fromage Documentation

Release 1.0

Miguel Rivera, Michael Dommett, Rachel Crespo-Otero

May 05, 2020

GENERAL

fromage (FRamewOrk for Molecular AGgregate Excitations) is a library designed to support investigation of photophenomena in molecular crystals. Among the features are:

- Cross-program ONIOM-style calculations with different electrostatic embedding methods
- Location of energy minima and minimal energy conical intersections
- Command line geometry manipulation tools
- Evaluation of exciton coupling values using multiple schemes (under development)

The current version is 1.0

To cite the use of the program, please use:

Rivera, M., Dommett, M., Sidat, A., Rahim, W., Crespo-Otero, R. fromage: A library for the study of molecular crystal excited states at the aggregate scale. *J Comput Chem* 2020; 1– 14. https://doi.org/10.1002/jcc.26144

And if you are using one of the ONIOM implementations:

Rivera, M., Dommett, M., Crespo-Otero, R. ONIOM(QM:QM) Electrostatic Embedding Schemes for Photochemistry in Molecular Crystals. *J. Chem. Theory Comput.* 2019; 15, 4, 2504-2516 https://doi.org/10.1021/acs.jctc.8b01180

CHAPTER

ONE

FEATURES

- Cross-program ONIOM calculation
- Interface with Gaussian, Turbomole, Molcas
- Reads output from CP2K, Quantum Espresso
- Mechanical, electrostatic, Ewald and Self-Consistent Ewald embedding
- Unique dimer detection
- Excitonic coupling via diabitazation
- Voronoi volume evaluation and visualisation

CHAPTER

TWO

REQUIREMENTS

- UNIX type system
- Python 2.7+ or 3.3+
- numpy (installed automatically)
- scipy (installed automatically)
- SWIG
- Ewald (custom fork; only necessary for Ewald embedding)

CHAPTER

THREE

INSTALLATION

1. Clone the repository to wherever you want to install it:

```
cd /path/to/dir/
git clone https://github.com/Crespo-Otero-group/fromage.git
cd fromage/
```

2. Install

```
sudo pip install .
```

3. Set your environment variables

In your `.bashrc`, add

```
export FRO_GAUSS=g16
export FRO_EWALD=Ewald
```

If you are using different binaries for Gaussian or Ewald, change accordingly.

Voilà!

CHAPTER

FOUR

ESSENTIAL GLOSSARY

Throughout this documentation, we employ terms and abbreviations which may not be immediately clear to the new user. Herein is a compiled list of important terms accompanied by brief descriptions. Further discussion for many of these terms is included in the user documentation.

4.1 Cluster models

In order to take into account the environmental energetical contributions to an excited molecule inside a crystal, we offer several kinds of non-intrusive embedding methods:

ONIOM Our own N-layered Integrated molecular Orbital and Molecular mechanics. An extrapolative (subtractive) embedding paradigm for multilevel calculations[6][16][11]

EC Embedded Cluster. The application of the ONIOM method with electrostatic embedding to a cluster of molecules taken from their crystalline positions

EEC Ewald Embedded Cluster. Similar to EC but using an electrostatic embedding scheme aimed at reproducing the Ewald potential of the crystal

SC-EEC Self-Consistent Ewald Embedded Scheme. The EEC model where the embedding charges are computed self consistently

Mechanical embedding ONIOM embedding where no point charges are included and the intersystem electrostatic interaction is purely at low level

Electrostatic embedding ONIOM embedding where the point charges from the surrounding cluster are included in the Hamiltonian of the high level of theory to be treat Coulombic terms in the excited state

4.2 Calculations

In the expression of the 2-level ONIOM energy, the system under scrutiny is partitioned into two regions:

Model system The central molecule(s) whose properties are under scrutiny

Real system The complete cluster of molecules taken from their crystalline positions, including and surrounding the model system

By nature, 2-level ONIOM combines two levels of theory:

High level A level of theory which describes the required properties of the model system. In photochemistry this will often be an excited state method

Low level A less computationally expensive level of theory than the high level. This will typically be a ground state method

When optimising the geometry of a molecule using one of the cluster models, a few concurrent calculations must be carried out.

mh Model system high level calculation

ml Model system low level calculation

rl Real system low level calculation

mg Model system calculation in the ground state of the high level

4.3 General photochemistry

MECI Minimum Energy Conical Intersection. The minimum energy point in the crossing seam between potential energy surfaces. This point is located in **fromage** by using the penalty function method.[15]

Exciton coupling A measure of the splitting of the excited state energy levels due to the formation of a molecular dimer[4]

CHAPTER

FIVE

TUTORIAL FOR GEOMETRY OPTIMISATION

This tutorial will guide you step by step through a typical ONIOM calculation done with **fromage**. This page is meant to be self contained but for a refresher on the abbreviations used, visit the *glossary*. Everything discussed herein is elaborated upon in the rest of the documentation. The example calculation is rather computationally costly and time intensive. This is because we choose an model system with a distinctive excited state minimum and conical intersection geometry.

The example for this tutorial is the investigation of the emission properties of the (2-hydroxyphenyl)propenone (HP) crystal using the Ewald Embedded Cluster model (EEC) and Gaussian. The extension of this protocol to other programs is straightforward. The following steps will involve calculating the geometries and associated energies of the ground state, first excited state and S_1 - S_0 Minimal Energy Conical Intersection (MECI).

The necessary input files are in the `tutorial` directory. We follow the ONIOM nomenclature for the different regions of the embedded cluster. In other words the whole cluster is the *"real system"* and the central molecule is called the *"model system"*. We abbreviate the principal calculations involved in the ONIOM method as `mh` for the *model system* at a high level of theory, `ml` for the *model system* at a low level of theory and `rl` for the *real system* at a low level of theory. For the location of conical intersections, a fourth calculation is typically involved, `mg`, which indicates the *model system* at the high level of theory but in the ground state.

5.1 Calculation set up

The initial step is to generate the Ewald charge background as well as all of the necessary files needed for geometry optimisation.

5.1.1 Input

Make a directory where the preparatory calculation will take place:

```
mkdir opt_1/
cd opt_1/
cp path/to/fromage/tutorial/* .
```

The input files that you have just copied are:

- **`cell.xyz`** A file containing the optimised positions of the atoms in the unit cell
- **`config`** The **fromage** configuration file, described below
- **`high_pop.log`** A Gaussian output file of a population analysis for one molecule using the high level of theory (in this case PBE 6-31G*)
- **`low_pop.log`** The same but for the low level of theory (here, HF STO-3G)

- **`*.template`** The template files for your upcoming ONIOM calculation. This is includes `mh.template`, `ml.template`, `rl.template` and `mg.template`. Note that the checkpoint file name is `gck.chk` in all cases and that the levels of theory match those of the population log files

5.1.2 Execution

If your installation was successful, all of the **fromage** scripts should be in your system path already. In that case, running the program simply involves typing:

```
fro_prep_calc.py
```

5.1.3 Output

After a few minutes, you will be greeted with a series of outputs:

- **`prep.out`** Output file with some information about the setup calculation
- **`mol.init.xyz`** The initial position of the *model system*
- **`shell.xyz`** The molecules surrounding the *model system*
- **`mh/ ml/ rl/ mg/`** Directories containing a `.temp` file each. For example `mh/` contains `mh.temp`
- **`ewald/`** The directory where the ewald calculation is run. The outputs in here are not important for this tutorial

5.2 Geometry optimisation

We will calculate the geometries and associated energies of the ground state minimum, first excited state minimum and S_1 - S_0 Minimal Energy Conical Intersection (MECI).

5.2.1 Ground state

Input

These are all the files needed for the geometry optimisation. Most of them were already generated from the previous step.

- **`fromage.in` (not required for this tutorial)** The input file which contains the specifications for the geometry optimisation. This tutorial uses default specifications, so this is not required
- **`mol.init.xyz`** See above
- **`shell.xyz`** See above
- **`mh/ ml/ rl/`** Directories containing their respective `.temp` files

Execution

An important part of calculations in **fromage** is the assignment of memory to each component calculation. Some times, depending on the system size and the combination of methods used, `rl` will need more memory than `mh`. Make sure to adapt the memory requested in all three `.temp` files to match the capacity of your system.

When this is ready, submit your job with the command:

```
fro_run.py
```

On the command line or in your job queue.

Output

You can expect this calculation to take a few hours depending on your computational resources. The convergence criterion of the optimisation is very strict by default so it is up to the user's judgement whether they wish to abort the calculation once they have achieved a satisfactory precision.

- **`fromage.out`** The main output file. This contains information about the energies and gradients at each step of the optimisation
- **`geom_mol.xyz`** The positions of the *model system* throughout the optimisation
- **`geom_clust.xyz`** The position of the real system throughout the optimisation. Only the *model system* will change

`geom_mol.xyz` should show very slight rearrangement of the molecule since its Gaussian-optimised ground state geometry is close its crystal.

Vertical excitation

To calculate the vertical excitation, make a new directory called `exci/` and copy the `mh.com` file to it (it should now contain the optimised geometry):

```
mkdir exci/
cp mh/mh.com exci/
cd exci/
```

Then edit the `mh.com` file to remove the `force` keyword and add `td(nstates=5,root=1)`. Now run Gaussian (or use a submission script):

```
g16 mh.com
```

This will give the excitation energies of the first five excited states, easily accessible with a judicious `grep`:

```
grep 'Excited State' mh.log
```

The output will look like this:

```
Excited State   1:      Singlet-?Sym    4.1338 eV  299.93 nm  f=0.6373  <S**2>=0.000
Excited State   2:      Singlet-?Sym    4.3687 eV  283.80 nm  f=0.0068  <S**2>=0.000
Excited State   3:      Singlet-?Sym    4.5466 eV  272.70 nm  f=0.0760  <S**2>=0.000
Excited State   4:      Singlet-?Sym    5.4041 eV  229.43 nm  f=0.0318  <S**2>=0.000
Excited State   5:      Singlet-?Sym    5.8322 eV  212.59 nm  f=0.0431  <S**2>=0.000
```

5.2.2 First excited state

We now wish to optimise the geometry of the molecule in the first excited state. The procedure will be almost identical to the one for the ground state optimisation but with an added keyword to `mh.temp`.

Input

First, copy the whole `opt_1` directory to conserve the ground state data. Presuming you are still in `opt_1/exci/`, just type:

```
cd ../../
cp -r opt_1/ opt_2/
cd opt_2/
```

Now edit the file `mh/mh.temp` to add the keyword `td(nstates=1,root=1)`.

And edit your `mol.init.xyz` to match the last geometry in `geom_mol.xyz` from your `opt_1/` directory.

Execution

As usual, type:

```
fro_run.py
```

Or submit it to your job scheduler.

Output

This should typically take longer than your ground state calculation if you have succeeded in setting it up in a way that the limiting calculation is `mh`.

As described above, you will receive `fromage.out`, `geom_mol.xyz` and `geom_clust.xyz`.

This time, you should be able to see the excited state proton transfer in `geom_mol.xyz` as the optimised structure is in keto form.

5.2.3 MECI

Input

One final time, copy the whole directory:

```
cd ..
cp -r opt_2/ opt_3/
cd opt_3/
```

Edit the `mol.init.xyz` file with the final geometry of `opt_2/geom_mol.xyz`.

And in `fromage.in`, add a line at the bottom `bool_ci 1`. This turns on MECI search. Keep in mind that this calculation will use `mg` so change the memory requested in all of your `.temp` files accordingly.

Execution

Again:

```
fro_run.py
```

And wait however long it takes.

Output

The usual `fromage.out`, `geom_mol.xyz` and `geom_clust.xyz` will be generated.

`fromage.out` will contain different information, pertaining to the value and gradients of the penalty function which is being minimised instead of the energy.

CHAPTER

SIX

LICENSE

fromage uses the MIT license cited below:

CHAPTER

SEVEN

CONTACT

fromage is developed at Queen Mary University of London by the Crespo-Otero group.

For anything question or request, you may contact:

Dr. Rachel Crespo-Otero

r.crespo-otero@qmul.ac.uk
Queen Mary University of London
Mile End Road, London
E14NS

Miguel Rivera

m.rivera@qmul.ac.uk
Queen Mary University of London
Mile End Road, London
E14NS

Michael Dommett

m.dommett@qmul.ac.uk
Queen Mary University of London
Mile End Road, London
E14NS

CHAPTER

EIGHT

REFERENCES

CHAPTER

NINE

ONIOM CLUSTER MODELS

9.1 Embedded Cluster

In order to investigate the local photochemical behaviour of molecular crystals, it is practical to use a multiscale method which partitions the system in an excited and a ground state region.

First, a molecule (or several) is selected from the unit cell to become the *model system*. It is placed in the middle of a large supercell from which a surrounding shell of molecules is selected, producing a cluster (the *real system*). A molecule is selected if any of its atoms fall within a chosen distance of the centroid of the model system.

With this partitioned cluster, we use the *ONIOM* energy expression[6] to recover a multiscale energy:

$$E_{ONIOM} = E_{mh} + E_{rl} - E_{ml}$$

For straightforward *mechanical embedding*, the terms E_{mh}, E_{rl} and E_{ml} are simply the energy of the *model system* at a high level of theory, the *real system* at a low level of theory, and the *model system* at a low level of theory.

However this scheme only includes intersystem interactions at the low (ground state) level of theory. To include intersystem Coulombic interactions in the excited state, we use point charges. This is called *electrostatic embedding* in the general *ONIOM* literature and the *Embedded Cluster (EC) model* in this implementation.

The *mh* calculation is embedded in point charges located at the atomic sites of the surrounding cluster molecules. The value of these point charges is not uniquely defined and a discussion is offered in a different *section*. In order to avoid the double counting of electrostatic interactions, point charges must also be included in the *ml* term.

Embedded cluster models have successfully been used by Ciofini's group in characterising excited states in molecular crystals.[17][18][19]

9.2 Ewald Embedded Cluster

The *EC* method described above represents short range interactions to a reasonable extent. However the long range interactions, which are predominantly electrostatic, are completely omitted. In fact they cannot be approximated by increasing the size of the real system because the Madelung sum is conditionally convergent[13]. To remedy this, we use an Ewald embedding scheme[14][7] in *mh* where a large array of point charges at lattice positions is generated and then fitted to match the Ewald potential. The combination of point charge Ewald embedding and *EC* is the *Ewald Embedded Cluster (EEC) method*.

This method requires some justification. First of all, the long range electrostatic charges of the crystal are not cancelled in the *ml* term. If we wished, we could embed *rl* and *ml* in Ewald fitted point charges. However when we perform geometry optimisation, the surrounding cluster is fixed in place. Therefore the additional computation of the Ewald point charges in the ground state Hamiltonians would only add a correcting constant term.

Another first-glance objection is that the *mh* charges from the surrounding molecules which were included in the *EC* model have potentially been modified to match the Ewald potential, thus rendering the cancellation of ground state interactions by the embedding of *ml* inexact. However by definition the Ewald potential contains the totality of the Coulombic potential of the crystal, both short and long ranged. Furthermore a spherical region of the Ewald point charge array of a chosen radius can be chosen to remain of fixed charge, providing a 'buffer zone' from any highly deviated charges which might break the point charge approximation.

9.3 Self-Consistent Ewald Embedded Cluster

A major omission from the *EC* and *EEC* models is the electrostatic response to the excitation of the *model system* by the surrounding cluster. To recover mutual polarisation effects, we employ an extreme model where the entire crystal is excited at an electrostatic equilibrium. The model system is embedded in Ewald point charges as in *EEC* at the optimised ground state position. A population analysis is carried out on the model system whose charges are then redistributed in the embedding supercell and gain fitted to the Ewald potential. This loop is repeated until self-consistency.

The method is adapted from the work of Wilbraham *et al.*[23][20] Self-consistent Ewald embedding schemes were previously used in the determination of NMR parameters[22]

This scheme is termed the Self-Consistent Ewald Embedded Cluster (*SC-EEC*-S_1). It accurately represents short range electrostatic interactions from a mutually polarising delocalised excitation. Alternatively, the self consistent loop can be performed in the ground state which would give similar results to *EEC* (*SC-EEC*-S_0).

CHAPTER

TEN

EWALD POINT CHARGE FITTING

Traditional electrostatic embedding for cluster models truncates the terms of the Madelung summation up to the range of the cluster. This can sometimes be ill advised since the Madelung summation is known to be slowly and conditionally convergent.[13]

A fast convergent reformulation of the Madelung summation is the Ewald summation, expressed as a sum of both real and reciprocal space terms:

$$V^{Ewald}(\mathbf{r}) = \sum_{\mathbf{L}s} q_s \frac{\mathrm{erfc}(\gamma|\mathbf{r}-\mathbf{L}-\mathbf{R}_s|)}{|\mathbf{r}-\mathbf{L}-\mathbf{R}_s|} + \frac{4\pi}{v_c} \sum_{\mathbf{G}\neq\mathbf{0}} \frac{1}{G^2} e^{-G^2/4\gamma^2} \left[\sum_s q_s e^{i\mathbf{G}(\mathbf{r}-\mathbf{R}_s)} \right]$$

Where $\mathbf{L}$ are the lattice translations, erfc is the complementary error function, γ is the arbitrary Ewald constant, $\mathbf{R}_s$ the atomic sites in the unit cell, v_c the unit cell volume, $\mathbf{G}$ the reciprocal lattice translations and q_s the partial charge of the atoms in the unit cell.[12]

To generate point charges fitted to an Ewald potential, **fromage** relies on the program `Ewald`, developed by Klintenberg, Derenzo and Weber.[14][7]

In this program, an a supercell of point charges is generated. The Ewald potential is computed in a central region and the points in the outer region are fitted to reproduce said potential via direct summation. See citations above for more details.

Ewald point charge embedding has successfully been used to describe excited states in molecular crystals.[8][23][20]

CHAPTER

ELEVEN

MINIMAL ENERGY CONICAL INTERSECTION OPTIMISATION

To optimise conical intersection geometries, the penalty function method of Levine[15] is used, removing the need for nonadiabatic coupling vectors. A function of the averaged S_1 and S_0 energies ($\bar{E}_{1-0}$) and the S_1-S_0 energy gap (ΔE) is minimised:

$$F = \bar{E}_{1-0} + \sigma \frac{\Delta E^2}{|\Delta E| + \alpha}$$

σ is a Lagrangian multiplier and α is a parameter such that $\alpha \ll \Delta E$.

CHAPTER

TWELVE

POPULATION ANALYSIS

12.1 Methods

At the heart of of any point charge embedding scheme is the assumption that the electrostatic potential generated by an atom can be approximated at a certain distance by that of a point charge:

$$\sum_i \frac{q_i}{|\mathbf{r_i} - \mathbf{r}|} \approx \sum_i \int \frac{|\Psi_i(\mathbf{r}')|^2}{|\mathbf{r}' - \mathbf{r}|} \mathbf{dr}'$$

With $\Psi_i(\mathbf{r})$ being the wavefunction of atom i, q_i its point charge value and $\mathbf{r_i}$ its position. This approximation naturally breaks down at small $|\mathbf{r_i} - \mathbf{r}|$ but by including ground state exchange via the *rl* term, this situation is avoided.

The choice of q_i is a non trivial matter. Considerable effort is spent on parametrising forcefields for various applications, however in this case we intend to use the point charges as one-electron terms in the *mh* and *ml* Hamiltonians.

A first approximation can be the use of Mulliken charges. This has the advantage of being trivial to compute in both the molecular and crystal calculations. Unfortunately the well-known basis-set dependence of Mulliken charge values can become a real problem.

Charge-density based methods represent the most direct way of fulfilling the equation above. Hirshfeld and AIM charges[10] fall in this category. RESP are practically defined by the equation, albeit in a radial range around the atoms.

12.2 Crystal or molecular

The population analysis methods described above can be applied to periodic crystal calculations or single molecule calculations. Using the crystal calculation has the advantage of including an interacting charge distribution at equilibrium.

On the other hand, charges from molecular calculations could be more judicious since they can be matched to the level of theory of the molecular calculation that they are embedding. In this way if the high level of theory is TD-DFT B3LYP and the low level of theory is DFT PBE, the charges assigned to *mh* can come from B3LYP and the ones to *ml* from PBE. This is particularly important in order to most totally cancel out intersystem electrostatic interactions from *rl*.

CHAPTER

THIRTEEN

INTERFACES

For geometry optimisation, **fromage** communicates between different quantum chemistry codes. They can be selected in the *fromage.in* by using the *high_level* and *low_level* keywords. The current list of available programs is:

- Gaussian `gaussian`[9]
- Molcas `molcas`[2]
- Turbomole `turbomole`[1]
- DFTB+ `dftb`[3] (under development)

CHAPTER

FOURTEEN

INPUT FILE DESCRIPTION

fromage uses input files at two points of its execution. During the preparatory calculation (using `fro_prep_calc.py`), the `config` file is required. During the actual geometry optimisation (`run_fromage.py`), the file `fromage.in` is read if present.

14.1 config file

All **fromage** input files follow the same input structure. The order of the keywords is irrelevant and blank lines are ignored. The keyword is stated and then its value(s) after any number of whitespaces. Therefore:

```
bond_thresh 1.9
```

Is the same as:

```
bond_thresh      1.9
```

Below are listed the most important keywords available.

name The name of your calculation. Default: `fromage_calc`

a_vec, b_vec and c_vec The lattice vectors in Angstrom. This section has no default value. Example:

```
a_vec         8.9638004303          0.0000000000          0.0000000000
b_vec         0.0000000000         10.5200004578          0.0000000000
c_vec        -3.8748910079          0.0000000000         10.7924653741
```

vectors_file Alternatively the vectors can be stored in a file (called e.g. `vectors`) of the form:

```
8.9638004303          0.0000000000          0.0000000000
0.0000000000         10.5200004578          0.0000000000
-3.8748910079          0.0000000000          10.7924653741
```

In which case the file name should be specified in the config file:

```
vectors_file vectors
```

cell_file The file containing the atomic positions in the unit cell in .xyz format. Default: `cell.xyz`

high_pop_file The file containing the population analysis used for the embedding of `mh`. Default: `gaussian_h.log`

high_pop_program The program used to calculate the above file. Default: `gaussian`

high_pop_method The method of population analysis. "Mulliken" or "ESP". Default: `ESP`

low_pop_file The file containing the population analysis used for the embedding of `ml`. Default: `gaussian_l.log`

low_pop_program The program used to calculate the above file. Default: `gaussian`

low_pop_method The method of population analysis. "Mulliken" or "ESP". Default: `ESP`

bond_thresh The distance between two atoms in Angsrom below which **fromage** will consider the atoms to be bonded together. The definition of bond_thresh can be altered by using the keyword `bonding`. Default: `1.7`

bonding The method which determines whether two atoms are bonded. The options are `dis`, `cov` and `vdw`. `dis` measures the distance between two nuclei whereas `cov` measures the distance from the edge of the spheres of covalent radius and `vdw` from the edge of the sphere of van der Waals radius. Default: `dis`

atom_label The number of the atom in the `cell_file` which belongs to the molecule which will become the *model system*. Several atoms can be specified and must be separated by whitespaces, however they must not belong to the same molecule. Default: `1`

ewald Whether or not to use the Ewald embedding. To turn off, use "false", "no", "off", "zero", "none" or "nan" or any capitalisations. Default: `off`

nchk The number of random points sampled around the model system by `Ewald` to check the accuracy of the fit. Default: `1000`

nat The number of atoms included in the fixed charge region generated spherically by Ewald. Default: `500`

an, bn and cn Multiplications of the unit cell along each cell direction to generate the Ewald supercell. The cell is multiplied 2N times per direction (N positive and N negative). Default: `2`, `2` and `2`

clust_rad The radius in Angstrom used to generate the cluster which will constitute the *real system*. The cluster includes all molecules which fall within the threshold distance from any of the atoms of the central region. Default: `5`

self_consistent Whether or not to use the Self Consistent Ewald Embedding. Be sure to also turn on `ewald`. Default: `off`

sc_temp The template file for the self consistent population analyses. Default: `sc_temp.template`

dev_tol The convergence threshold for the self consistent loop. This corresponds to the average deviation between two successive steps of the loop. Units in e^- Default: `0.001`

damping Damping factor for the self-consistent loop to solve certain convergence problems. Choose a value between 0 to 1 with 0 being no damping and 1 being complete damping (won't get you anywhere). Default: `0`

print_tweak Whether or not to print the tweaked version of the cell with the selected molecule(s) completed and the whole cell centred around its centroid. This is useful for debugging and more involved analysis. Default: `off`

14.2 fromage.in file

The input structure is the same as for `config`.

mol_file File name for the .xyz file containing the inital position of the *model system*. Default: `mol.init.xyz`

shell_file File name for the .xyz file containing the molecules surrounding the *model system*. Default: `shell.xyz`

out_file File name for the output file containing the geometry optimisation information. Default: `fromage.out`

bool_ci Whether or not to optimise for *MECI*. "1" for yes "0 for no. Default: `0`

sigma The Lagrangian multiplier for the penalty function method for the location of *MECI*. Only use if `bool_ci` is on. Defualt: `3.5`

high_level The program used for the high level calculation. The options are `gaussian`, `dftb`, `turbomole` and `molcas`. Default: `gaussian`

low_level The program used for the low level calculation. The options are `gaussian`, `dftb`, `turbomole` and `molcas`. Default: `gaussian`

CHAPTER

FIFTEEN

STRUCTURE MANIPULATION IN FROMAGE

As well as offering cluster model geometry optimisation, **fromage** can be used as a library for the manipulation of molecular structures using Python.

To this end, importing fromage like e.g. `import fromage as fro` supplies the user with a few useful functions such as `fro.mol_from_file` or `fro.dimer_from_file` which allow, in turn, grant the access to a host of useful member functions. To understand how these functions operate, first it is useful to get familiar with the data structure which is used to represent the atoms of the system, the Atom object.

15.1 The Atom object

The Atom object is a contains information about an atom in the system, e.g. position, element or charge. In fact, the Atom object can also represent point charges or any point in space, but as its name reflects, the bulk of its methods are only useful when a chemical system is being represented. In practice, one often deals with molecules or unit cells as opposed to standalone atoms. For this, the Mol object should be employed.

15.2 The Mol object

The Mol object is related to usual Python lists by composition. It is meant to contain lists of Atom objects which represents points in space with associated properties. The Mol object has a multitude of methods which are ready to use for quick geometry manipulation.

Many of these methods take advantage of the modularity of molecular crystals, as such the intermolecular distance which constitutes a bond is a crucial parameter. The default parameters for this should usually be sufficient but in case the geometry being investigated is particularly distorted, the bonding can be defined in different ways: the distance between nuclei, between vdW radii and covalent radii. `set_bonding()` is used to tweak this definition.

The Mol object has an attribute `mol.geom` of the type `GeomInfo`. This is initialised as empty by default to save on computing effort, however useful information can be stored and manipulated such as the atomic positions in numpy array form using `mol.coord_array()`. There are also functions to find the plane which best contains the atomic coordinates with `mol.plane_coeffs()` or three orthonormal vectors giving a fingerprint to the molecular orientation: `mol.axes()`.

15.3 The Dimer object

Dimer objects represent pairs of molecules, each one being a Mol object stored in the attributes `dimer.mol_a` and `dimer.mol_b`. Since Mol objects can be characterised by three axes, the angle between the axes of the two monomers can supply a fingerprint for the dimer. They are supplied by the function `dimer.angles()`.

15.4 IO

The functions for reading files can be found in `read_file` and the ones for writing and editing in `edit_file`. The most common file format for the positions of atoms is .xyz which can be read by `read_pos()` for the final set of coordinates or `read_xyz()` for all sets of coordinates in a list of lists. `write_xyz()` will write a list of atoms to an .xyz file.

CHAPTER

SIXTEEN

MISCELLANEOUS SCRIPTS

16.1 Assign charges

For the assignation of charges to molecular structures, fromage uses their connectivity. This is achieved through a set of tools all based on the definition of a bond from just the information in an .xyz file. Two atoms are considered bonded if their distance is below a certain threshold.

Armed with this connectivity information, the charges assigned to the atoms in one molecule can be redistributed to a whole cluster comprised of the same molecule. One important approximation is that atoms with equivalent connectivity (say the hydrogens in a methyl group) will have the same value: an average of the three original ones.

This redistribution may be useful for other applications outside of fromage, for instance for the assignation of forcefield charges. To address this, the module `fro_assign_charges.py` can be called as a script with as arguments the Gaussian .log file containing the molecular charges, followed by a .xyz file containing the cluster of molecules which need assignation.

In other words, the usage is:

```
fro_assign_charges.py mol.log clust.xyz
```

The output file will be called `out_char` by default. Several options are available including specification of the maximum bond length or the type of charge (ESP or Mulliken). More information can be found using:

```
fro_assign_charges.py --help
```

16.2 Population statistics

It is also often useful to quickly get some statistical information from a population analysis done in Gaussian. For this, `fro_pop_stat.py` can help.

Simply:

```
fro_pop_stat.py mol.log
```

This will output the maximum and minimum charge as well as the average absolute charge, the standard deviation and the total calculated energy.

16.3 Pick a molecule

This script selects a number of molecules from an .xyz file by indicating the atom label of on of the constituent atoms. Something similar can often easily be achieved by a visual program such as Chemcraft or VESTA, however this has the advantage of working through a terminal if it becomes necessary:

```
fro_pick_mol.py clust.xyz 7 13 42
```

This will pick out the molecules containing atoms 7, 13 and 42 from the cluster of molecules `clust.xyz`.

16.4 Manipulate unit cell .xyz files

`fro_uc_tools.py` can be used to produce supercells, molecular clusters and other ouputs based on the unit cell. It requires two inputs: a unit cell file in .xyz format and a vectors file like this:

```
8.9638004303          0.0000000000          0.0000000000
0.0000000000         10.5200004578          0.0000000000
-3.8748910079          0.0000000000          10.7924653741
```

There are many options to choose from so it is suggested to use `fro_uc_tools -h` for instructions.

For example:

```
fro_uc_tools.py cell.xyz vectors -r 12
```

Will use a unit cell geometry file (`cell.xyz`) and its associated lattice vectors (`vectors`) to produce a molecular cluster of all complete molecules with atoms falling within a radius of 12 Angstroms from the origin (`-r 12`). The output is called `cluster_out.xyz` by default.

16.5 Analyse dimers in aggregate geometries

Whether one is presented with a cell stemming from a periodic boundary condition calculation, or an oligomer in a localised cluster, the geometric properties of the dimers present within can help elucidate some of the intermolecular features of the system. The script `fro_dimer_tools.py` can identify the unique dimers in the supplied geometry, taking into account periodicity if relevant. The dimers can further be characterised by the angles between their principal, secondary and normal axes, as well as their centroid-to-centroid distance. As before, many parameters can be altered so using `fro_dimer_tools.py -h` is encouraged.

A suggested use is the following:

```
fro_dimer_tools.py clust.xyz -v -p
```

Will analyse an `.xyz` geometry file of a cluster of molecules (`clust.xyz`), with a verbose output (`-v`), and print (`-p`) all of the unique dimers it finds within the cluster. A file called `dimers.dat` will also be printed with some geometric information related to each dimer and a suggested classification, being `S-S`, `E-F` or `F-F` (side-by-side, edge-to-face or face-to-face).

16.6 Voronoi volume evaluation

It can be useful to determine the available volume of a molecule in an aggregate environment. To do this, one could use the union of the van der Waals volumes of each atom, or the Voronoi volume of the molecule, scaled by van der Waals radii.

```
fro_volumetrics.py clust.xyz -l 13
```

This will produce cube files of the available volume of the molecule containing atom 13 (`-l 13`) within the cluster of molecules (`clust.xyz`). The ouputs are the Voronoi volume (`voro.cube`), the van der Waals volume (`vdw.cube`) and the union of the two (`add.cube`). A file called `volumes` prints the integrated volume of each of the three.

16.7 Exciton coupling evaluation

Exciton coupling evaluation from Gaussian output files can also be carried out, using `fro_coupling.py`. A diabatisation of the Hamiltonian is employed which relies on the calculation of excited state properties such as population analysis or transition dipole moments.[4] More options are also available.

As an example of use, the line:

```
fro_coupling.py -m DIA -p TDM -mf a.log b.log -of dim_ab.log -os 2
```

Will use the diabatisation method (`-m DIA`) and use the transition dipole moment property (`-p TDM`) to read the Gaussian log files of monomer S_1 calculations (`-mf a.log b.log`) and the dimer S_2 calculation (`-of dim_ab.log`) with state of interest S_2 (`-os 2`). The output will show the diabatic Hamiltonian, whose off-diagonal elements are the exciton coupling values.

16.8 Exciton classification

It is sometimes useful to classify excitons as localised, delocalised or charge-transfer. To this end, the script `fro_exciton_classification.py` uses a Mulliken partition scheme to analyse the migration of charge density within a single excitation from a TDDFT or CIS calculation.[5] [21]

To use this, prepare an Gaussian calculation of a dimer in a given excited state using either TDDFT or CIS. Make sure that all of the atoms of one molecule (A) appear before all of the atoms of the second molecule (B). Also make sure to print the `.rwf` file by using the input tag `%rwf=filename.rwf`.

Here is an example Gaussian input for a range-separated hybrid TDDFT calculation up to the fourth excited state:

```
%chk=title.chk
%mem=[X]GB
%nproc=[X]
#p wb97xd 6-31g* gfprint pop=full

title

0 1
     H   0.00 0.00 0.00
     C   0.00 0.00 0.00
     .
     .
```

(continues on next page)

(continued from previous page)

```
     .

--link1--
%chk=title.chk
%rwf=title
#p wb97xd 6-31g* td(nstates=4,root=4) gfprint pop=full IOP(3/33=3) geom=allcheck␣
↪guess=read density=current
```

Now run:

```
fro_exciton_classification.py tddft.log tddft.rwf 3
```

This reads first the Gaussian log file (`.log`), then the read-write file (`rwf`) and analysises the third excited state (`3`). Two indices relating to the electron density migration will be printed, and a classification of the exciton will be suggested as `LOC(A)`, `LOC(B)`, `CT A->B`, `CT B->A` or `Delocalised` (localised on A, or on B, charge transfer from A to B, vice versa, or delocalised). More details can be found in the references above.

CHAPTER

SEVENTEEN

FROMAGE

17.1 fromage package

17.1.1 Subpackages

fromage.fdist package

Submodules

fromage.fdist.fdist module

Module contents

Fast distance

A distance calculator in C++. On certain machines, this can speed up the calculation of a proximity volume by three. For optimal performance, use fdist2 (the distance squared) rather than fdist if possible. This avoids the evaluation of unnecessary square roots.

Module

fdist Contains _fdist and _fdist2 for fast distance calculations

fromage.io package

Submodules

fromage.io.edit_file module

fromage.io.parse_config_file module

fromage.io.read_file module

Module contents

fromage.scripts package

Submodules

fromage.scripts.fro_assign_charges module

fromage.scripts.fro_coupling module

fromage.scripts.fro_dimer_tools module

fromage.scripts.fro_pick_mol module

fromage.scripts.fro_pop_stat module

fromage.scripts.fro_prep_run module

fromage.scripts.fro_run module

fromage.scripts.fro_uc_tools module

fromage.scripts.fro_volumetrics module

Module contents

fromage.utils package

Subpackages

fromage.utils.array_operations package

Module contents

fromage.utils.exci_coupling package

Submodules

fromage.utils.exci_coupling.CATC module

Contains the tools required to calculate the exciton coupling based on Coulomb Atomic Transition Charges method

`fromage.utils.exci_coupling.CATC.`**`CATC_coupling`**(*NTO_1*, *NTO_2*, *coordinates_1*, *coordinates_2*)

Calculates the CATC exciton coupling J based on the Coulomb interaction between Atomic Transition Charges in two molecules

Parameters

- **NTO_1** (*List of floats*) – List of floats of N-atoms for molecule 1
- **NTO_2** (*List of floats*) – List of floats of N-atoms for molecule 2
- **coordinates_1** (*Nx3 array of floats*) – Array of x,y,z coordinates for molecule 1

- **coordinates_2** (*Nx3 array of floats*) – Array of x,y,z coordinates for molecule 2

Returns **J** – Exciton coupling

Return type float

fromage.utils.exci_coupling.PDA module

fromage.utils.exci_coupling.diabatize module

Tools to carry out the diabatization of the adiabatic Hamiltonian to the diabatic form, producing the exciton coupling J on the off-diagonal.

The diabatization scheme used herein is proposed by Troisi et. al PRL 114, 026402 (2015) (https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.114.026402). The major detail for the implementation can be found in the supplementary information of the manuscript.

fromage.utils.exci_coupling.diabatize.**diabatize**(*dimprops*, *monprops*, *energies*)

Uses the either the TDMs or ATCs of the s1 and s2 states of the dimer and the s1 state of the two monomers, to diabatize the adiabatic Hamiltonian of first two excited states (E1 and E1) to the diabatic Hamiltonian, where the off diagonal terms are the couplings J

Accepts 1xn matrices and state energies as inputs

Parameters

- **dimprops** (*numpy array*) – The excited state property for the dimer where dimprops[n] corresponds to the nth excited state
- **monprops** (*numpy array*) – The excited state property for the monomers where monprops[n] corresponds to the nth monomer
- **energies** (*numpy array*) – The energies of the dimer in the excited states in increasing order

Returns **H** – Diabatic Hamiltonian

Return type numpy array

fromage.utils.exci_coupling.elements module

Shortened periodic table data

class fromage.utils.exci_coupling.elements.**element**(*identifier*)

Bases: object

Class representing an element with information on the symbol,name, atomic number and relative atomic mass

symbol

Atomic symbol is passed to the class

Type String

name

Element name

Type String

mass

Relative atomic mass associated with the passed symbol

Type float

number
Atomic number associated with the passed symbol

Type integer

fromage.utils.exci_coupling.read_g09 module

Read and extracts information from gaussian 09 log files

fromage.utils.exci_coupling.read_g09.**read_ES**(*g09_file*, *state*)
Opens a g09 log file and returns the energy difference between the ground and specified state in atomic units

Parameters

- **g09_file** (*Path to g09 log file*) – File path
- **state** (*Integer*) – Excited state number (<1)

Returns **ES** – Energy difference in atomic units between the ground and specified state

Return type float

fromage.utils.exci_coupling.read_g09.**read_NTO**(*g09_file*, *natoms*)
Reads a G09 logfile and returns the atomic centred Natural Transition Charges, obtained via the G09 input line: td=(nstates=1) nosymm Pop=NTO Density=(Transition=1)

Parameters

- **g09_file** (*Path to g09 log file*) – File path
- **natoms** (*Integer*) – Number of atoms

Returns **NTO** – N array of NTO charges in order of atomic positions

Return type np.array

fromage.utils.exci_coupling.read_g09.**read_SCF**(*g09_file*)
Opens a g09 log file and returns the final SCF energy

Parameters **g09_file** (*Path to g09 log file*) – File path

Returns **SCF** – Final SCF energy

Return type float

fromage.utils.exci_coupling.read_g09.**read_TD**(*g09_file*, *state*)
Reads a G09 logfile and returns the Transition Dipole vector for the specified electronic state

Parameters **g09_file** (*Path to g09 log file*) – File path

Returns **TD** – 1x3 array of x,y,z components of TD vector

Return type np.array

fromage.utils.exci_coupling.read_g09.**read_natoms**(*g09_file*)
Opens a g09 log file and returns the number of atoms in the system

Parameters **g09_file** (*Path to g09 log file*) – File path

Returns **natoms**

Return type Integer

fromage.utils.exci_coupling.read_g09.**read_xyz**(*g09_file*)
Open a g09 log file and returns the first geometry in Input orientation

Iterators are used so that the file is not all loaded into memory, which can be expensive.

The function searches for the following text pattern in the log file:

> Input orientation: > ——————————————————————————— > Center Atomic Atomic Coordinates (Angstroms) > Number Number Type X Y Z > ———————————————————————————

And will save the coordinates and the atomic symbols succeeding it

Parameters **g09_file** (*str*) – File path

Returns **coordinates** – Outer list is whole xyz file, each inner list is a line of the file containing the symbol and x,y,z coordinates

Return type List of lists

fromage.utils.exci_coupling.xyz module

Opens xyz files and returns the atoms and cooridnates in various forms

fromage.utils.exci_coupling.xyz.**open_xyz**(*xyz_file*)
Opens an xyz file and returns a list of lists

Parameters **xyz_file** (*String*) – File path

Returns **List of lists** – Outer list is whole xyz file, each inner list is a line of the file containing the symbol and x,y,z coordinates

Return type List of lists

fromage.utils.exci_coupling.xyz.**symbols_from_xyz**(*xyz_list*)
Takes a list of lists containing xyz file lines and the elemental symbols

Parameters **xyz_list** (*List of lists*) – Outer list is whole xyz file, each inner list is a line of the file containing the symbol and x,y,z coordinates

Returns **symbols** – List of atomic symbols

Return type List of strings

fromage.utils.exci_coupling.xyz.**xyz_to_matrix**(*xyz_list*)
Takes a list of lists containing xyz file lines and returns a coordinate matrix

Parameters **xyz_list** (*List of lists*) – Outer list is whole xyz file, each inner list is a line of the file containing the symbol and x,y,z coordinates

Returns **coordinate_matrix** – Nx3 matrix where N=number of atoms

Return type np.ndarray

Module contents

Exciton coupling

Functions supporting the exciton coupling command line utility

fromage.utils.mol package

Module contents

Submodules

fromage.utils.atom module

fromage.utils.calc module

fromage.utils.dimer module

fromage.utils.fit module

fromage.utils.per_table module

This module contains the information from the periodic table

fromage.utils.per_table.**num_to_elem**(*num*)
Return the element symbol according to input atomic number

Parameters **num** (*int*) – Atomic number of the element

fromage.utils.run_sequence module

fromage.utils.volume module

Module contents

Utilities

API supporting the callable scripts in fromage

Modules

atom Defines the Atom object which represents a single point in space with a label, position and potentially a charge and connectivity

calc Defines different Calc classes which run different electronic structure programs

handle_atoms Manipulates lists of Atom objects

per_table Data from the periodic table

volume Tools for the calculation of vdW spheres and Voronoi volumes in a molecular crystal

17.1.2 Module contents

CHAPTER

EIGHTEEN

REFERENCES AND INDICES

- *References*
- genindex
- modindex
- search

BIBLIOGRAPHY

[1] TURBOMOLE V6.2 2010, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from http://www.turbomole.com.

[2] Francesco Aquilante, Jochen Autschbach, Rebecca K. Carlson, Liviu F. Chibotaru, Mickaël G. Delcey, Luca De Vico, Ignacio Fdez. Galván, Nicolas Ferré, Luis Manuel Frutos, Laura Gagliardi, Marco Garavelli, Angelo Giussani, Chad E. Hoyer, Giovanni Li Manni, Hans Lischka, Dongxia Ma, Per Åke Malmqvist, Thomas Müller, Artur Nenov, Massimo Olivucci, Thomas Bondo Pedersen, Daoling Peng, Felix Plasser, Ben Pritchard, Markus Reiher, Ivan Rivalta, Igor Schapiro, Javier Segarra-Martí, Michael Stenrup, Donald G. Truhlar, Liviu Ungur, Alessio Valentini, Steven Vancoillie, Valera Veryazov, Victor P. Vysotskiy, Oliver Weingart, Felipe Zapata, and Roland Lindh. Molcas 8: New Capabilities for Multiconfigurational Quantum Chemical Calculations Across the Periodic Table. *J. Comput. Chem.*, 37(5):506–541, 2 2016. URL: http://doi.wiley.com/10.1002/jcc.24221, doi:10.1002/jcc.24221.

[3] B. Aradi, B. Hourahine, and Th. Frauenheim. DFTB+, a Sparse Matrix-Based Implementation of the DFTB Method. *J. Phys. Chem. B A*, 111(26):5678–5684, 2007. doi:10.1021/jp070186p.

[4] Juan Aragó and Alessandro Troisi. Dynamics of the Excitonic Coupling in Organic Crystals. *Phys. Rev. Lett.*, 114(2):026402, 1 2015. URL: https://link.aps.org/doi/10.1103/PhysRevLett.114.026402, doi:10.1103/PhysRevLett.114.026402.

[5] Rachel Crespo-Otero and Mario Barbatti. Spectrum Simulation and Decomposition With Nuclear Ensemble: Formal Derivation and Application to Benzene, Furan and 2-Phenylfuran. *Theor. Chem. Acc.*, 131(6):1237, 6 2012. URL: http://link.springer.com/10.1007/s00214-012-1237-4, doi:10.1007/s00214-012-1237-4.

[6] Stefan Dapprich, István Komáromi, K.Suzie Byun, Keiji Morokuma, and Michael J. Frisch. A New ONIOM Implementation in Gaussian98. Part I. The Calculation of Energies, Gradients, Vibrational Frequencies and Electric Field Derivatives. *Theochem*, 461-462:1–21, 4 1999. URL: http://linkinghub.elsevier.com/retrieve/pii/S0166128098004758, doi:10.1016/S0166-1280(98)00475-8.

[7] Stephen E Derenzo, Mattias K Klintenberg, and Marvin J Weber. Determining Point Charge Arrays That Produce Accurate Ionic Crystal Fields for Atomic Cluster Calculations. *J. Chem. Phys.*, 112(5):2074–2081, 2 2000. URL: http://10.0.4.39/1.480776%5Cnpapers3://publication/uuid/8C7829E9-67D7-47CA-BE88-0038C9769774http://aip.scitation.org/doi/10.1063/1.480776, doi:10.1063/1.480776.

[8] Michael Dommett, Miguel Rivera, and Rachel Crespo-Otero. How Inter- And Intramolecular Processes Dictate Aggregation-Induced Emission in Crystals Undergoing Excited-State Proton Transfer. *J. Phys. Chem. Lett.*, 8(24):6148–6153, 12 2017. URL: http://pubs.acs.org/doi/10.1021/acs.jpclett.7b02893, doi:10.1021/acs.jpclett.7b02893.

[9] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C.

Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, O. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J. Fox. Gaussian09 revision e.01. Gaussian Inc. Wallingford CT 2009.

[10] Graeme Henkelman, Andri Arnaldsson, and Hannes Jónsson. A Fast and Robust Algorithm for Bader Decomposition of Charge Density. *Comput. Mater. Sci.*, 36(3):354–360, 6 2006. URL: http://linkinghub.elsevier.com/retrieve/pii/S0927025605001849, doi:10.1016/j.commatsci.2005.04.010.

[11] Stéphane Humbel, Stefan Sieber, and Keiji Morokuma. The IMOMO Method: Integration of Different Levels of Molecular Orbital Approximations for Geometry Optimization of Large Systems: Test for N -butane Conformation and S N 2 Reaction: RCl+Cl . *J. Chem. Phys.*, 105(5):1959–1967, 8 1996. URL: http://scitation.aip.org/content/aip/journal/jcp/105/5/10.1063/1.472065http://aip.scitation.org/doi/10.1063/1.472065, doi:10.1063/1.472065.

[12] Lev Kantorovich. *Quantum Theory of the Solid State: An Introduction*. Springer Netherlands, Dordrecht, 2004. ISBN 978-1-4020-2153-4. URL: http://link.springer.com/10.1007/978-1-4020-2154-1, doi:10.1007/978-1-4020-2154-1.

[13] Charles Kittel. *Introduction to Solid State Physics*. Volume 8th ed. John Wiley & Sons, Inc., New York, 2005. ISBN 9788126510450.

[14] M. Klintenberg, S.E. Derenzo, and M.J. Weber. Accurate Crystal Fields for Embedded Cluster Calculations. *Comput. Phys. Commun.*, 131(1-2):120–128, 9 2000. URL: http://linkinghub.elsevier.com/retrieve/pii/S0010465500000710, doi:10.1016/S0010-4655(00)00071-0.

[15] Benjamin G. Levine, Joshua D. Coe, and Todd J. Martínez. Optimizing Conical Intersections Without Derivative Coupling Vectors: Application to Multistate Multireference Second-Order Perturbation Theory (MS-CASPT2). *J. Phys. Chem. B*, 112(2):405–413, 1 2008. URL: http://pubs.acs.org/doi/abs/10.1021/jp0761618, doi:10.1021/jp0761618.

[16] Feliu Maseras and Keiji Morokuma. IMOMM: A New Integratedab Initio + Molecular Mechanics Geometry Optimization Scheme of Equilibrium Structures and Transition States. *J. Comput. Chem.*, 16(9):1170–1179, 1995. URL: http://dx.doi.org/10.1002/jcc.540160911\protect\T1\textbraceleft\T1\textbackslash{}%\protect\T1\textbraceright5Cnhttp://doi.wiley.com/10.1002/jcc.540160911, doi:10.1002/jcc.540160911.

[17] Davide Presti, Fréderic Labat, Alfonso Pedone, Michael J. Frisch, Hrant P. Hratchian, Ilaria Ciofini, Maria Cristina Menziani, and Carlo Adamo. Computational Protocol for Modeling Thermochromic Molecular Crystals: Salicylidene Aniline as a Case Study. *J. Chem. Theory Comput.*, 10(12):5577–5585, 12 2014. URL: http://pubs.acs.org/doi/10.1021/ct500868s, doi:10.1021/ct500868s.

[18] Davide Presti, Frédéric Labat, Alfonso Pedone, Michael J. Frisch, Hrant P. Hratchian, Ilaria Ciofini, Maria Cristina Menziani, and Carlo Adamo. Modeling Emission Features of Salicylidene Aniline Molecular Crystals: A QM/QM' Approach. *J. Comput. Chem.*, 37(9):861–870, 4 2016. URL: http://www.nature.com/articles/ncomms7215http://doi.wiley.com/10.1002/jcc.24282, doi:10.1002/jcc.24282.

[19] Davide Presti, Alfonso Pedone, Ilaria Ciofini, Frédéric Labat, Maria Cristina Menziani, and Carlo Adamo. Optical Properties of the Dibenzothiazolylphenol Molecular Crystals Through ONIOM Calculations: The Effect of the Electrostatic Embedding Scheme. *Theor. Chem. Acc.*, 135(4):86, 4 2016. URL: http://link.springer.com/10.1007/s00214-016-1808-x, doi:10.1007/s00214-016-1808-x.

[20] Davide Presti, Liam Wilbraham, Cecilia Targa, Frédéric Labat, Alfonso Pedone, Maria Cristina Menziani, Ilaria Ciofini, and Carlo Adamo. Understanding Aggregation-Induced Emission in Molecular Crystals: Insights From Theory. *J. Phys. Chem. C*, 121(10):5747–5752, 3 2017. URL: http://pubs.acs.org/doi/abs/10.1021/acs.jpcc.7b00488http://pubs.acs.org/doi/10.1021/acs.jpcc.7b00488, doi:10.1021/acs.jpcc.7b00488.

[21] Kakali Sen, Rachel Crespo-Otero, Oliver Weingart, Walter Thiel, and Mario Barbatti. Interfacial states in donor-acceptor organic heterojunctions: Computational insights into thiophene-oligomer/fullerene junctions. *J. Chem. Theory Comput.*, 9(1):533–542, 2013. doi:10.1021/ct300844y.

[22] Johannes Weber and Jörn Schmedt auf der Günne. Calculation of NMR Parameters in Ionic Solids by an Improved Self-Consistent Embedded Cluster Method. *Phys. Chem. Chem. Phys.*, 12(3):583–603, 2010. URL: http://xlink.rsc.org/?DOI=B909870D, doi:10.1039/B909870D.

[23] Liam Wilbraham, Carlo Adamo, Frédéric Labat, and Ilaria Ciofini. Electrostatic Embedding to Model the Impact of Environment on Photophysical Properties of Molecular Crystals: A Self-Consistent Charge Adjustment Procedure. *J. Chem. Theory Comput.*, 12(7):3316–3324, 7 2016. URL: http://pubs.acs.org/doi/10.1021/acs.jctc.6b00263, doi:10.1021/acs.jctc.6b00263.

PYTHON MODULE INDEX

f

INDEX

S

X