Frazier-PipeLine Documentation

Release 0.0.1

Henry Herbol

Oct 20, 2016

CONTENTS

Contents:

CHAPTER

ONE

FRAZIER PIPELINE

Pipeline for submitting solubility simulations. Bonus output may include Unsaturated Mayer Bond Order amongst other things.

1.1 Installing

Currently installation involves first installing clancelot:

cd ~; git clone git@github.com:clancylab/clancelot2.0.git

And then cloning this project:

cd ~; git clone git@github.com:hherbol/frazier-pipeline.git

Note, you'll also have to append the frazier-pipeline/pys folder to your PYTHONPATH variable.

echo 'nexport PYTHONPATH="/PATH/TO/FRAZIER/PIPELINE/pys:$PYTHONPATH"' >> ~/.zshrc

1.2 Documentation

Documentation is necessary, and the following steps MUST be followed during contribution of new code:

Setup

1. Download Sphinx. This can be done simply if you have pip installed via *pip install -U Sphinx*
2. Wherever you have *frazier-pipeline* installed, you want another folder called *frazier-pipeline-docs* (NOT as a subfolder of frazier-pipeline).

   cd ~; mkdir frazier-pipeline-docs; cd frazier-pipeline-docs; git clone -b gh-pages git@github.com:hherbol/frazier-pipeline.git html

3. Forever more just ignore that directory (don't delete it though)

Adding Documentation

Documentation is done using ReStructuredText format docstrings, the Sphinx python package, and indices with autodoc extensions. To add more documentation, first add the file to be included in *docs/source/conf.py* under *os.path.abspath('example/dir/to/script.py')*. Secondly, ensure that you have proper docstrings in the python file, and finally run *make full* to re-generate the documentation and commit it to your local branch, as well as the git *gh-pages* branch.

For anymore information on documentation, here is a link to the followed sphinx_tutorial.

1.3 Using FPL

Automated calculations for the Frazier Pipeline are as follows:

- *get_MBO()*
- *get_UMBO()*

The following is still prone to bugs. It should work for num_solvents=1; however, any more and it is prone to blowing up (ie, irrationally large enthalpy of solvations). For now, it has been hidden to avoid incorrect calculations.

- `get_enthalpy_solvation()`

For a quick start guide, in theory the following will suffice:

```
import fpl_auto

halides = [ "Cl", "Br", "I" ]
cations = ["MA","FA","Cs"]
solvents = ["acetone",
                        "ACN",
                        "gbl",
                        "DMF",
                        "dmso",
                        "nmp",
                        "methacrolein",
                        "nitromethane",
                        "odcb",
                        "THF",
                        "THTO"]

# Note, you can also do mixed halides by passing a list of halides
# such as h = ["Cl","Cl","I"].  In this example though, we just
# have h = "Cl" which is equivalent to h = ["Cl","Cl","Cl"]
h, c, s = halides[0], cations[0], solvents[0]

# Get the UMBO of the Oxygen-Carbon bond in a single Acetone molecule
# adsorbed onto a solute of PbCl3MA
umbo = fpl_auto.get_UMBO(h,c,s)
```

`fpl_auto.`**`get_MBO`**(*halide, cation, solvent, num_solvents=1, route_lvls=[0, 0, 0, 0], avg=True, criteria=[['O', 'C'], ['O', 'N'], ['O', 'S']]*)

Get the mayer bond order. The Mayer Bond Order (see *get_UMBO()*) for more details.

Parameters

halide: ***str*** The halide within the perovskite.

cation: ***str*** The cation within the perovskite.

solvent: ***str*** The solvent of the system.

num_solvents: ***int, optional*** The number of solvents to model explicitly (implicit is always on in background).

route_lvls: ***list, int, optional*** The level of theory to use.

avg: ***bool, optional*** Whether to average together all UMBO's that match the given criteria.

criteria: ***list, list, str, optional*** A list of lists, each list holding a list describing what bonds you want the UMBO for. By default, it is every bond with an oxygen atom involved.

Return

MBO: ***list, float, or float*** The Mayer Bond Order. If avg is False and there are more than one MBO matching criteria, a list is returned.

fpl_auto.**get_UMBO**(*halide, cation, solvent, offset=2.0, num_solvents=1, route_lvls=[0, 0, 0, 0], avg=True, criteria=[['O', 'C'], ['O', 'N'], ['O', 'S']]*)

Get the unsaturation (average?) mayer bond order. The Mayer Bond Order (MBO) is well described here. In short, it is a numerical representation of the probability of how many electrons partake in a bond. For instance, a single bond would have a theoretical bond order of 1.0; however, in practice it may have more or less depending on how electrons distribute across the molecule. The MBO helps describe this, and the Unsaturated MBO (UMBO) helps represent this in a more understandable fashion. That is, if the UMBO is larger than zero, the bond is weaker than theory. If the UMBO is less than zero, then the bond is stronger than theory.

Parameters

halide: ***str*** The halide within the perovskite.

cation: ***str*** The cation within the perovskite.

solvent: ***str*** The solvent of the system.

num_solvents: ***int, optional*** The number of solvents to model explicitly (implicit is always on in background).

route_lvls: ***list, int, optional*** The level of theory to use.

offset: ***float, optional*** The offset supplied to get the UMBO. In most cases we consider, this is 2.0 as that is the theoretical bond order of a double bonded oxygen to sulfur.

avg: ***bool, optional*** Whether to average together all UMBO's that match the given criteria.

criteria: ***list, list, str, optional*** A list of lists, each list holding a list describing what bonds you want the UMBO for. By default, it is every bond with an oxygen atom involved.

Return

UMBO: ***list, float, or float*** The Unsaturation Mayer Bond Order. If avg is False and there are more than one UMBO matching criteria, a list is returned.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

f

F

G