PAL Documentation

Release 0.0.1

Henry C. Herbol

Sep 05, 2017

CONTENTS

Contents:

CHAPTER

ONE

PHYSICAL ANALYTICS PIPELINE (PAL)

An automated interface to DFT and MD codes for specific properties.

If you wish to calculate the MBO of a given solvent to perovskite composition, for example the average O-H MBO in water to PbI3MA, you can do so with the following command:

```
MBO = pal.get_MBO("Cl", "Cs", "WATER",
                  ion="Pb",
                  route_lvl=0,
                  on_queue=False,
                  criteria=[["O", "H"]],
                  queue="batch", nprocs=2, xhost=None)
```

Similarly, for the UMBO you can simply do:

```
UMBO = pal.get_UMBO("Cl", "Cs", "WATER",
                    ion="Pb",
                    route_lvl=0,
                    on_queue=False,
                    criteria=[["O", "H"]],
                    offset=1.0,
                    queue="batch", nprocs=2, xhost=None)
```

Note the offset value here. This simply gives the Formal Bond Order (FBO).

Now, for binding energy calculations between a solvent and perovskite, the following works:

```
E = pal.get_enthalpy_solvation("Cl", "Cs", "WATER",
                               ion="Pb",
                               route_lvl=0,
                               unit="kT_300",
                               num_solvents=4,
                               R_cutoff=2000.0,
                               on_queue=False,
                               queue="batch", nprocs=2, xhost=None)
```

Note, this will calculate the binding energy of four water molecules!

If you wish to use a mixed solvent, the following can be done:

```
E = pal.get_enthalpy_solvation("Cl", "Cs", {"WATER": 0.75, "GBL": 0.25},
                               ion="Pb",
                               route_lvl=0,
                               unit="kT_300",
                               num_solvents=4,
                               R_cutoff=2000.0,
```

```
                                    on_queue=False,
                                    queue="batch", nprocs=2, xhost=None)
```

This specifies a 3:1 ratio of water to GBL. By setting the number of solvents to 4, we get exactly 3 water molecules and one GBL molecule.

Finally, if one wishes to calculate any of the above properties for a mixed halide system, they need only pass a list of halides instead. For instance:

```
E = pal.get_enthalpy_solvation(["Cl", "Cl", "Br"], "MA", "DMSO",
                                    ion="Pb",
                                    route_lvl=0,
                                    unit="kT_300",
                                    num_solvents=1,
                                    R_cutoff=2000.0,
                                    on_queue=False,
                                    queue="batch", nprocs=2, xhost=None)
```

1.1 H_solv

A function to return the enthalpy of solvation for a given Perovskite system.

H_solv.**get_enthalpy_solvation**(*halide*, *cation*, *solvent*, *ion='Pb'*, *route_lvl=1*, *unit='kT_300'*, *num_solvents=25*, *R_cutoff=2000.0*, *charge_and_multiplicity='0 1'*, *charge_and_multiplicity_solute='0 1'*, *charge_and_multiplicity_solvent='0 1'*, *on_queue=False*, *rerun=False*, *queue='batch'*, *nprocs=1*, *xhost=None*)

This automates the process of calculating the enthalpy of solvation for a solvent bonded to a given solute.

NOTE - DO NOT USE THIS METHOD FOR NOW!

Parameters

solute: ***str*** A solute for which to be solvated.

solvent: ***str* or *dict, str, float*** The solvent of the system. If using a mixed solvent, pass as a dictionary: {"ACETONE": 0.3, "GBL": 0.7}

on_queue: ***bool, optional*** Whether to run this simulation on the queue.

queue: ***str, optional*** Which queue to run the simulation on.

nprocs: ***int, optional*** How many processors to use.

xhost: ***list, str or str, optional*** A list of processors, or a single processor for which to submit the simulations (on the queue).

unit: ***str, optional*** What units the energy should be returned in.

num_solvents: ***int, optional*** The number of solvents to use for enthalpy of solvation calculation.

R_cutoff: ***float, optional*** The radial distance from the ion for which you want to select molecules for the calculation.

charge_and_multiplicity: ***str, optional*** The charge and multiplicity of the full system.

charge_and_multiplicity_solute: ***str, optional*** The charge and multiplicity of the solute.

charge_and_multiplicity_solvent: ***str, optional*** The charge and multiplicity of the solvent.

name_append: ***str, optional*** What to append to the name for these simulations.

route_lvls: ***list, int, optional*** What level of theory (DFT) to run these calculations at.

rerun: ***bool, optional*** By default if an Orca simulation exists, it is used. By setting rerun to True this will not be the case.

Return

H_solv: ***float*** The enthalpy of solvation.

1.2 MBO

A function to return the Mayer Bond Order of a given Perovskite system.

MBO.**get_MBO**(*halide, cation, solvent, ion='Pb', num_solvents=1, route_lvl=1, avg=True, criteria=[['O', 'C'], ['O', 'N'], ['O', 'S']], on_queue=False, queue='batch', nprocs=1, xhost=None, debug=False*)

Get the Mayer Bond Order (MBO) of all bonds that are satisfied in the given criteria. By default, avg=True and, thus, all the MBOs are averaged together. Setting this to False will instead return a list of MBOs that satisfy the given criteria.

Parameters

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

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

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

ion: ***str, optional*** The ion of the perovskite. By default this is Pb, but can also be Sn.

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

route_lvl: ***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.

on_queue[*bool, optional*] Whether to run this simulation on the queue.

queue[*str, optional*] Which queue to run the simulation on.

nprocs[*int, optional*] How many processors to use.

xhost[*list, str or str, optional*] A list of processors, or a single processor for which to submit the simulations (on the queue).

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.

1.3 UMBO

A function to return the Unsaturated Mayer Bond Order of a given Perovskite system.

UMBO.**calculate_umbo**(*self, solvent, criteria=[['O', 'C'], ['O', 'N'], ['O', 'S']], avg=True, offset=2.0, name=None*)

Calculate the UMBO of the system with the given criteria.

Parameters

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

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.

name: ***str, optional***The name of the system for which to calculate the UMBO for.

UMBO.**get_UMBO**(*halide, cation, solvent, ion='Pb', offset=2.0, num_solvents=1, route_lvl=1, avg=True, criteria=[['O', 'C'], ['O', 'N'], ['O', 'S']], on_queue=False, queue='batch', nprocs=1, xhost=None, debug=False*)

Get the unsaturation mayer bond order. The Mayer Bond Order (MBO) is well described 'here <http://pubs.rsc.org/en/Content/ArticleLandi

ng/2001/DT/b102094n#!divAbstract>'_. In short, it is a numericalrepresentation 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.

ion: ***str, optional***The ion of the perovskite. By default this is Pb, but can also be Sn.

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

route_lvl: ***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.

on_queue[*bool, optional*] Whether to run this simulation on the queue.

queue[*str, optional*] Which queue to run the simulation on.

nprocs[*int, optional*] How many processors to use.

xhost[*list, str or str, optional*] A list of processors, or a single processor for which to submit the simulations (on the queue).

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

h

m

u

C

G

H

M

U