Hands on Simulations of Interfacial Electron Transfer

Snoeberger III, Robert C.

Ding, Wendu
wendu.ding@yale.edu

Rudshteyn, Benjamin
benjamin.rudshteyn@yale.edu

Matula, Adam
adam.matula@yale.edu

Negre, Christian F.A.
christianfannegre@gmail.com

Batista*, Victor S.
victor.batista@yale.edu

Department of Chemistry, Yale University, P.O. Box 208107, New Haven, CT, USA 06520-8107
Energy Sciences Institute, Yale University, P.O. Box 27394, West Haven, CT, USA 06516-7394

Created November 17th, 2010
Updated September 19th, 2016

This 'hands-on' tutorial introduces the reader to simulations of interfacial electron transfer (IET) in sensitized TiO_2 surfaces, allowing the students of CHEM505 to solve Exercise 3 in Lecture2 (colors indicates links that should be clicked on) by using the software package IETsim, developed by Robert C. Snoeberger III in the Batista research group. As discussed in Lecture1 and Lecture2, IET involves the injection of an electron from a molecular adsorbate into an electrode surface, as occurs when photoexcited electrons in an adsorbate molecule are transferred into the conduction band of TiO_2 in the photoanodes of dye-sensitized solar cells (see Fig. 1).

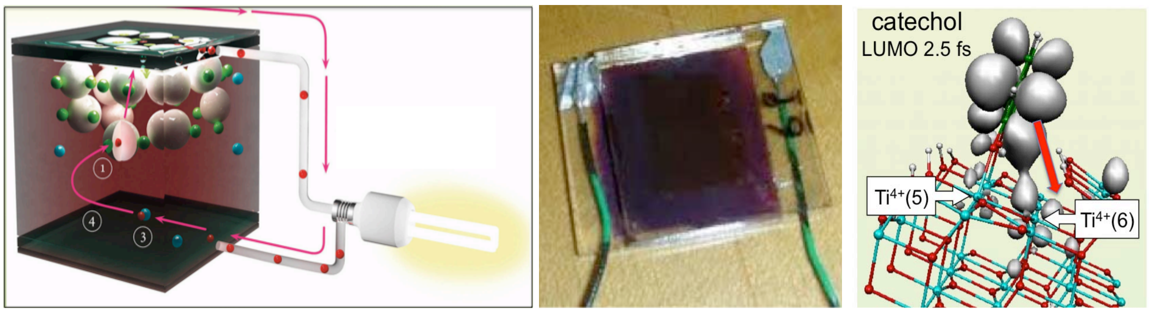

Figure 1: Left panel: Diagram of a DSSC. Middle panel: DSSC based on TiO_2 sensitized WICred-dye (see movie). Right panel: Atomistic description of IET from a catechol adsorbate (green) into the conduction band of TiO_2 (Ti: blue; O:red; H: white; e^- density: gray).

A detailed description of the computational methodology applied for simulations of IET can be found in references [1], [2], [3], and [4]. Here, we outline the methodology as implemented for a fixed nuclear configuration in the IETsim package, and we illustrate it with the specific example shown in Fig. 1 (*i.e.*, IET from a catechol adsorbate into TiO_2). The tutorial aims to encourage students to continue working in the field, or at least to have something of an insider's point of view as the field develops in the next few decades.

IETsim expands the initial electronic state $|\Psi(0)\rangle$ (*e.g.*, the photoexcited electron in the adsorbate molecule) as a linear combination of atomic orbitals $|K_j\rangle$ and evolves it in time by integrating the time dependent Schrödinger equation analytically, as follows: $|\Psi(t)\rangle = \sum_k \langle\phi_k|\Psi(0)\rangle e^{-\frac{i}{\hbar}E_k t}|\phi_k\rangle$. Here, $|\phi_k\rangle = \sum_j C_{j,k}|K_j\rangle$ is the eigenstate with eigenvalue E_k obtained by solving the extended-Hückel (EH) eigenvalue problem $\mathbf{HC} = \mathbf{SC}E$ in the basis of atomic orbitals $|K_j\rangle$ with $S_{ij} = \langle K_i|K_j\rangle$. The EH problem is solved by using the program Yaehmop developed by the research group of Roald Hoffmann at Cornell University. The EH Hamiltonian is an improvement over the familiar Hückel method taught in undergraduate P-chem courses. In the Hückel method, only π orbitals are included in the description of valence electrons, using the basis set of P atomic orbitals to give a qualitative description of conjugated hydrocarbons. The EH hamiltonian expands the Hückel basis set and includes both σ and π orbitals formed by S, P and D atomic orbitals, representing the full valence shell of a wide range of molecular structures. The EH method is a good starting point for simulating IET in sensitized semiconductor surfaces because it is computationally efficient and provides a semi-quantitative description of IET processes central to photoconversion in DSSC's.

The tutorial is organized as follows: Section 2 details the installation of IETsim on a Linux/Unix/Mac system. Section 3 describes the input file. Section 4 shows how to run an example calculation. Section 5 maps the construction of a few key figures, similar to those presented in Reference [1], using the output file from IETsim and highlights the ability of the methodology to provide insight on IET at the molecular level. Section 6 describes the creation of animations depicting the process of injection. Section 7 comments on the accuracy of Fermi levels in EHT as related to this code. Finally, Section 8 describes a general procedure for generating new input files and sensitizer-TiO_2 nanostructures. References are given at the end of the file.

1 The Exercise

Consider a TiO_2 slab with atomic coordinates defined in file TiO2_base.com. Download the software package IETsim and compute:

1. The density of states (DOS) of TiO_2.
2. The DOS of TiO_2 sensitized with catechol covalently attached to the (101) surface, as shown in Fig. 2.
3. The time-dependent electronic population of catechol PMOL(t), when the initial state is defined as the LUMO+1 orbital of the isolated catechol on the TiO_2-anatase (101) surface. Plot the survival amplitude and estimate the rate. Compare your result with Figure 3 (also, Figure 13 in Reference [1]).
4. Simulate the IET from the HOMO orbital of catechol on the TiO_2-anatase (101) surface. Explain why the probability PMOL(t) does not decay to zero.

Solution to the Exercise:

Follow the instructions in the tutorial below to install, compile and run IETsim using the input file provided by the embedded links. The tutorial also provides guidelines to construct figures of the DOS, the time evolution of the electronic density during IET and the time-dependent electronic population.

2 Compiling IETsim

The source code of IETsim is available as a zipped tar file, dynamics-0.4.2.tar.gz. To compile these codes, both the gcc (for C) and gfortran (for Fortran) compilers are needed. Either this tutorial should be set up on a cluster where these are usually installed by default or one should install them using a package, such as Macports, as described in Section 6. The following command will be needed to install them both via Macports:

```
$ sudo port install gcc48
```

A tar file is used to enclose a filesystem tree, preserving the files and their enclosing directories. To open the tar file, type:

```
$ tar -xzvf dynamics-0.4.2.tar.gz
```

where the switches used are x for extract, z for compressed file, v for verbose output, and f indicating the filename will follow. The directory dynamics-0.4.2 will be created and you should move into the directory using the command

```
$ cd dynamics-0.4.2
```

The following directories will be present: cubebuilder, docs, dynamics2, examples, kpoints, and makemovie. Remember, you can use the command

```
$ ls
```

to view a list of files and directories in the current working directory. There are a few codes in the different directories that will need to be compiled before you can complete the tutorial.

We begin with compiling IETsim that includes Yahemop and the module to simulate IET. You need to move into the directory dynamics2, by typing:

```
$ cd dynamics2
```

and you should be able to find the file, Makefile. The Makefile is used to compile the code but before you can do that you will need to change one line in the makefile. The 11^{th} line of the code currently reads

```
PARM_FILE_LOC = \"/home/robert/dynamics-0.4.2/dynamics2/eht_parms.dat\"
```

since it is assumed the tar file was downloaded in directory /home/robert/. You need to change '/home/robert' by the name of the directory where you downloaded the tar file in your filesystem. To find the name of your current working directory, use the command

```
$ pwd
```

which in the current case outputs

```
/home/robert/dynamics-0.4.2/dynamics2
```

but in your case would be different. Copy the analogous output from applying that command in your system and paste in place of /home/robert/dynamics-0.4.2/dynamics2 in your Makefile. This can be done by using a convenient text editor (*e.g.*, nedit, emacs, vi, nano, etc.). Additionally. check the the 25^{th} line of the code, which currently reads

```
FC = gfortran
```

If your gfortran looks different e.g. *gfortran-mp-4.8*, then change the line accordingly

```
FC = gfortran-mp-4.8
```

After editing the Makefile, compile the code by using the command

```
$ make clean ; make all
```

which would be followed by copious amounts of output printed to the terminal. The semicolon lets you run two commands one by one without pressing ENTER in between. Warnings will appear in the output and they can be ignored, but if you see a message indicating an error then the code did not compile properly. You should consult your system administrator for help with compiling the code.

The next code that needs to be compiled is used to extract the density of states (DOS) information from the Yaehmop output file. The name of the code is "fitdos" that will be discussed in the following sections. You can compile the code by changing directory to /dynamics-0.4.2/dynamics2/utils,

```
$ cd utils
```

and using the Makefile to compile

```
$ make clean ; make
```

generating a few warnings that can be ignored.

The final code that needs to be compiled is called cubebuilder2. This code is used to generate cube files that allow the electron density isosurface to be visualized with VMD or GaussView. You should change directory into dynamics-0.4.2/cubebuilder

```
$ cd  ../../cubebuilder
```

and once again use the Makefile to compile the code

```
$ make all
```

which is a much smaller code and should have minimal output. If you were confused by the previous change directory command, that is because

```
$ cd ../
```

moves to the parent directory of the current working directory. Therefore,

```
$ cd ../../
```

moves up two directories in the directory tree from your current working directory.

3 Input file

This section describes the input file of IETsim with emphasis on how to setup calculations of the electronic density of states (DOS) and simulations of IET. The input file typically has the file extention .bind, but file extentions are not as important as in Windows. Inside the .bind file is the molecular structure, unit cell dimensions, kpoint sampling scheme, etc. An example of an input file is available in the directory dynamics-0.4.2/examples. The file can be viewed using 'less', a linux tool used to view files. To view the file with 'less', use the command:

```
$ less catechol_anatase101.bind
```

when your current working directory is the 'example' directory. To quit the 'less' text-viewer, simply type the letter 'q'. If you need to modify the file do it with a text editor (*e.g.*, nedit, emacs, vi, etc.) since unfortunately 'less' is a read-only command.

3.1 Molecular Structure

The begining part of the file 'dynamics-0.4.2/dynamics2/examples/catechol_anatase101.bind' gives the description of the molecular structure coordinates. The structure can either be formatted in cartesian coordinates, which is convenient for a moleculer structure, or in crystallographic coordinates for periodic structures. In this tutorial we will focus on periodic slabs of TiO_2 so we will use crystallographic coordinates. Crystallographic coordinates give the location of atoms in the periodic cell in fractions of the crystal vectors. In Section 8, you will be guided through an example for generating crystallographic coordinates from cartesian coordinates. Lets take a look at the first 10 lines of the file catechol_anatase101.bind and discuss the function of each line,

```
catechol_anatase101.com

Geometry Crystallographic
127
  1   O   0.467536   0.404248   0.202821
  2   O   0.467125   0.596532   0.202983
  3   C   0.505225   0.453304   0.156686
  4   C   0.505082   0.547690   0.156798
  5   C   0.545836   0.410086   0.104319
  6   C   0.545685   0.591452   0.104593
```

The first line is a comment line so you can write anything you want to keep track of the action of the input file. For example, you can enter the name of the .com file generated by GaussView, when you use the graphics interface GaussView to generate the atomic coordinates of the molecular system, as discussed later in this tutorial. The second line is a required blank line. The third line indicates that the geometry will be formatted in crystallographic coordinates. In this case, there are 127 atoms in the file. Actually, there are only 124 because the last 3 are the so-called 'dummy atoms' that specify the crystal unit vectors, as discussed later in this section. The fourth line gives the total number of atoms in the system, including dummy atoms. The next 6 lines describe the first six atoms of the geometry. In this case, they correspond to the oxygens and four of the carbons of catechol. All atomic entries need to follow the same format: atom number, atom symbol and the xyz coordinates.

Now we discuss the dummy atoms at the end of the list of coordinates . The dummy atoms are used to specify the periodic lattice. Their use will be discussed in Section 3.3. Dummy atoms are defined by the atomic symbol ' &', as seen at the end of the input file:

```
123   H   0.704176   0.499003   0.557056
124   H   0.702664   0.748596   0.557086
125   &   1.467536   0.404248   0.202821
126   &   0.467536   1.404248   0.202821
127   &   0.467536   0.404248   1.202821
```

where atoms 125, 126 and 127 are defined as dummy atoms with atomic coordinates equal to the coordinates of atom 1, incremented by 1.0 in the x, y and z coordinates, respectively. In other words, the coordinates of atom 125 come from the first atom as so:
$0.467536 + 1\ 0.404248\ 0.202821 = 1.467536\ 0.404248\ 0.202821$

For atom 126:
$0.467536\ 0.404248 + 1\ 0.202821 = 0.467536\ 1.404248\ 0.202821$
For atom 127:
$0.467536\ 0.404248\ 0.202821 + 1 = 0.467536\ 0.404248\ 1.202821$

3.2 Molecular Charge Keyword

The molecular charge specifies the overall electrostatic charge of the system The format to specify the charge to be zero is given in lines $133 - 134$ of the file 'dynamics-0.4.2/examples/catechol_anatase101.bind':

```
Charge
 0
```

The keywork 'Charge' is followed by a new line with the value of the charge (e.g., 0 in this case).

3.3 Periodic Lattice Keyword

When you have a periodic system, you need to specify the lattice vectors. The way this is accomplished in Yaehmop is by using atoms in the geometry. In the example input file, the 'Lattice' keyword and associated data can be found on lines $136 - 147$. Here are those file lines

```
Lattice
3
1 1 1
1 125
1 126
1 127


Crystal Spec
10.227840  15.128000  22.197750
90  90  90
```

The first line gives the keyword. The second line gives the number of lattice vectors. In this case there is 3, because the system is periodic in 3 dimensions. Actually, the system is periodic in only 2 dimension (2D), but a 2D system can be modeled as a 3D system by making the magnitude of the lattice vector in the non-periodic direction larger than the length of the molecular structure. The third line indicates the number of cell images (the number of replicated units) used when calculating the overlap matrix for the atomic orbital basis. As long as you use a value ≥ 1 you should be fine, but realize that the calculation time will increase with increasing cell images. The next three lines are where those dummy atoms we talked about in the Section 3.1 enter into play. The three lines indicate that the three lattice vectors are the vectors defined as the vector from atom 1 to 125, 1 to 126 and 1 to 127. The position of the dummy atoms were constructed by adding 1.0 to either the x, y, or z coordinate of atom 1. Therefore, the first lattice vector corresponds to the x axis, the second the y axis and finally the third is the z axis.

The second keyword in this section is 'Crystal Spec' which is used to indicate the length of the lattice vectors on the following line. You can see that the length of the third lattice vector is 22.197750, indicating the length of the periodic cell in the z direction. The final line gives the angle between the lattice vectors. A value of 90 degrees between all the unit vectors indicates a rectangular periodic cell.

3.4 IET Dynamics Keyword

Finally, we come to the first keyword used in simulating IET. The "Dynamics" keyword gives the timestep and the length of the simulation. The keyword and needed information is

```
Dynamics
0.1 100.0
```

in the example input file and can be found on lines $149-150$. The first number on the second line is the timestep. The timestep of 0.1 is in units of fs. The second number on the second line is the total simulation time desired. In the example input file, the requested simulation time is 100.0 fs. If the goal is just to get the DOS for a particular system, this simulation time may be decreased to speed up the calculation.

3.5 Adsorbate Keyword

The next keyword used for IET simulations is the 'Adsorbate' keyword. This is where the atom numbers of the adsorbate and the initial adsorbate orbital is specified. The keyword and corresponding information should be formatted as

```
Adsorbate
12 22 1.0
1,2-12
```

which can be found in the example input file on lines $153-155$. The second line gives three numbers which are the number of atoms in the adsorbate (catechol has 12 atoms), the adsorbate orbital to initialize the photoexcited electron (orbital 22 corresponds to the LUMO for catechol, see below), and a weighting factor for the initial orbital. A contribution from a second adsorbate orbital can be introduced by replacing the second line with the following

```
12 22 0.5 23 0.5
```

which gives equal weight to both orbitals and the second orbital introduced is the catechol LUMO$+1$. Catechol has 12 atoms when both oxygens are deprotonated. When catechol binds to the TiO_2 surface, the hydrogens on the oxygens are replaced with titaniums from the surface.

The LUMO orbital is orbital number 22 because catechol with an overall molecular charge of -2 has 42 electrons. This number of electrons comes from the $C_6H_4O_2^{2-}$ molecular formula of this catechol anion. The valence configuration of C is $2s^22p^2$, the valence configuration of H is $1s^1$, and the valence configuration of O is $2s^22p^4$. Therefore, the number of electrons is $6*4+4*1+6*2+2=42$ from C, H, O, and the charge respectively. Remember, there are two electrons per orbital! If the adsorbate were to have a filled valence d-shell, it is not counted towards this number. For instance, Cd^{2+} would contribute no electrons and Se^{2-} would only contribute 8 or alternatively, Cd would contribute only 2 electrons and Se would only contribute 6

The third lines gives the atom number of catechol in the molecular structure. Give the atom numbers separated by commas, or if they are consecutive, use a hyphen as in the example. If you do not know the atom number of the adsorbate, use GaussView to visualize the adsorbate and record the atom numbers.

3.6 Occupation Keyword

The "Occupation" keyword asks the IET module to monitor the probability that the electron is located on a collection of atoms during the evolution. The example given in the example inputfile is

```
Occupation
112
13,14-124
```

which can be found on lines $157-159$. This keyword is followed by the number of atoms on the second line. In the example there are 112 atoms, corresponding to all the atoms except the adsorbate. The third line gives the atom numbers. This keyword is useful if you want to look at the population of the photoexcited electron on particular moieties of the adsorbate during IET.

3.7 Cube Keyword

The 'Cube' keyword tells the IET module that you want to generate cube files of the electron density evolution. The keyword is just "Cube" without further input and can be found in the input file as

```
Cube
```

The auxiliary code cubegen is used to generate the cubefiles but information is required from Yaehmop. The way to information is saved is by the creation of a .wave file when the ”Cube” keyword is present. If you do not want to generate the .wave file (you don’t want to generate cube files), then you can remove the ’Cube’ keyword. Not generating the .wave file should help speed-up the code since writing a rather large file is avoided. Running the cubegen code will be discussed in Section 5.3.

3.8 Density of States Keyword

A DOS plot is very helpful for identifying where the energetics of the adsorbate excited states are poised compared to the TiO_2 conduction band. DOS information is included in the output file when the following is present in your input file

```
Average Properties

Projected DOS
1
atoms 1 1,2 1,3 1,4 1,5 1,6 1,7 1,8 1,9 1,10 1,11 1,12 1
```

as can be found in the example input file one lines $163-167$. The first line tells the code to calculate certain average properties i.e. the density of states. When average properties are requested, you need to include the ’Kpoints’ keywork discussed in the next section. If you want to perform a projected DOS (pDOS) calculation, you need to include the appropriate keyword found on the third line above (line 165 of the example input file). The ’projected DOS’ keyword is followed by the number of projected DOS’s, in the example only one is requested. The fifth line (line 167 of the example input file) indicates which atoms will be used in the projected DOS. If more pDOS are desired, it would come in the line immediately following the fifth line above in the same format as it. The line must start with the word ’atoms’ followed by a list of comma separated numbers. There are two numbers between each comma with the first indicating the atom number and the second indicating the weight due to that atom. You should typically use a weight of 1.0 corresponding to that found in the example input file. Such a set of numbers can easily be generated in the bash shell this set of commands:

```
start=1
startPlus1=`expr $start + 1`
end=12
weight=1
echo -n "$start" ; for i in `seq $startPlus1 $end` ; do echo -n " $weight,$i" ; \
done ;  echo " $weight"
```

where the required input from the user is the *start* is the first atom in the series of atoms in the list, *end* is the last atom in the series of atoms in the list, and the *weight* is the weight described above. Be sure to modify this as needed for your system. Also, note that there is a limit to how many characters can be on one line i.e. 2046. You may easily reach this limit if your atom numbers go from 1 to 500, for example. Therefore, to circumvent this problem, add a backslash before this limit is reached as follows (... indicate continuing numbers not shown):

```
...,452 1,453 1,454 1,455 1,456 1,457 1,458 1,459 1 \
460 1,461 1,462 1,463 1,464 1,465 1,466 1,467 1,468 1,...
```

The DOS results available in the output file can be extracted in a form ready to be plotted using the fitdos code. The code will be discussed further in the Output Files and View Results section.

3.9 Kpoints Keyword

The ’Kpoints’ keyword is required for average properties calculations and IET simulations of periodic systems. The kpoints are points in the reciprocal lattice used to calculate average properties of the system. In this tutorial, we will restrict ourselves to a single kpoint at the origin with a weight of 1.0. The way to specify a single kpoint at the origin with a weight of 1.0 is as follows

```
kpoints
1
0.000000  0.000000  0.000000 1
```

which can be found in the input file. A larger number of kpoints gives better results for DOS calculations and other properties but the generation of the kpoints is beyond the scope of this tutorial. The 'Kpoints' keyword should always be given in the shown format and the format will not be discussed in more detail. Those interested in using more kpoints should consult the Yaehmop user manual, located at 'dynamics-0.4.2/docs/bind_manual.pdf'.

3.10 Absorbing Boundary Keyword

To model the surface of TiO_2, we have used a small periodic slab in the example input file. Due to the finite size of the slab, the photoexcited electron will artificially return to the adsorbate. The artificial recurrences can be reduced by making the slab larger. Unfortunately, the calculation becomes too slow when more atoms are included in the molecular structure. A way to circumvent this problem is to include 'absorbing potentials' that absorb (i.e., damp) the electron density when the scattering electron populates the orbitals of the atoms at the base of the TiO_2 slab. Such an absorbing potential mimics the effect of the rest of the TiO_2 surface, absorbing the electronic density out of the simulations box. The 'Absorbing' keyword requires a format specified in lines $175 - 184$ of the input file: Looking at those lines

```
Absorbing
72 1.000
118, 119, 120, 121, 122, 123, 124, 125, 126, 136   \
137, 138, 139, 140, 141, 142, 143, 144, 154, 155   \
156, 157, 158, 159, 160, 161, 162, 172, 173, 174   \
175, 176, 177, 178, 179, 180, 253, 254, 255, 256   \
257, 258, 259, 260, 261, 262, 263, 264, 265, 266   \
267, 268, 269, 270, 271, 272, 273, 274, 275, 276   \
277, 278, 279, 280, 281, 282, 283, 284, 285, 286   \
287, 288
```

we see there are many numbers. The first line after the 'Absorbing' keyword has two numbers. The first number is the number of orbitals with the absorbing potential and the second number is the value of the absorbing potential. The absorbing potential creates an exponential damping of the electron population. The larger the damping factor, the faster the electron population will be damped in the orbital. The timestep is included in the exponential damping so a smaller timestep with the 'Dynamics' keyword will result in quicker damping. The next eight lines list the orbital numbers where the damping will be applied. You should have the same number of orbitals as stated on the previous line. In this case there are 72 orbitals.
Remember that a Ti atom has 9 valence orbitals that are available to it, regardless of whether they are occupied or not, (it does have a valence configuration of $4s^2 3d^2$, but here just the fact that it has 1 s, 5 d, and 3 p orbitals is relevant. Note $1 + 5 + 3 = 9$), so 72 orbitals located on Ti atoms correspond to 8 atoms (since $8 * 9 = 72$). Notice in the list above, 118-126 (first Ti) has 9 members in it, so does 136-144 (second Ti), and so on. The eight Ti atoms are atom numbers 22, 24, 26, 28, 37, 38, 39 and 40 in the molecular structure. You can check that these Ti atoms correspond to those in the bottom of the slab, by opening the .com file available in the examples directory with GaussView. The orbital numbers need to be separated by commas. The backslashes indicate that more orbital numbers will follow on the next line.

To ease the construction of the 'Absorbing' keyword input, a python script is available in the dynamics directory. The python script requires as input the .com file and a list of atoms where you want absorbing potentials. From the examples directory, the command to generate the input found in the example input file is

```
$ cd ~/dynamics-0.4.2/examples
$ ../dynamics2/prep_absorbing.py catechol_anatase101.com 22 24 26 28 37 38 39 40
```

and you can copy and paste the output into the input file that you are generating.

3.11 More Information

More information about the Yaehmop input file can be found in the user manual located in the docs directory. You should be able to open the manual using the open command which will use your default pdf program.

```
$ cd ~/dynamics-0.4.2/docs
$ open bind_manual.pdf
```

The user manual is for the unmodified version of Yaehmop. Therefore, no information regarding the simulation of IET is enclosed.

4 Running the Example

In this section we will discuss how to run the code so we can start generating results. You should change directory back to your home directory and then create a new directory called 'tutorial'

```
$ cd ~
$ mkdir tutorial
$ cd tutorial
```

From here, you should copy the bind input file available in the examples directory

```
$ cp ~/dynamics-0.4.2/examples/catechol_anatase101.bind .
```

and the period is important at the end of the command because it tells the copy command to copy the file into the current working directory.

If you have installed this program on a cluster, you will want to launch the executable through the use of a submission script. One is provided for you in the dynamics-0.4.2/examples folder. Copy it into your folder

```
$ cp ~/dynamics-0.4.2/examples/submitIET_catechol_anatase101.sh .
```

It is appropriate for the $Grace$ cluster at Yale's HPC with the LSF system. It may be easily edited for another LSF cluster or for another system such as TORQUE or SLURM. On the LSF system, one would submit the script with the following command (after verifying the paths of the files inside).

```
$ bsub < submitIET_catechol_anatase101.sh
```

More documentation is given in the script itself as a comment. The output normally printed to the screen will be given as a numbered (by the jobid) .out file at the end of the calculation.

Otherwise, if you have installed this program locally, proceed through the following instructions.

Now, launch the executable with the input file

```
$ ~/dynamics-0.4.2/dynamics2/dynamics catechol_anatase101.bind
```

and output should be written to the terminal.

Either way, the code will finish with

```
Done!
```

indicating that the execution ended properly. The code should run for about a minute before it finishes. Finally, it is time to analyze the results.

5 Output Files and Visualizing Results

After running Yaehmop with the input file, catechol_anatase101.bind supplied in the examples directory, you will notice that a couple new files are present in your working directory. In the following, we will discuss how you can use these output files to visualize results.

5.1 Generating a Density of States Plot

To generate a DOS plot we will use the .out file produced by Yaehmop and the fitdos tool. To run the fitdos tool use the command

```
$ ~/dynamics-0.4.2/dynamics2/utils/fitdos catechol_anatase101.bind
```

from the tutorial directory. The tool will ask you to enter: E min, E max, broadening and Energy Step. The values I suggest you use are

```
Enter E min: -18.0
Enter E max: -6.0
Enter broadening: 50
Enter Energy Step: .01
```

The min and max set the energy range for the DOS. The energy step gives the grid size for the plot of the DOS. The broadening specifies the width of gaussians used to convolute the plot. The convolution is used to produce a smooth DOS plot.

The fitdos tool will produce the file catechol_anatase101.bind.DOS. The number of columns in the .DOS file depends on the number of projected DOS's requested. In the example, 1 projected DOS was requested so there are 3 columns. The first column is the DOS, the second column is the projected DOS, and the final column is the energy range. If you were to request 2 pDOS, then you would have 5 columns where the first column is the DOS, the second column is the first requested projected DOS, the third column is the second requested projected DOS, and the fourth column is the energy range. You should plot the DOS using your preferred plotting package.

There is some extra stuff past the line that says "END OF DOS" in this file. You may delete this line and the ones following it as shown below so that it does not appear as a stray line in your DOS. The version of catechol_anatase101.bind.DOS before the change would be saved as catechol_anatase101.bind.DOS.bk.

```
 $ sed -i.bk '/END OF DOS/,$d' catechol_anatase101.bind.DOS
```

We show how to make the plot using gnuplot below. The following commands will generate Figure 2. Note the difference between the letter l (in lt as in linetype) and the number 1 (as in one of the linetypes).

```
 $ gnuplot
 gnuplot> set xrange [-18:-6]
 gnuplot> set xlabel "ENERGY (eV)"
 gnuplot> set ylabel "DOS"
 gnuplot> set format y ""
 gnuplot> set style line 1 lt 1 lw 0.8
 gnuplot> set style line 2 lt 1 lw 2
 gnuplot> plot "catechol_anatase101.bind.DOS" u 3:1 w l ls 1 t "DOS"
 gnuplot> replot "catechol_anatase101.bind.DOS" u 3:2 w l ls 2 t "pDOS"
```

When you are ready to leave gnuplot, type the command exit

```
 gnuplot> exit
 $
```

You can compare your results to Figure 2 and with the corresponding Figure 6 in Reference [1]. One might note that the Fermi level of TiO_2 is off from its expected value. It is not a serious concern in this case, but there is a way to correct this, as described in Section 7.

You may also put these commands into a text file (*e.g.* plotIET.gpt) as so:

```
  set xrange [-18:-6]
  set xlabel "ENERGY (eV)"
  set ylabel "DOS"
  set format y ""
  set style line 1 lt 1 lw 0.8
  set style line 2 lt 1 lw 2
  plot "catechol_anatase101.bind.DOS" u 3:1 w l ls 1 t "DOS"
  replot "catechol_anatase101.bind.DOS" u 3:2 w l ls 2 t "pDOS"
```

and then submit it as so

```
gnuplot -persist < plotIET.gpt
```

If you do this on a cluster, find and load the relevant gnuplot module as so beforehand (replace the phrase "locationofmodule").

```
module load locationofmodule
```

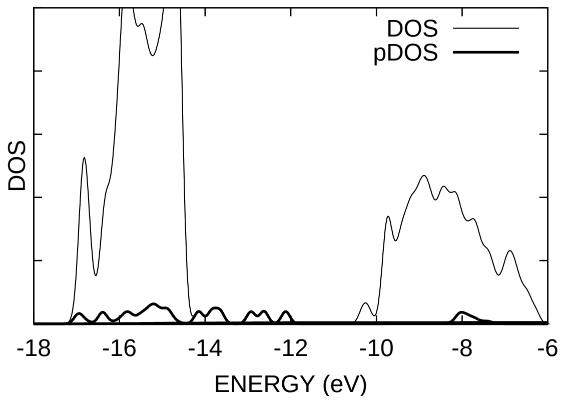

Figure 2: Density of states of the catechol on TiO_2 nanostructure. The thick curve at the bottom of the bottom of the graph represents the projected density of states on the catechol basis functions. The thin curve is the total density of states for the full nanostructure.

You can also include colors by replacing the previous style commands with these below.

```
set style line 1 lt 1 lc rgb "black" lw 5
set style line 2 lt 1 lc rgb "red" lw 5
```

You can also manipulate the data mathematically prior to graphing. For example to make the pDOS twice as intense use:

```
  replot "catechol_anatase101.bind.DOS" u 3:($2*2) w l ls 2 t "pDOS"
```

5.2 Visualizing the IET Curves

The results from the IET simulation are output into a .bind.edyn file. For the current example, the filename is catechol_anatase101.bind.edyn. In the file there are four columns. The first column is the simulation time. The second column is the survival probability. The survival probability gives the density of the photoexcited electron remaining on the adsorbate. The third column is the norm of the wavefunction for the photoexcited electron. When absorbing potentials are not in use, the norm should remain at 1.0. However, when absorbing potentials are in play, the norm of the wavefunction will decrease. The fourth and final column is the occupation (or density) of the photoexcited electron on the atoms you selected with the 'Occupation' keyword (Section 3.6).

Using the .bind.edyn file produced with the example input file for catechol on TiO_2, we can plot the survival probability as a function of time using gnuplot with the following commands

```
 $ gnuplot
 gnuplot> set xrange [0:100]
 gnuplot> set yrange [0:1]
 gnuplot> set xlabel "TIME (fs)"
 gnuplot> set ylabel "{/Symbol r}_{MOL} (t)"
 gnuplot> set style line 1 lt 1 lw 0.8
 gnuplot> set style line 2 lt 1 lw 0.8
 gnuplot> plot "catechol_anatase101.bind.edyn" u 1:2 w l ls 2 t "Injection Curve for Dye"
 gnuplot> replot exp(-x/6) w l ls 1 t "{/Symbol t} = 6 fs"
```

The resulting output should be the same as Figure 3. Remember, you can leave the gnuplot environment using the command "exit". You can compare Figure 3 with the corresponding Figure 8 in Reference [1].

5.3 Visualizing the Electron Density

To view the density of the photoexcited electron, we need to generate cube files and open the resulting files with either VMD or GaussView. The code we compiled earlier called cubebuilder2 is used to construct cube

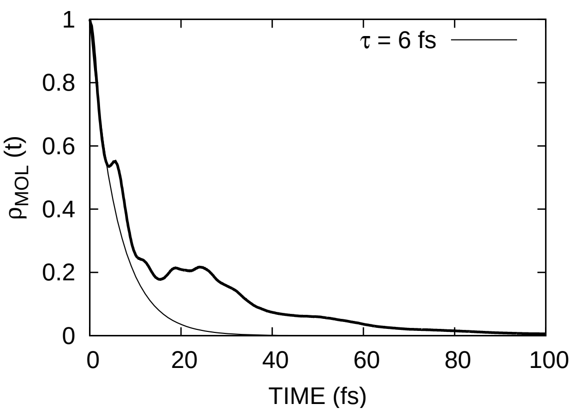

Figure 3: Time-dependent survival probability in a catechol sensitized TiO_2 nanostructure. The electronic excitation is initialized by populating the catechol-LUMO. The thin line is an exponential fitting to the elementary step.

files from the .wave output file. The 'Cube' keyword needs to have been included in the input file otherwise the .wave file will not been generated. The cubebuilder2 needs its own input file. An example input file is available under the name dynamics-0.4.2/examples/catechol.in.

Taking a look at the file dynamics-0.4.2/examples/catechol.in, we see:

```
region
0.25
0.0000  10.22784
0.0000  15.12800
0.0000  22.19775


make
1 11
11
1 0
0-10

name
cat_tio2
```

- The 'region' keyword specifies the size of the box.
- The electron density is output on a regular grid.
- The first number after the keyword is the distance between grid points.
- The second line is the minimum and maximum range on the x axis (a rule of thumb might be to use ± 3 Å of the x-axis range in the .com file).
- The third line is the minimum and maximum for the y axis and the fourth line is for the z axis as for the second line.
- The 'make' keyword determines how many cube files ("cubes") are generated.
- The first number after this keyword is the number of nonconsecutive time periods to consider (1 section here).
- The second number after this keyword is the total number of cubes to be generated for the whole calculation (11 cubes here).

- The next line gives the number of cubes to be generated in the first section (11 cubes here).
- The next line is the time step in units of 0.1 fs and offset (here they are 0.1 fs and 0 respectively so time starts at t=0 and each cube represents a 0.1 fs step into the future; see next point). This is a significant change from a previous version of this code.
- The next line indicates "step numbers" in that the cubes for $t = 0$ fs, $t = 0.1$ fs, $t = 0.2$ fs . . . $t = 1$ fs are desired.

The next line gives the step number in the IET evolution with time zero, $t = 0$, corresponding to step 1. In this example we will make cube files of the first 0.9 fs. The 'name' keyword gives the "basename" for the cubefiles. The filename will be constructed using the basename plus the step number and finished with the .cube file extension, cat_tio2_0000.cube, cat_tio2_0001.cube, cat_tio2_0002.cube . . . cat_tio2_0010.cube for example.

If one wanted to do two nonconsecutive explorations of electron density evolution over time, but say explore not only the first 1 fs in steps of 0.1 fs, but also the next 9 fs (up to $t = 10$ fs in steps of 1 fs) , one would do it as below. The numbers, in consecutive order, under "make" ask the code to make *2* sections of cubes with *20* cubes overall. The first section makes *11* cubes in time steps of 0.1 fs with no offset until 1 fs. The second section makes *9* cubes in time steps of 1 fs ($10 * 0.1$ fs) with no offset until 10 fs. The cubes generated will include the ones mentioned above as well as cat_tio2_0020.cube, cat_tio2_0030.cube, cat_tio2_0040.cube . . . cat_tio2_0100.cube

```
region
0.25
0.0000  10.22784
0.0000  15.12800
0.0000  22.19775


make
2 20
11
1 0
0-10
9
10 0
2-10

name
cat_tio2
```

Using these kinds of cubebuilder2 input files, the code can be run to generate the cube files. The .wave file produced from running Yaehmop with the example input file is called catechol_anatase101.bind.edyn.wave. Assuming you are in the Tutorial directory on your local machine, run the commands

```
$ cp ~/dynamics-0.4.2/examples/catechol.in .
$ ~/dynamics-0.4.2/cubebuilder/cubebuilder2 catechol.in catechol_anatase101.bind.edyn.wave
```

Otherwise, if you are running this program on a cluster, another submission script has been provided. Follow these commands (submit the script only after verifying the paths therein) if you are running on $Grace$. Otherwise, edit the script to your needs. See Section 4 for more instructions on clusters.

```
$ cp ~/dynamics-0.4.2/examples/submitCube_catechol_anatase101.sh
$ bsub < submitCube_catechol_anatase101.sh
```

Either way, enerating cube files can take a significant amount of time, especially if you are producing many of them. You will need to be patient. For a large molecule, it may take an hour per cube. The first cubefile generated from the example catechol on TiO_2 system, cat_tio2_0000.cube, is the catechol-HOMO initial state used in the IET simulation. Figure 4 shows the photoexcited electron density. You can compare the electron density with Figure 11 in Reference [1].

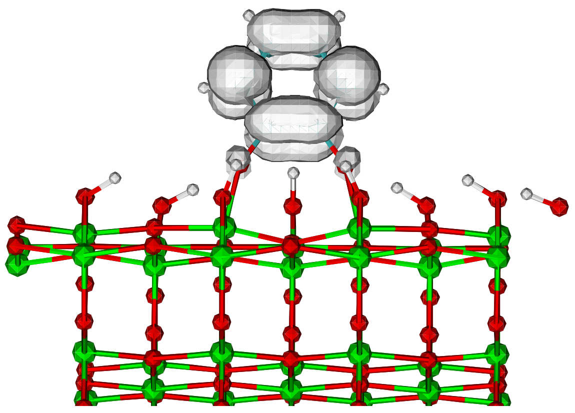

Figure 4: Charge distribution after populating the catechol LUMO.

The cube files can be opened with either VMD or GaussView. Here we will walk through the procedure for visualizing the cube file with GaussView. Open the cube file in GaussView and follow these steps:

1. Click ’Results’
2. Select ’Surfaces’
3. In the ’Surfaces and Cubes’ window Select the ’Surface Actions’ button
4. Select ’New Surface’

The surface should appear in the molecule window.

To visualize the cube file in VMD using the terminal, consider whether you have configured vmd to work through the terminal or not. If not, for a Mac, install the following alias (edited as appropriate for your system) into the .bash_profile and sourcing.

```
alias vmd='/Applications/VMD/VMD\ 1.9.1.app/Contents/vmd/vmd_MACOSXX86'
```

Then open the cube file using the command

```
 $ vmd cat_tio2_0001.cube
```

Otherwise, load the molecule by following these steps:

1. Click ’File’
2. Click ’New Molecule’
3. Select the ’Gaussian Cube’ type under ’Determine file type’
4. Click ’Browse’ and find your molecule and double click on it.
5. Find your molecule’s file and double click.
6. Click the ’Load’ button.

Once the molecule has been loaded, follow these steps:

1. Click ’Graphics’
2. Select ’Representations’
3. In the ’Graphical Representations’ window, Click ’Create Rep’
4. In the ’Graphical Representations’ window, Change ’Drawing Method’ from ’Lines’ to ’Isosurface’

5. Change 'Draw' from 'Points' to 'Solid Surface'
6. Change 'Isovalue' from 0 to 0.002 by dragging the bar next to the value or by typing it in.

The proper way to quit VMD is to follow these steps:

1. Click 'File'
2. Click 'Quit'
3. Click 'Yes'

These cube files may also be concatenated together to make a short movie with VMD. See Section 6.

6 Making Injection Movies using VMD

Follow these instructions in the folder where you generated the cube files described in Section 5.3. This set of instructions refers to the 'dynamics-0.4.2/makemovies/' folder. The movies (made from individual image files of the .tga format) are made per nonconsecutive section as described there. It is suggested to move the cube files for the first fs into its own folder and then the next set of cube files into another and execute this section for each separately.

These instructions copy over three files. First, mystate.vmd is a VMD script that helps one load the first file. Second, loadmols_clean.tcl is based on a VMD tcl script called "animatepdbs" written by John Stone of UIUC, which would "load a consecutively numbered sequence of PDB files into VMD for animation purposes" in the preferred orientation. It effectively creates the frames for the movie. Third, postproc_clean.sh is a bash script that prepares the files to be concatenated into a movie by renumbering and labeling them with the timestamp. The command also creates a file called mycube.cube out of the first cube in the series for mystate.vmd to use.

```
$ cp ~/dynamics-0.4.2/makemovie/* .
$ cp cat_tio2_0000.cube mycube.cube
$ vi loadmols_clean.tcl
```

At this point, we have to make sure the parameter definitions in loadmols_clean.tcl are correct, hence the step to open it in vi.

Below is the explanation for count paraphrased from the comments on top of the file.

- count - This should be equal to 0.
- count2 - This is the first index number for the output .tga number, matching the same index used to number the cubes. For example, for the cubes from 0 to 1 fs in steps of 0.1 fs, one would type 0.
- start - This the "frame number" of first cube file in the sequence, which is also set to 0 in the example above.
- end - This the "frame number" of last cube file in the sequence, which is set to 10 in the example above.
- step - This the increment of the frame numbers, which is set to 1 here, for 0.1 fs.
- fileformat - This is the Tcl format string which describes the filename/numbering, which here is

  ```
  cat_tio2_%04d.cube
  ```

  where the part between the tio2_ and the .cube is replaced by the program with the proper numbering.

For the second set of cubes from 1 fs to 10 fs, the following definitions would be appropriate.

```
set count 0
set count2 11
set start 20
set end 100
set step 10
set fileformat "cat_tio2_%04d.cube"
```

At the end of this tcl file, there is a section called ”Rotate an axis according to convenience: what viewpoint do you want to have?”. There are three main variables to change here are given below. If you do not want to change it, put a # sign in front of the command, which turns the line into a mere comment. Each line indicates a different way to rotate the frame about the axes.

- rotate x will rotate around the red axis in VMD.
- rotate y will rotate around the green axis in VMD.
- rotate z will rotate around the blue axis in VMD.

So for example, the following section will only rotate the frame around the blue axis axis by 270° as well as around the green axis by 90°.

```
#rotate x by 90
rotate y by 90
rotate z by 270
```

The next steps are are as follows. If you set up vmd to work in the terminal, do

```
$ vmd -e mystate.vmd
```

Otherwise, do the following after opening VMD.

- Click ’Extensions’
- Click ’Tk console’
- Type

  ```
  source loadmols_clean.tcl
  ```

- Press the Enter key.

Be sure to quit VMD so that the next few steps can be followed (see the end of Section 5.3 for detailed instructions).

Now the bash script postproc_clean.sh file needs to be edited. Note the following section at the top of the file.

```
count=0
step=1/10
start=0
end=10
```

These variable definitions seem to agree with the ones for loadmols_clean.tcl, but the variables are defined differently.

- count - will end up being the starting time / step. If the starting time is $t = 0$ fs and the step size is 0.1 fs, then $count = 0/0.1 = 0$. If the starting time is $t = 2$ fs and the step size is 1 fs, then $count = 2/1 = 2$.
- step - the step is always in units of 1 fs, so to get 0.1 fs, one needs to indicate it as the fraction ’1/10’. If the step size really is 1 fs, then $step = 1$.
- start - the start of single digit numbering for the files. As computer scientists do, we start counting from 0. If we are considering the second set, it starts from $start = 1 + 10 = 11$.
- end - the last of single digit numbering for the files, ending in 10 for the first set and 19 for the second set.

Therefore, for the next set of cube files, the variables would be defined this way:

```
count=2
step=1
start=11
end=19
```

Now in the terminal window, execute the following commands (the first command makes the script executable for the second and third commands). The second command checks for syntax errors (but not logical errors) in the bash script. If it finds none, it returns no output. The third command executes the script while giving every step explicitly to the screen, as prefaced by the "+" sign. This helps in making sure the script is working correctly as well as not making it look like the terminal has crashed. The fourth command creates the .gif file that will store the animation. The number of frames per second (fps) in this case is related to the first argument after delay, that is, 20:

$frame\ rate = (100/20)\ fps = 5\ fps$

The second argument after the delay refers to the images files to be concatenated. In our case they all start with the digit 0 therefore we use 0∗. The '-loop 0' part will make the animation loop until it is closed. Finally, 'playme.gif' will be the name assigned to this gif file. The fifth command will play the animation on your screen. If your terminal does not recognize the command 'convert' in the terminal, then skip these commands for now and see the instructions further below for installing this command.

```
 $ chmod 744 postproc_clean.sh
 $ sh -n ./postproc_clean.sh
 $ sh -x ./postproc_clean.sh
 $ convert -delay 20 0* -loop 0 playme.gif
 $ animate playme.gif
```

On the Mac, at least, the last four commands require the installation of ImageMagick under the Macports suite (on Linux this would require the use of the sudo apt-get command). Follow their instructions to install. To get the program working in the command line, the following line must be added to the .bash_profile as so:

```
export PATH=$PATH:/opt/local/bin
```

after which point you would 'source' the file:

```
$ source ~/.bash_profile
```

To install the program do,

```
$ sudo port install ImageMagick
```

At the end of this procedure, a file will be generated named 'playme.gif' which will be an image file that is continuously loop through the movie with a label in the upper-lefthand corner indicating the timestamp for the frame.

7 Alignment of Fermi Levels

There is a problem with Extended Hückel in terms of the alignment of Fermi levels due to the varying ways its empirical parameters define the vacuum energy level for different elements. An approximation of the Fermi level is to take the average between the edges of the conduction band (or HOMO) and valence band energies (or LUMO). Using this approximation, we find that the Fermi levels for catechol and anatase (101) TiO_2 are around -4.5 eV [5] and around 2.0 eV [6] respectively.

Looking at Figure 2, it is clear that the Fermi levels of these two parts deviate from these values. The band gaps are fine, since both the valence and conduction band edges are systematically off from their true values by the same amount. The results in this tutorial are also fine, since the pDOS from the catechol are well within the conduction band of the sensitized TiO_2. Correcting for the different relative positioning of the Fermi levels will only bring this small bump to another maximum in the total DOS. However, there may be other situations in which it is required to correct the Fermi levels (which will also move everything else around it by the same amount). The instructions for doing so are given in a separate tutorial, found here.

If the band gap itself is not satisfactory, then it is a separate issue, probably due to the chosen parameters. If you open /dynamics-0.4.2/dynamics2/eht_parms.dat in a text editor. You will notice several columns of letters and numbers, some beginning with the ; and some not. Below are the first 4 lines of this file. Every line starting with a ; is a comment or an inactive parameter. The first column, indicated by $Atomlab$, is the element name. $AtNo$ is the atomic number. $Nvalen$ is the number of valence electrons. The other columns the EHT parameters.

```
;EHParameters
;numberOfBasisFunctions: 300
;Atomlab AtNo Nvalen Nzeta Nquant Ang  IP       exp1     exp2    coeff1    coeff2
H         1      1      1     1     s  -13.600   1.3000   0.0000   1.0000   0.0000
```

If one is not satisfied with the parameters, they can switch to another set. For example, for carbon, the section reads as follows:

```
; Hoffmann
 C        6      4      1     2     s  -21.400   1.6250   0.0000   1.0000   0.0000
 C        6      4      1     2     p  -11.400   1.6250   0.0000   1.0000   0.0000
; Muller
;C        6      4      1     2     s  -19.376   1.5760   0.0000   1.0000   0.0000
;C        6      4      1     2     p  -11.072   1.4350   0.0000   1.0000   0.0000
; Cerda graphite
;C        6      4      1     2     s  -19.8892   2.0249   0.0000   0.76422   0.0000
;C        6      4      1     2     p  -13.08   1.62412   2.17687   0.27152   0.73886
;C        6      4      1     3     p  -2.04759   1.1944   0.0000   0.49066   0.0000
```

The Hoffmann parameters are active and the Muller and Cerda parameters are not. If one wanted to make the Hoffmann parameters inactive and the Muller active, one has to move the placement of the ; as follows:

```
; Hoffmann
 ;C        6      4      1     2     s  -21.400   1.6250   0.0000   1.0000   0.0000
 ;C        6      4      1     2     p  -11.400   1.6250   0.0000   1.0000   0.0000
; Muller
C         6      4      1     2     s  -19.376   1.5760   0.0000   1.0000   0.0000
C         6      4      1     2     p  -11.072   1.4350   0.0000   1.0000   0.0000
; Cerda graphite
;C        6      4      1     2     s  -19.8892   2.0249   0.0000   0.76422   0.0000
;C        6      4      1     2     p  -13.08   1.62412   2.17687   0.27152   0.73886
;C        6      4      1     3     p  -2.04759   1.1944   0.0000   0.49066   0.0000
```

Changing these parameters may improve the calculation.

8 Generating Your Own Structure

The .com file used to the create the example input file is available in the examples directory. You can open the file catechol_anatase101.com using GaussView and make modifications to the catechol adsorbate using the structure building features. You can also delete the catechol adsorbate and insert your favorite dye.

8.1 Preparing a Catechol TiO_2 Structure

To prepare a new catechol on TiO_2 structure we will first optimize the catechol adsorbate in vacuum. The resulting gas phase optimized structure will be adsorbed on the surface by hand using GaussView.

The catechol molecule can be constructed using GaussView. Open GaussView and follow these steps:

```
1. Click 'View' then 'Builder'
2. Click 'File' then 'New', a new window 'G1:M1:V1  New' pops out for building structures.
3. Choose the 'Ring Fragment' icon (the 1st one in the 2nd row) in the 'Builder' window
4. Click on the inside the window of  'G1:M1:V1 - New', and then a benzene ring is build.
5. Select 'O' atom (by clicking the 1st icon in the 1st row) and click on the H-atom of
   benzene ring in 'G1:M1:V1 - New' to make  a structure of catechol.
6. delete the two H atoms on the O atoms to make an anion (-2 charge) of catechol.
7. Click 'File' then 'Save', and save the structure as 'catechol.com'.
```

The file catechol.com should look like the following

```
%chk=Catechol.chk
# hf/3-21g geom=connectivity

Title Card Required

0 1
 C                  0.63636364    1.61157022    0.00000000
 C                  2.03152364    1.61157022    0.00000000
 C                  2.72906164    2.81932122    0.00000000
 C                  2.03140764    4.02783022   -0.00119900
 C                  0.63658264    4.02775222   -0.00167800
 C                 -0.06101836    2.81954622   -0.00068200
 H                  3.82874164    2.81940122    0.00063400
 H                  2.58160764    4.97997322   -0.00125800
 H                  0.08646064    4.98003322   -0.00263100
 H                 -1.16062236    2.81972922   -0.00086200
 O                 -0.07857634    0.37311938    0.00058521
 O                  2.74610797    0.37291518    0.00171004

 1 2 1.5 6 1.5 11 1.0
 2 3 1.5 12 1.0
 3 4 1.5 7 1.0
 4 5 1.5 8 1.0
 5 6 1.5 9 1.0
 6 10 1.0
 7
 8
 9
 10
 11
 12
```

Delete the first 6 lines of the file 'catechol.com' and replace with these 5 lines.

```
#p b3lyp/6-31g(d) opt=tight int=ultrafine

tight geometry optimization using an ultrafine integration grid: comment line

-2 1
```

You may do this using any editors of your choice, like 'emacs' or 'vi'. Submit 'catechol.com' for geometry optimization using Gaussian 09 using the 'Gaussian Calculation Setup' under 'Calculate' in the menu bar. An output file, 'catechol.log', is generated once the geometry optimization is complete.

Build the structure where the catechol molecule (anion) is attached to the TiO_2 anatase surface with a bi-dentate binding mode. Figure 5 shows an image of the nanostructure that you are building.

1. Open the structure file '/dynamics-0.4.2/examples/TiO2_base.com' for the TiO_2 anatase lattice.
2. Click the 'Custom Fragment' icon (the 2nd one in the 1st row of the 'Builder' window), and then click the square button in the 3rd row. A new window 'Custom Fragments' pops out. Move cursor to the first line

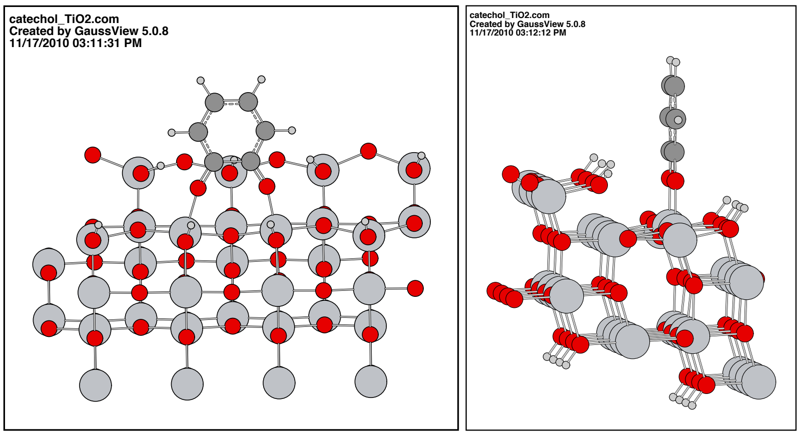

Figure 5: Nanostructure of TiO_2-anatase sensitized by catechol.

of the text part and click it. The first line is highlighted in blue, and then press the keyboard for 'Ctrl-N'. In this way, the active molecule (here, TiO_2) set to be a custom fragment 'Fragment A'. Click 'Ok' to close the window of 'Custom Fragments'.

3. Click 'File' then 'Open' to open the file 'catechol.log',

4. Click the 'Custom Fragment' icon (the 2nd one in the 1st row of the 'Builder' window), you will see that 'Fragment A' is displayed on the button in the 3rd row, and the structure of TiO_2 is visualized in the main window. If the structured shown in the main window is other existed custom fragment, you can click the button in the 3rd row to select the custom fragment of TiO2.

5. In the main window, click the atom ('O') on TiO_2 that you will use as the connecting point to the catechol molecule inside the window of 'G2:M1:V1 catechol.com', and connecting atom is then highlighted in light blue color.

6. Once you highlighted the connecting point (the 'O' atom) of the TiO_2 fragment, click on one of the O-atoms inside the window of 'G2:M1:V1 catechol.com'. The catechol molecule is now attached to the TiO_2 surface.

7. Adjust the catechol molecule to a proper position (by changing angles and dihedral angles) like the structure shown below, delete the extra 'O' atom on the TiO2 surface, and connect the other 'O' atom of catechol to the Ti atom of the TiO_2 surface. Save the properly adjusted geometry to a file named 'catechol_TiO2.com'.

A python script is available in the 'dynamics2' directory to facilitate the conversion of the .com file to the crystallographic format required in the Yaehmop input file. It requires that the lattice vectors be given at the end of the .com file. For example, in the examples directory, the last six lines of TiO2_base.com looks like the following:

```
 H   7.20288   3.81065 12.36338
 H   7.20220   7.54891 12.36539
 H   7.18674 11.32476 12.36605
Tv 10.22784   0.00000  0.00000
Tv   0.00000 15.12800  0.00000
Tv   0.00000  0.00000 22.19775
```

The first 3 lines are the final 3 atoms of the system. The last 3 lines are the x, y, and z lattice vectors in that order as indicated by "Tv". These vectors indicate the box which appears to enclose the system when opened in Gaussview. These lines should be set to the desired value as discussed in Section 3.3.

The python script is called 'com_to_crystbind.py' and when executed from the command line the name of the .com file should be included as an argument. For example

```
 $ ˜/dynamics-0.4.2/dynamics2/com_to_crystbind.py catechol_TiO2.com
```

will generate the file 'catechol_TiO2.bind' with the molecular structure in the required format for Yaehmop. The input for the 'Lattice' and 'Crystal Spec' keywords will be included in the generated file. The rest of the keywords needed for your desired calculation will need to be included using your favorite text-editor.

References

[1] Luis G.C. Rego and Victor S. Batista. *J. Am. Chem. Soc.*, 125:7989–7997, 2003.

[2] Sabas G. Abuabara, Luis G.C. Rego, and Victor S. Batista. *J. Am. Chem. Soc.*, 127:18234–18242, 2005.

[3] Sabas G. Abuabara, Clyde W. Cady, Jason B. Baxter, Charles A. Schmuttenmaer, Robert H. Crabtree, Gary W. Brudvig, and Victor S. Batista. *J. Phys. Chem. C*, 111:11982–11990, 2007.

[4] William R. McNamara, Robert C. Snoeberger III, Gonghu Li, James M. Schleicher, Clyde W. Cady, Macarena Poyatos, Charles A. Schmuttenmaer, Robert H. Crabtree, Gary W. Brudvig, and Victor S. Batista. *J. Am. Chem. Soc.*, 130:14329–14338, 2008.

[5] Noa Marom, Thomas Krzdrfer, Xinguo Ren, Alexandre Tkatchenko, and James R. Chelikowsky. *J. Phys. Chem. Lett.*, 5:2395–2401, 2014.

[6] Hongfei Li, Yuzheng Guo, and John Robertson. *J. Phys. Chem. C*, 119:18160–18166, 2015.