Frazier-PipeLine Documentation

Release 0.0.1

Henry Herbol

Oct 17, 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, the tutorial follwed can be found here.

1.3 Using FPL

```
fpl_auto.get_UMBO()
```

Once installed, FPL can be used to automate some work. The following is an example use of how to (1) generate a system of a solute with solvent, (2) equilibrate in lammps, (3) equilibrate with less solvents in lammps, (4) equilibrate in DFT using Orca, and (5) calculate the enthalpy of solvation.

```
########################################
solute = "pb2+"
solvent = "THTO"
run_name = "%s_%s" % (solvent, solute)

# Generate initial object
fpl_obj = fpl.fpl_job(run_name, solvent, solute)
fpl_obj.cml_dir="/fs/home/hch54/frazier-pipeline/cml/"

# Set parameters
fpl_obj.num_solvents=1

# Generate system
fpl_obj.generate_system()

########################################
# Add the tasks here

## Task 1 - Large Lammps Simulation
### PARAMETERS
task1 = run_name + "_large_lammps"
fpl_obj.queue=None
fpl_obj.procs=1
fpl_obj.lmp_run_len=10000
fpl_obj.trj_file=None
### ADD TASK
task = lmp_large_job(fpl_obj, task1)
fpl_obj.add_task(task)

## Task 2 - Small Lammps Simulation
### PARAMETERS
task2 = run_name + "_small_lammps"
fpl_obj.queue=None
fpl_obj.procs=1
fpl_obj.lmp_run_len=10000
### ADD TASK
        # Note, you can always overwrite the callback function
        # tsk = lmp_small_job(fpl_obj, task2)
        # tsk.callback = None
        # fpl_obj.add_task( task2, tsk )
task = lmp_small_job(fpl_obj, task2)
fpl_obj.add_task(task)

## Task 3 - Orca Simulation
### PARAMETERS
task3 = run_name + "_orca"
fpl_obj.queue="batch"
fpl_obj.procs=4
fpl_obj.route = "! OPT B97-D3 SV GCP(DFT/TZ) ECP{def2-TZVP} Grid7 SlowConv LooseOpt"
### ADD TASK
```

```
task = orca_job(fpl_obj, task3)
fpl_obj.add_task(task)

########################################

# Run the simulation here
fpl_obj.start(save=False)

########################################

## Task 4 - Calculate Enthalpy of Solvation
### PARAMETERS
task4 = run_name + "_Hsolv"
fpl_obj.queue = "batch"
fpl_obj.procs = 4

fpl_obj.route = "! B97-D3 SV GCP(DFT/TZ) ECP{def2-TZVP} Grid7 SlowConv"
fpl_obj.extra_section = "%basis aux auto NewECP Pb \"def2-SD\" \"def2-TZVP\" end␣
↪NewECP Cs \"def2-SD\" \"def2-TZVP\" end NewGTO S \"def2-TZVP\" end end"
fpl_obj.charge_and_multiplicity = "0 1"

fpl_obj.route_solute = "! B97-D3 SV GCP(DFT/TZ) ECP{def2-TZVP} Grid7 SlowConv"
fpl_obj.extra_section_solute = "%basis aux auto NewECP Pb \"def2-SD\" \"def2-TZVP\"␣
↪end NewECP Cs \"def2-SD\" \"def2-TZVP\" end end"
fpl_obj.charge_and_multiplicity_solute = "0 1"

fpl_obj.route_solvent = "! B97-D3 SV GCP(DFT/TZ) ECP{def2-TZVP} Grid7 SlowConv"
fpl_obj.extra_section_solvent = "%basis aux auto NewGTO S \"def2-TZVP\" end end"
fpl_obj.charge_and_multiplicity_solvent = "0 1"

### ADD TASK
tasks = fpl_calc.enthalpy_solvation(fpl_obj, task4)
fpl_obj.add_task(tasks, parallel=True)

fpl_obj.start(save=False)

H_solv = fpl_calc.post_enthalpy_solvation(fpl_obj)
```

To make things easier, this whole process can be automated for varying solute, solvent combinations by simply using the fpl_auto class:

```
import fpl_auto

e_solv = fpl_auto.get_enthalpy_solvation("pb2+","THTO")
print e_solv
```

Or, if you want to submit it to the queue:

```
import fpl_auto

e_solv = fpl_auto.get_enthalpy_solvation("pb2+","THTO",on_queue=True)
e_solv.wait()
H = e_solv.enthalpy()
print H
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search