together and with the libraries, to produce an executable named **a**.

If a modification is made in any of the source files – files with suffixes **.f** of **.F** – the make utility will recompile **only** the modified source files and link it with the other unchanged object files – files with suffixes **.o**. This produces an updated version of the executable. Because of the dependencies amongst the object files, sometimes that recompilation process renders incomplete, producing a faulty executable. When it happens, to correct that faulty compilation, a total recompilation is necessary. There are two ways of doing it: the simplest is to type **touch *.f**, then **make**; it will recompile all source files with **.f** suffix. The other is accomplished by typing **clean**, which will erase all object files, and then **make** to compile everything again.

If openmp is implemented during compilation, it will use all the processors in a node. The code can run in a laptop or workstation, as long as it has the compiler and libraries. Openmp is not necessary; MPI is not implemented yet.

2.2 Example

To download the **DinEMol** suite of source codes from the Google Code repository to a directory named **test** on your computer just type

```
> svn checkout https://charge-transfer.googlecode.com/svn/branches/QMMM/    test/
```

It will produce:

```
 A test/md_read.f
A test/my_eht_parameters.dat
.
.
.
A test/main.f
U test
checked out revision 406.
```

Then go to directory /**test**, make the necessary adjustments in the makefile as well as in the the input the control source files, and do:

```
 > make
/intel/bin/ifort -O3 -I/intel/composer/mkl/include/intel64/lp64 -c integ-Coul.F
.
.
.
/intel/bin/ifort -O1 -openmp -parallel -free -I/intel/composer/mkl/include/intel64/lp64
-c main.f  ·
.
```

.

An executable named **a** will be produced.

Chapter 3

The System Input files

The program reads system input files in **pdb**, **xyz** or **vasp-POSCAR** formats. However, the pdb format is strongly recommended because i it is much clearer and conveys a lot more information about the system's structures than the other formats. The input files can describe a single (static) molecular configuration, or multiple molecular frames and trajectories. Accordingly, the input file types can be either **structure** or **trajectory**.

Together with **structure** file types it is possible to propagate electronic wavepackets upon a rigid molecular structure, or to perform calculations of Density of states (DOS), optical spectroscopy, and dipole moments for a rigid molecular structures. i In this case, the input file consists of a single molecular configuration, given by files named **input.pdb** , **xyz.dat** , or **poscar.in**.

The **trajectory** file types can be used to carry out ensemble averages over multiple molecular configurations, in the case of DOS, spectrum, and dipole moment calculations. The multiple configurations can also be used as distinct initial conformations in the time propagation of electronic wavepackets upon rigid molecular structures; at the end the survival probabilities are averaged. Still in regard to the **trajectory** file types, it can be used to propagate electronic wavepackets are propagated upon a dynamic molecular structure. To use the **trajectory** keyword one must provide several molecular configurations in a file named **frames.pdb**, or **XYZ.trj**. More information about the use of **structure** or **trajectory** input files can be found in chapter 4.

The **trajectory** file type can be used to provide information about the nuclear dynamics, if the time interval between successive frames is short ($\delta t < 1fs$) [1], or it can be used to provide the program with an ensemble of independent molecular configurations ($\delta t > 1ps$).
The file **IO_file_formats** describes in detail the file format for each case, including the **required tabulation**. If the tabulation rules are not followed the code will not read correctly the data.
The system input files are named as:

- xyz.dat : single (static) molecular configuration in xyz format;
- input.pdb : single (static) molecular configuration in pdb format;
- poscar.in : single (static) molecular configuration format similar to the POSCAR file produced by the VASP package;

[1] Note that the quantum dynamics simulations are usually carried for intervals with the picosecond time scale.

- frames.pdb : multiple frames in pdb format;
- XYZ.trj : multiple frames in xyz format.

Additional information about the input files: following the keyword **CRYST1**, T_x,T_y,T_z are the lengths of a rectangular bounding box. If periodic boundary conditons (PBC) are used the lengths correspond to the dimensions of the unit (super)cell. Information about the angles of the bounding box is unecessary (90^o is assumed). Regarding the bottom line of **input.pdb**, the code looks for the string **MASTER** to count the number of atoms, thus you need not to inform the number of atoms in the structure.

3.1 Alternatives for converting chemical file formats

3.1.1 OpenBabel

OpenBabel is a free software mainly used for converting chemical file formats. This program belongs more to the category cheminformatics than to molecular modelling. It is available for Windows, Unix, and Mac OS. It is distributed under the GNU GPL. Open Babel is a community-driven scientific project assisting both users and developers as a cross-platform program and library designed to support molecular modeling, chemistry, and many related areas, including interconversion of file formats and data. More information can be found at http://openbabel.org/wiki/Main_Page

The general use is as

```
babel -i input_format input_file -o output_format output_file
```

for example, to convert input.mol to input.pdb, type

```
babel -i mol input.mol -o pdb input.pdb
```

Notice: check whether the tabulation produced by running OpenBabel is consistent with **IO_file_formats**.

3.1.2 Manipulate utility

Another useful suite of programs dedicated to manipulate molecular and extended system structures can be found at https://charge-transfer.googlecode.com/svn/trunk/manipulate. The later can be used to:

- convert molecular structures for various chemical file formats,
- read and manipulate molecular frames,
- carry out copy, delete, paste, translate and rotate operations on the molecular structure,
- define residues, fragments, special atoms types, and more,
- to perform statistical analysis of molecular frames, such bond properties (length, angle, torsion), radial (and linear) density function distributions, and correlation analysis

3.1.3 Final Comments

In principle, any program (or command) can be used to generate the pdb files. Notice that the pdb files generated somewhere else may have more columns than those actually read by our code; it doesn't matter, as long as the necessary columns are present and correctly tabulated. The same principle holds for any of the input files.

Chapter 4

File parameters.f

The parameters.f file is the main input file, with command parameters for controlling the basic program functioning. It contains parameters with driver options and auxiliary commands for determing the type of the input file, the IO options, the physical potentials to be included in the hamiltonian, the boundary condition options, and more. Some of them are strings and some are simply logical flags. This document presents the command variables in the order of appearance. The commands are grouped according to their purpose. Note that the string keywords are case-sensitive.

4.1 Action flags

The DRIVER string options are:

1. **diagnostic** : used for basic diagnostics of the system, which includes the calculation of molecular orbital energies, density of states (total and projected on the system fragments), dipole moments, optical spectrum, finite-field polarizabilities, *etc.*. This DRIVER option is intended to provide the means for a quick analysis of the various properties of the system. In especial, this option accepts arguments along with the line command, for designating the indexes of molecular orbitals whose isosurface cube files are produced. For instance, if the keyword Gaussian_cube = T_, and the executable is named a, then
   > `./a 34 40 45`
   will generate isosurface cube files for the molecular orbitals whose indexes are 34, 40 and 45.

2. **q_dynamics** : used for propagating of an electronic wavepacket in a rigid molecular structure provided by **input.pdb**;

3. **slice** : all drivers named **slice** can be used for propagation of wavepackets upon dynamic nuclear structures.

4. **avrg_confgs** : used for averaging over independent molecular structures (Thermodynamical averages);

5. **Genetic_Alg** : used for optimizing new Hückel parameters. Another command key, the CG_ logical flag, is used to post-optimize the Hückel parameters by the conjugate-gradient method. The penalty (cost) function used in both optimization methods is defined in the **evaluate_cost** function, which is located in the **cost_tuning.f** file.

Some auxilliary command flags are:

- Survival : logical flag, for dynamics calculations of survival probability set as T_;
- DP_moment : logical flag, for calculations of the dipole moment. The dipole moment can be calculated for a specific fragment molecular orbital (FMO), as set in the **ad_hoc** subroutine, or for the entire system (default option);
- Alpha_Tensor : logical flag, for calculating the diagonal elements of the polarization tensor by the finite-field method;
- spectrum : logical flag, for calculations of the optical spectrum. It is also necessary to set the interval of occupied and unoccupied orbitals to perform the calculations, as done through the **occupied** and **empty** variables, defined ahead in the same **parameters.f** file;
- OPT_parms : logical flag. The program uses Hückel optimized parameters read from **OPT_eht_parameters.input.dat**. If set as false, F_, otherwise it just uses the standard Hückel parameters that can be found in the file **my_eht_parameters.dat**;
- ad_hoc : logical flag. By setting as T_, it determines that some fine-tuning of the system's parameters is performed by the directives of the **subroutine ad_hoc**. The directives are set directly at the **subroutine ad_hoc** that is found in the **tuning.f** file. If no special directives are necessary just use F_. In practice, the ad_hoc keyword calls a subroutine where specific information about the system can be introduced "by hand". For instance, if one wants to define a new residue, or define a new atom type, or define the residue where the electronic wavepacket is initially located, etc., assuming that this information was not already in the input.pdb or frames.pdb. This is very convenient for fine tuning the information that is conveyed from the pdb files.

The **state_of_matter** keyword determines how the molecular (nuclear) system is to be described. It can be one of the following:

- extended_sys : default input format, used in almost all calculations;
- solvated_sys : for solvated molecules, creates a droplet with the solute in the center of the solvent sphere;
- MDynamics : nuclear dynamics is calculated on the fly by the Molecular Mechanics method; for details about the implementation and execution of the Molecular Mechanics method see Chapter 8. The electronic quantum dynamics propagation is coupled to the Molecular Mechanics propagation of the nuclei.

4.2 File Format

The keyword **file_type** determines whether the molecular structure is dynamic or static:

- structure: the input file contains information of a static molecular structure. Together with file_type=structure, the following driver options can be used: **q_dynamics** and **diagnostic**, for propagating the electronic wavepackets upon a rigid molecular structure

or to perform calculations of DOS, spectrum, dipole moment for a rigid molecular structure. The input file in this case consists of a single molecular configuration, given by files named input.pdb, input.xyz or poscar.in.

- trajectory: the input file consists of a sequence of molecular frames (conformations). Together with file_type=trajectory, the following driver options can be used: **avrg_configs**, **slice_[Cheb, AO, ElHl]**. In the first case ensemble averages over multiple molecular configurations are performed for DOS, spectrum, and dipole moment calculations. In the second case, the electronic wavepackets are propagated upon a dynamic molecular structure. To use the trajectory keyword one must provide several molecular configurations in the files called frames.pdb or XYZ.trj.

- file_format determines the format of the input file that contains the molecular structure. Possible options are xyz, pdb or vasp.

4.3 Visualization flags

The visualization flags determine the files that are produced for visualing the isosurface of molecular orbitals and the charge dynamics. The respective keywords are:

- Gaussian_cube : logical flag. Produces gaussian cube files. It takes some time to perform such calculations, thus set to F_ unless you really want to visualize the orbital isosurfaces. The possible output files are named, for example, as el_MO_shot00031.cube or el_dens_shot00031.cube. The former contains information about molecular orbital 31, including the phases of the wavefunction; this is produced only in static calculations such as in DRIVER=diagnostic or Genetic_Alg. The latter file contains information about the probability density of the dynamics wavepacket and the number of the shot designates, in this case, not the molecular orbital but the time index; there are no phases in the probability density. This file type can be generated for either the electron (el_dens) and hole (hl_dens) wavepacket, by using DRIVER=q_dynamics or slice.

- GaussianCube_step : sets the step for calculating Gaussian cube files of electron and hole wavepackets.

- NetCharge : logical flag. Calculates the Mulliken atomic charges from the dynamic wavepackets. The atomic charges are stored in the file /tmp_data/NetCharge.inpt, to be used with the visualization program **vmd**. The **DinEMol** package provides a script file named **script-vmd** to be used with the VMD (Visual Molecular Dynamics) molecular graphics software, one can view the time dependent atomic charges or polarization with the line command **vmd -e script-vmd**.

- CH_and_DP logical flag. Calculations of the Mulliken atomic charges as well as the atomic dipole moments from the dynamic wavepackets. The atomic charges are stored in the file /tmp_data/NetCharge.inpt and the atomic dipole moments are stored in the file /tmp_data/DipoleSize.inpt, to be used with the visualization program **vmd**. In addition to these files the file /tmp_data/DipoleFrames.pdb is also saved.

- CH_and_DP_step : sets the step for calculating NetCharge or CH_and_DP frames of electron and hole wavepackets.

4.4 Security Copy

The Security Copy flags are used for saving and recovering calculation data, if the program is killed for any reason:

- restart : logical flag. If set to false (F_) the program starts a new calculation, from the beginning, and saves the backup data periodically in the unformated file **Security_copy.dat**. The old version of this file is overwritten when the program resumes. If set as true (T_) the program resumes execution by reading the data contained in **Restart_copy.dat**. Thus, for restarting a calculation, in addition to set this flag as T_ the user must move the file **Security_copy.dat** to **Restart_copy.dat**. After restarting, the program will continue to update the backup **Security_copy.dat**, which can be used if the execution stops again.

- step_security : sets the step for saving the backup file.

4.5 Potentials

The Potential logical flags determine which hamiltonian terms are to be included with the usual Extended Hückel tight-binding hamiltonian. The potentials are not mutually excludent, and their use is decided on the basis of the system's characteristics and the needs of the calculation. Nevertheless, the chosen Potential flags must be consistent with the DRIVER option and with the number of wavepackets in the simulation. In general the inclusion of extra Potentials render the simulation slower. The logical flags are:

- DP_field_ : to be used in the case of a polar (solvent) environment. It determines the calculation, on the fly, of the dipole field produced by the (polar) solvent molecules. It also calculates the dipole of the solute, which must be defined as a specific fragment in the **ad_hoc** subroutine. The matrix elements produced by the dipole fields are calculated and added to the EHT hamiltonian. It works with n_part = 1 or 2 and with all drivers, with the exception of Genetic_Alg.

- Coulomb_ : must be set along with **n_part** = 2. The flag determines the calculation of the direct Coulomb interaction between electron and hole wavepackets. It works only with the drivers q_dynamics, slice_Cheb and slice_ElHl. For electron and hole dynamics calculations the flag Coulomb_ can be set as T_ or F_, if it is set as false electrons and holes will evolve independently, otherwise the Coulomb potential will couple them. Note that coupled electron-hole calculations need a smaller time step, as compared to independent electron-hole calculations.

- Induced_ : used to calculate the induced (intramolecular) polarization field produced by the presence of the electronic and/or hole wavepackets during their propagation. It works only with n_part = 1 or 2 and with drivers q_dynamics, slice_AO and slice_ElHl.

4.6 Sampling and Quantum Dynamics parameters

The Sampling Parameter variable frame_step has a double purpose:

- used along with the **avrg_confgs** it determines how many frames, of the file frames.pdb, are skipped for calculating average properties. Ideally the structures cannot be correlated with each other, therefore a reasonably large **frame_step** value must be set. Usually, for being considered statistically independent, the time interval between molecular conformation frames varies approximately from 5 ps to 15 ps, depending on the characteristics of the system.

- used along with the time-propagation DRIVERS (slice_$\cdots$) it determines how many frames are skipped for describing the nuclear motion in the time slice dynamics. In constrast with the average calculations, the time-propagation calculations usually produce better results for smaller frame steps.

For example, assuming an input file containing pdb frames of interval dt= 0.2 fs, then **frame_step** = 1 will will calculate electron dynamics for molecular time-slices of 0.2 fs. For time_step=3 calculations are performed for time-slices=0.6fs, and so on.

The QDynamics parameters control the time propagation routines:

- t_i : initial time (usually t_i = 0).

- t_f : final time.

- n_t : number of time slices. The program stops if (t = t_f) or (it = n_f).

- n_part : The number of particles to be propagated, possible options are n_part=1 (only electron or hole), n_part = 2 (electron and hole). If the flag Coulomb_=T_, n_part must be 2. For ground state calculations of the Dipole Moment n_part=0.

- hole_state : designates the the molecular orbital of the fragment where the hole is initially located, that is, the value set to this variable corresponds to the molecular orbital index of the FMO where the hole is initially located. The ad_hoc_tuning subroutine, in the file the tuning.f, must be used to determine the fragment where the hole is initial located, by setting the variable univ%atom%Hl accordingly. For instance, if electron and hole are located at the residue MOL one must set
  where(univ % atom % residue == "MOL") univ % atom % El = .true.
  where(univ % atom % residue == "MOL") univ % atom % Hl = .true.
  or, otherwise for the hole but producing the same result
  where(univ % atom % residue == "MOL") univ % atom % El = .true.
  where(univ % atom % El) univ % atom % Hl = .true.
  Electron and hole need not be initially located at the same fragment (or residue) as, for instance, in the case
  where(univ % atom % residue == "C60") univ % atom % El = .true.
  where(univ % atom % residue == "MOL") univ % atom % Hl = .true.
  The hole state must correspond to the HOMO (or HOMO-1, or HOMO-2, ...) of a specific molecular fragment, otherwise the program will stop with an error message. If no holes are simulated use hole_state=0s, in this case the variable univ%atom%Hl is ignored.

- initial_state : designates the molecular orbital where the electron is initially located; this is mandatory for any dynamics calcualtion. If n_part=1 only electrons will be considered and hole_state = 0. The value set to this variable corresponds to the molecular orbital

index of the FMO where the electron is initially located. The ad_hoc_tuning subroutine, in the file the tuning.f, must be used to determine the fragment where the electron is initial located, by setting the variable univ%atom%El, as shown above. By default, the FMO where the electron is initially located is considered as the donor fragment, which is set as where(univ % atom % El) univ % atom % fragment = "D"
in ad_hoc_tuning.

4.7 Structural Parameters

The Structural Parameter variables determine some general characteristics of the system such as replication of the original structure or periodic boundary conditions. The variables T_x,T_y,T_z are used in the replication of the structure.

- nnx and nny : are integer variables: 0 means no replication, multiple replications are allowed. They determine whether the original structure, as read from the input files (pdb, xyz, etc.), is replicated along the $\hat{x}$ and/or $\hat{y}$ directions, respectively. The copies of the original system are applied symmetrically on each side of the original system; the donor unit cell always remains in the center of the augmented structure. For example, to create a "wire" along the $\hat{x}$ direction, just set nnx equal to some value $\neq 0$. For a larger planar struture nnx and nny are $\neq$ zero. Structures replicated like that increase the size of the basis states in the same proportion, therefore the replications cannot go too far, that is, usually nnx and nny $lesseq 1$. [1] Such structures can be used to avoid unphysical recurrences that occur with periodic boundary conditions.

- mmx , mmy , mmz : are used to set periodic boundary conditions along particular directions. The only possible values are: 0 that means no periodic boundary condition on that direction and 1 for applying the periodic condition. Boundary conditions can be applied independently on each direction. This feature does not increase the size of the basis states. Boundary conditions can be used to eliminate undesired surface states.

4.8 DOS parameters

These keywords are used for density of states (DOS) and optical spectrum calculations. Projected DOS calculations on specific **residues** are also possible if those residues are defined

[1] For instance, assume that your Hčkel matrix has dimensions 1000×1000: $\otimes$. If you set nnx = 1 and nny = 0, you end up with a Hückel matrix that is 3000×3000:$\diamond \otimes \diamond$. If you set nnx = 0 and nny = 1, you also end up with a Hückel matrix that is 3000×3000:

$$\begin{matrix} \diamond \\ \otimes \\ \diamond \end{matrix}$$

For both nnx=1 and nny=1, you end up with a matrix that is 9000×9000:

$$\begin{matrix} \diamond & \diamond & \diamond \\ \diamond & \otimes & \diamond \\ \diamond & \diamond & \diamond \end{matrix}$$

with the donor unit cell always in the center.

originally in the input.pdb or frames.pdb files, or set in the **ad_hoc_tuning** subroutine. Results for the total DOS are written in the TDOS.dat file and the PDOS results are written individually on files named PDOS-<*residue_name*>.dat. For DOS calculations we have:

- **sigma** : width of the gaussians used to broaden the DOS peaks.
- **DOS_range** : window of energy to be considered in the DOS calculations. States with energy out of this range will not be considered.

The broadening of DOS peaks can also be produced by means of ensemble averages over various structures gained from the **frames.pdb** file. Ideally the structures cannot be correlated with each other, therefore a reasonably large **frame_step** value must be set. Usually, the time interval between consecutive frames varies from 5 ps to 15 ps, depending on the characteristics of the system. Thus, if DOS calculations are evaluated with the DRIVER = avrg_confgs, the sigma variable is automatically divided by the number of ensemble elements, so that the gaussians become narrower. The same applies to spectrum calculations.

4.9 Spectrum parameters

Parameters for controlling the optical spectrum calculations:

- **sigma** : width of the gaussians used to broaden the oscillator strength peaks.
- **occupied** : sets the range of energies for occupied molecular orbitals; these are the orbitals up to the HOMO.
- **empty** : sets the range of energies for empty molecular orbitals; these are orbitals located up from the LUMO.

Optical transitions are calculated from occupied to unocuppied (empty) orbitals, generally a boundary is established between HOMO and LUMO. For instance, if E_HOMO = -11.0 eV and E_LUMO = -9.0 eV , possible choices for the intervals are: occupied = [-14.d0,-10.d0] and unoccupied = [-9.9d0,-6.d0]. Transitions will be calculated only between the states that are found within these intervals. The spectrum can also be calculated within for DRIVER = avrg_confgs, in which case the arguments presented above, for DOS parameters, apply likewise. The avrg_confgs method is recommended for flexible molecular structures, molecules in solution, and systems at high temperatures.

4.10 Genetic_Alg and CG optimization parameters

The Optimization flags and variables control the running of the Genetic Algorithm (GA) and Conjugate Gradient (CG) optimization procedures. These routines are used to optimize the semiempirical parameters of the Slater-type orbitals and the Wolfsberg-Helmohlz coupling constants of the extended Hückel hamiltonian matrix elements. They parameters are

- Pop_Size integer variable : determines the size of the ensemble of possible solutions (parameter sets to be optimized). Since the Genetic Algorithm is an stochastic method, large values for this variable will increase the sampling, however the actual value will depend on the size of the system. It is recommended to choose a medium size value to

perform exploratory optimizations, with the goal of finding out the best definition for the cost (penalty) function, that is, the cost function that will most efficiently lead to the desired parameter set. After the preliminary analysis larger values of Pop_Size should be used.

- N_generations integer variable : determines the number of recurrent iterations applied to select the best parameters.

- Top_Selection integer variable : determines how many parameter sets, out of Pop_Size, that are preserved to the next generation (iteraction). The selection criteria is to select the sets that minize the penalty (cost) function. If mutation and crossing are allowed it is more efficient to make Top_Selection approximately as half of Pop_Size, otherwise it should be around 10 or 20% of Pop_Size. The top parameters are perserved to the next generation, whereas the remaining parameters are re-generated either by mutation and crossing or by random generation.

- Pop_range double real variable : determines the variation range that will be applied to the input parameter sets. Reasonable values are $0.05 <$ Pop_range < 0.3.

- Mutate_Cross logical variable : if true, mutation and crossing operations are applied to suply the parameter sets from Top_Selection+1 to Pop_Size.

In addition, the conjugate-gradient (CG) method can also be applied after the genetic algorithm (GA) method as a post-optimization procedure to refine the solution:

- CG_ logical flag : If true the CG methos is used as a post-optimizaton procedure. It will use the top selection parameter sets gained from the GA algorithm as a grid to perform CG optimizations from different grid points. The procedure avoids the trapping of the search method in local minima, so much that it is common to find that be best parameter set overall is not proceduced by the best GA parameter. The CG is less stable than the GA method and the performance of the former is dependent on the starting parameter set, therefore it is generally used as a post-optimization procedure. The penalty (cost) function used in both optimization methods is defined in the **evaluate_cost** function, which is located in the **cost_tuning.f** file.

- profiling logical flag : If true the CG routine saves the evolution of the penalty (cost) function along the optimization process in the file **CG-log.dat**; by plotting this data file it is possible to analize the CG optimization.

Chapter 5

File tuning.f

The subroutines contained in this file are intended to provide an easy and versatile method for including *ad-hoc* information about the system's structure and residues; information that is not provided by the input **pdb** and **xyz** structure files. Before describing the way to use it, we describe its data structure.

All the derived data types used throughout the program are defined in the files **types_EHT.f** (module type_m) and **types_MM.f** (module MM_types). The former describes the derived data types used in the quantum mechanical part of the code whereas the variables of the latter are used in the Molecular Mechanics part of the code. Since the parts exchange data during the QM-MM simulations, both the derived data types have similar structures

$$\begin{aligned} \text{type atomic} &\equiv \text{type MM\_atomic} \\ \text{type molecular} &\equiv \text{type MM\_molecular} \\ \text{type universe} &\equiv \text{type MM\_system} \end{aligned}$$

5.1 Ad-hoc tuning of the electronic system

The main subroutine of module tuning_m is **ad_hoc_tuning**, it assigns ad-hoc information to a derived data of type universe. This subroutine has the purpose of including only the extra information, which is not brought by the input files but are necessary to characterize a particular simulation. It can also be used to change the information originally contained in the input files, for a different simulation protocol. Some examples of how to use it are presented below.

- to set a specific hardcore value to all Carbon atoms (AtNo=6) of the system:
  where(univ % atom % AtNo == 6) univ % atom % hardcore = 2.45
- to (re)define the residue properties of the molecule:
  univ % atom(40:50) % residue = "CCC"
- to determine that the electronic wavepacket is located at the PPH residue within the structure: [1]
  where(univ % atom % residue == "PPH") univ % atom % El = .true.
  or, it may happen that the electron wavepacket is occupying initially various residues of

[1] For time-propagation simulations it is mandatory to determine the location of the electronic wavepacket.

the molecular structure:
where(univ % atom % residue == "BP1") univ % atom % El = .true.
where(univ % atom % residue == "BP2") univ % atom % El = .true.

- for electron-hole wavepacket dynamics it is also necessary to determine the location of the hole, which may be in the same location as the electron:
  where(univ % atom % El) univ % atom % Hl = .true.
  or in a different residue:
  where(univ % atom % residue == "BP1") univ % atom % El = .true.
  where(univ % atom % residue == "BP2") univ % atom % Hl = .true.

- to determine that different residues will be taken into account in the calculation of the dipole moment of the molecule:
  where(univ % atom % residue == "ION") univ % atom % DPF = .true.
  where(univ % atom % residue == "BP1") univ % atom % DPF = .true.
  where(univ % atom % residue == "BP2") univ % atom % DPF = .true.
  where(univ % atom % residue == "BP3") univ % atom % DPF = .true.

- to reset residue numbers:
  where(univ % atom % nr <= 4) univ % atom % nr = 1
  or
  univ % atom(40:50) % nr = 2

In summary, different attributes can be ascribed easily as needed. The complete structure of the derived data type **system** is shown below

univ % atomic(:) % { xyz(3), mass, charge, solvation_hardcore, hardcore, polar, my_id, AtNo, Nvalen, nr, copy_No, residue, Symbol, MMSymbol, TorF(3), fragment, solute, DPF, El, Hl } .

Notice that there are two ways of setting the residues of the individual atoms:

1. - via the fourth column of the input file (input.pdb or frames.pdb), then a fragment symbol (letter or number) can be ascriebd to the atom by either of the subroutines **ad_hoc_tuning** or **Setting_Fragments** (both located in the same module);

2. - by setting both residue and fragment in the tuning.f file.

5.2 Ad-hoc tuning of the nuclear system

Subroutine **ad_hoc_MM_tuning**, located in module **module MM_tuning_routines**, works in the same away to attach ad-hoc information to the molecular structure, in regard to the Molecular Mechanics simulations. However, in this particular case there are two instances that can be chosen: instance = General or instance = SpecialBonds.

The first case is similar to the situation described in the previous section, for instance, we can use this subroutine to determine whether an atom is fixed or free to move:

- for anchoring a specific atom:
  where(system % my_id == 50) system % free = .false.

- for anchoring all the atoms that comprise a residue:
  where(system % residue == "C60") system % free = .false.

- for (re)defining the charge of an atom:
  where(system % MMSymbol == "CT") system % MM_charge = 0.2
  where(system % MMSymbol == "YC") system % MM_charge = -0.1

- to define the molecular species to which an atom belongs:
  where(system % MMSymbol == "YC") system % my_species = 3

- an so son for instance="General" in the calling of the subroutine.

If instance = "SpecialBonds" it is possible to define

- special bond parameters as:
  allocate(SpecialBonds(2))
  SpecialBonds(1) % label = "bond_gb16"
  SpecialBonds(1) % kbond0(1) = 392459.2
  SpecialBonds(1) % kbond0(2) = 0.14010
  SpecialBonds(2) % label = "bond_gb53"
  SpecialBonds(2) % kbond0(1) = 392459.2
  SpecialBonds(2) % kbond0(2) = 0.14580

- or special angle paremeters for a bond:
  allocate(SpecialAngs(2))
  SpecialAngs(1) % label = "angle_ga07"
  SpecialAngs(1) % kang0(1) = 459.2
  SpecialAngs(1) % kang0(2) = 110.14
  SpecialAngs(2) % label = "angle_ga27"
  SpecialAngs(2) % kang0(1) = 457.2
  SpecialAngs(2) % kang0(2) = 130.14

The different attributes that can be ascribed to the derived data type **MM_atomic** are shown below

MM_atomic(:) % AtNo
my_id
my_intra_id
my_species
nr
residue
Symbol
MMsymbol
mass
MM_charge
eps
sig
free .

5.3 Debuging utility

The subroutine **debug_MM()** can be called from anywhere, within a module dedicated to MM calculations, to check whether the attributes of the system are correctly determined. The arguments of the subroutine can be either

- variable of the type MM_atomic;
- variable of the type MM_molecular;
- variable of the type MM_system.

A menu will be displayed in the screen and the user may choose from various options to view the attributes of the specific varible. Once the system is correctly defined, this debuging utility is not necessary and its calling line must be deleted for the execution of the simulation.

Chapter 6

Extended Hückel Paramter Optimization

6.1 Semiempirical Parameterization

The matrix elements of the tight-binding extended Hückel hamiltonian, $H^0_{ij} = \kappa_{ij}(H^0_{ii} + H^0_{jj})S_{ij}/2$, are built up from 3 classes of semiempirical parameters: the Wolfsberg-Helmholz coupling constant κ_{ij} (usually set as a unique parameter $\kappa_{WH} = 1.75$ for all the orbitals in the system), the valence state ionization potentials H^0_{ii} and the exponents of the Slater-type orbitals. Typical tight-binding model parameters [1, 2] are not transferable to different molecular structures. However, the overlap matrix elements S_{ij} render the extended Hückel theory quite transferable [3, 4, 5, 6]. A transferable semiempirical model should be capable of describing – for the same material – the delocalized properties of bands structures for bulk solids as well as the local chemical properties of surface states and clusters [4, 7]. Even so, there is no guarantee that the parameters are transferable to different chemical environments. The use of unsuitable parametrisation yields frontier molecular orbitals in the wrong order or unphysical values for transition energies and dipole moments. Molecular mechanics, which is essentially a semiempirical method, uses specific force fields for the same atom at different environments, that is, different atom types represent the same chemical element in distinct environments.

6.2 File cost_tuning.f

Both the genetic algorithm and the conjugate gradient optimization procedures aim at minimizing a penalty (or cost) function that represents the difference between a set of reference properties and the corresponding results provided by the model. The function **evaluate_cost**, located in the the file **cost_tuning.f** is used to define and calculate the cost corresponding to each parameter set throughout the optimization procedure.
The penalty function is calculated as root-mean-square deviation among different quantities, that is

$$evaluate\_cost = \sqrt{|chi^*(:) * chi(:)|} \ ,$$

where the complex vector $chi(:)$ is multiplied by weight factors, $chi(:) = chi(:) * weight(:)$.
There are many different ways to calculate the elements of the vector $chi(:)$. Some of the options are described below

- energy differences between frontier orbitals i and j:
  chi(:) = (OPT_UNI%erg(j) - OPT_UNI%erg(i)) - ΔE_{ij}
  several energy differences can be used simultaneously

- Mulliken population analysis. The function **Mulliken**
  Mulliken(GA , basis , MO , atom , AO_ang , EHSymbol , residue)
  can be used to calculate the Mulliken electronic population according to several criteria, which include the molecular orbital (MO), the atom, the Slater type orbital angular momentum (AO_ang), the Extended Huckel type of atom (EHSymbol) and the residue. The criteria can be set independently or in conjunction to determine the corresponding orbital population. Some examples are

  - chi(:) = Mulliken(OPT_UNI,basis,MO=100,residue="APT") - 0.50d0, for setting the 50% of the electronic population of the molecular orbital MO=100 at residue="APT"
  - chi(:) = Mulliken(OPT_UNI,basis,MO=100,EHSymbol="SAP"), for forcing the reduction of electronic occupation in all atoms defined by the atomic type EHSymbol="SAP". If the intention is to do the same for molecular orbitals 100 to 102, simply write MO=100:102.
  - chi(:) = Mulliken(OPT_UNI,basis,MO=100:102,EHSymbol="SAP", residue="ATP"), to apply the above criteria in a single Mulliken calculation.
  - chi(:) = Mulliken(OPT_UNI,basis,MO=100,EHSymbol="Si") * (zi), for compelling on the molecular orbital MO=100 the increase of charge occupation on the Si atoms. The factor $zi = i$ is the imaginary unit, which will be responsible for a negative contribution that will decrease the penalty (cost) function.

- the dipole moment:
  chi(:) = DP(i) - REF_DP(i)
  where the REF_DP(1:3) contains the reference dipole moment vector components and DP(1:3) contains the calculated components.

Each of the elements of the vector chi(:) can be multiplied by a weight factor to ascribe priorities among the different cost components.

6.3 File input-GA.dat

The optimization utility tries to find the best Extended Hückel (EH) parameters to describe a set of atomic elements, regarding some user defined reference properties of the system, as described in section 6.2. The file **input-GA.dat** is used to specify the atomic types that will undergo the optimization process together with the corresponding EH parameters to be optimized. An example of the **input-GA.dat** file is presented below.

MMsymbol	s	p	d	IP	zeta1	zeta2	k_WH
CAP	1	1	0	1	0	0	1
NAP	1	1	0	1	1	0	1
SAP	0	1	0	0	1	1	0

In this example the atoms designated by a special MMSymbol label will be optimized: they are specific carbon atoms (CAP), nitrogen atoms (NAP) and sulphur atoms (SAP). Notice that the other chemical elements that comprise the molecular structure will not have their original EHT parameters optimized if they are not designated by those particular MMSymbols, even if they are elements of the same type.

- For the CAP case, the valence ionization potential (diagonal elements H_{ii}) and the the Wolfsberg-Helmholz coupling constant (κ) associated with the s and p valence STO orbitals will be taken into account in the optimization process.
- For the NAP case, the valence ionization potential (diagonal elements H_{ii}), the ζ exponent of the single STO orbital and the the Wolfsberg-Helmholz coupling constant (κ) associated with the s and p valence STO orbitals will be taken into account in the optimization process.
- For the SAP case, the two ζ exponents of a double zeta linear combination of STO orbitals will be optimized for the p valence orbital only.

Chapter 7

The output files

Some of the output files are saved during the execution of the program in a temporary directory named tmp_data. Such files are produced by the time-propagation drivers, in which cases the completion time of the job may take several days, so that the user can follow status of the job by checking the output files in the tmp_data directory. However, this directory is erased and recreated every time the program is executed. The other files are written in the same directory of the executable.

The output files generated by the code are:

1. structure.log : summarizes the information about the molecular system, as read from by the program. It is used for checking whether the program is reading the data from the input files correctly. Always produced.

2. structure.pdb : similar to structure.log, but in pdb format, so that the user can visualize the structure. Always produced.

3. system-ergs.dat : contains the energy eigenvalues of the hamiltonian. Can be used to localize the HOMO and LUMO of the entire molecular system. Always produced.

4. TDOS.dat : Total density of states of the system. All the DOS files are comprised of four columns: the energy, the total DOS convoluted by gaussians, the total DOS peaks and the integrated DOS.

5. PDOS-⟨residue-name⟩.dat : The density of states projected on each of the residues that comprise the molecular system. Same data structure as the TDOS file.

6. spectrum.dat : The oscillator strength corresponding to the optical spectrum of the system. Same data structure as the TDOS file. Produced only when the keyword **spectrum** is set as true in the file **parameters.f**.

7. Security_copy.dat : backup for restarting the job if it is killed unexpectedly. The resume execution the file Security_copy.dat must be moved to Restart_copy.dat. To resume execution the logical flag **restart** must be set as true in the file **parameters.f**.

8. OPT_eht_parameters.output.dat : New Extended H"uckel parameters that are generated after the optimization procedure is concluded. If the output parameters are satisfactory, the name of the file must be changed to OPT_eht_parameters.input.dat, so that the program can read them if the logical flag **OPT_parms** are set as true in the file

parameters.f. In the default case, OPT_parms = F_ , the program reads all the EHT parameters and the supplementaru information from the file **my_eht_parameters.dat**.

9. CG-log.dat : data about the progress of the conjugate-gradient optimization procedure.

10. el_survival.dat : the time dependent electronic occupation on the fragments of the system, as calculated from the wavepacket time propagation. The file is comprised by several (at least 4) columns, which are in the left-to-right order: the time in picosenconds, the electronic density in the donor fragment (always the second column), the electronic densities on the other fragments that constitute the system, and, always at the end the total electronic density summed over all the fragments of the system (this column must be continually equal to unity). This file is always produced if one of the time-propagation drivers are chosen.

11. hl_survival.dat : the same as for el_survival.dat, but it is produced only if n_part=2.

12. dipole_dyn.dat : dynamics of the dipole moment corresponding to the molecular fragment defined as **univ%atom(:)%DPF** = T_. The file is comprised of five columns: the time in picoseconds, the x, y and z components of the dipole moment and the module of the dipole moment.

13. el_FMO-ergs.dat : energy eigenvalues of the hamiltonian constructed from the molecular fragment defined as donor (D), that is the molecular fragment where the initial electronic wavepacket is located. These eigenvalues can be examined to determine the occupied and unoccupied molecular orbitals of the photoexcited acceptor fragment and, therefore, set the variable **initial_state** that defines the wavepacket at time zero. This file is written if one of the time-propagation drivers are chosen.

14. hl_FMO-ergs.dat : the same as for el_FMO-ergs.dat, but it is produced only if n_part=2.

15. el_UNI-ergs.dat : the energy eigenvalues of the electron hamiltonian built up from the entire molecular system. If the logical flag **Coulom_** is set as true, the electron and hole hamiltonians are in general different because of the mutual electron-hole interaction. This file is produced if n_part=2.

16. hl_UNI-ergs.dat : the same as for el_UNI-ergs.dat.

17. el_MO_shot1234.cube : file in gaussian cube format for visualizing the isosurface of molecular orbital 1234. The actual number i of the orbital to be visualized is determined by subroutine **Gaussian_Cube_Format(UNI%L(i,:) , UNI%R(:,i) , i , 0.d0)**, that can be called from the drivers **diagnostic** or **Genetic_Alg**. These are the kind of cube files that are produced by the static drivers, they convey the negative and positive phases of the orbital wave functions that can be plotted with different colors.

18. el_dens_shot1234.cube : file in gaussian cube format for visualizing the isosurface of time dependent electronic wavepacket. In this case the number 1234 represents the time frame of the propagation. Such files represent actually cube files for the density probability and, therefore, its isosurface has only the positive component. These files are produced by the time-propagation drivers, only if the logical flag **GaussianCube** is set as true in the **parameters.f** file.

19. hl_dens_shot1234.cube : the same as for el_dens_shot1234.cube, but produced only for n_part=2.

20. NetCharge.inpt : written in the tmp_data directory, it contains the time-dependent Mulliken atomic charges. Produced by the dynamics drivers if the loogical flag **NetCharge** is set a true in the file **parameters.f**.

21. DipoleSize.inpt : written in the tmp_data directory, it contains the time-dependent induced atomic dipole moments. Produced by the dynamics drivers if the loogical flag **CH_and_DP** is set a true in the file **parameters.f**.

22. tmp_data/DipoleFrames.pdb written in the tmp_data directory, it is meant to be used together with either one of the the files NetCharge.inpt or DipoleSize.inpt, for visualization of the time dependent atomic charges or polarization. The package also provides a script file named **script-vmd** that reads the inpt files and the DipoleFrames.pdb. By using the VMD (Visual Molecular Dynamics) molecular graphics software, one can view the time dependent atomic charges or polarization doing **vmd -e script-vmd**.

Chapter 8

Molecular Mechanics

Description of the Molecular Mechanics Method, its implementation and execution will be presented here.

Bibliography

[1] D. A. Papaconstantopoulos, *Handbook of the Band Structure of Elemental Solids*, Plenum, New York, 1986.

[2] M. J. Mehl and D. A. Papaconstantopoulos, *Phys. Rev. B*, 1996, **54**, 4519.

[3] J. S. Cerda, *Phys. Rev. B*, 2000, **61**, 7965.

[4] D. Kienle, J. I. Cerda and A. W. Ghosh *J. Chem. Phys.*, 2006, **100**, 043714.

[5] A. Zienert, J. Schuster and T. Gessner *J. Phys. Chem. A*, 2013, **117**, 3650.

[6] Table of parameters for Extended Hückel calculations, collected by Santiago Alvarez, Universitat de Barcelona (1995).

[7] D. Kienle, K. H. Bevan, G. -C. Liang, L. Siddiqui, J. I. Cerda and A. W. Gosh, *J. Appl. Phys.*, 2006, **100**, 043715.