# Frazier-PipeLine Documentation

***Release 0.0.1***

**Henry Herbol**

**Sep 30, 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

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, and (4) equilibrate in DFT using Orca.

```
import os

import fpl
from fpl_lmp_large import job as lmp_large_job
from fpl_lmp_small import job as lmp_small_job
from fpl_orca import job as orca_job

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

# Generate initial object
fpl_obj = fpl.fpl_job(run_name, solvent, solute)

# Set parameters
fpl_obj.cml_dir="/fs/home/hch54/frazier-pipeline/cml/"
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 # Run locally
fpl_obj.procs=1 # On only 1 processor
fpl_obj.lmp_run_len=0 # For debugging, only run for 1 timestep
fpl_obj.trj_file=None # No trajectory file
### ADD TASK
fpl_obj.add_task(
        task1, lmp_large_job(fpl_obj, task1)
        )

## Task 2 - Small Lammps Simulation
### PARAMETERS
task2 = run_name + "_small_lammps"
fpl_obj.queue=None
fpl_obj.procs=1
fpl_obj.lmp_run_len=2000
### ADD TASK
fpl_obj.add_task(
        task2, lmp_small_job(fpl_obj, task2)
        )

## Task 3 - Orca Simulation
### PARAMETERS
task3 = run_name + "_orca"
fpl_obj.queue="batch" # Run on the queue
```

```
#fpl_obj.xhosts="whale" # Run on whale only
fpl_obj.procs=4
### ADD TASK
fpl_obj.add_task(
        task3, orca_job(fpl_obj, task3)
        )

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

# Run the simulation here
fpl_obj.start()

########################################
```

CHAPTER

TWO

# INDICES AND TABLES

- genindex
- modindex
- search