Welcome to asmd2

Rigoberto Hernandez
Gungor Ozer
Dale R. Merz Jr.
Hailey Bureau

April 23, 2013

1 Quickstart

1.1 Command Line

Command line:
gen.py(executable script) + engine(namd,amb) + molecule(da,ee)

```
./gen.py namd da
```

1.2 engine.gconf

Edit the gconf file for the engine selected as necessary. This is one possible configuration for performing ASMD on Decaalanine from 13.0 Åto 33.0 Åin 3 solvents and 10 stages, equally discretized at 10% of the total extension per stage.

```
[danvt]
mol = danvt
zcrd = 13.0
envdist = 01.vac:zcrd,02.imp:zcrd,03.exp:zcrd
dist = 20.0
ts   = 2.0
n    = 2.,3.
environ = 01.vac,02.imp,03.exp
gate = steele2
comp = cpu
cn   = 2
ppn_env = 01.vac:cn,02.imp:cn,03.exp:cn
```

```
wallt= mwt
wt_env  = 01.vac:wallt,02.imp:wallt,03.exp:wallt
queue= standby
q_env   = 01.vac:queue,02.imp:queue,03.exp:queue
lnspc   = True
path_seg= 0.1
path_svel= 1.0
freq = 50,50,50,50,50
howmany=100,7,7,2,1
langevD = 5
```

2 Description

2.1 Steered Molecular Dynamics

Steered Molecular Dynamics, pulling a peptide with an harmonic potential.

2.2 Adaptive Steered Molecular Dynamics

By performing steered molecular dynamics adaptively, i.e. stage by stage, one obtains the potential of mean force more quickly, with fewer trajectories.

3 Setup for NAMD

Downloaded a new molecule from the PDB(Protein Data Bank), generated a protein structure file with psfgen, run an equilibration in the desired force field? Now you're ready to configure that molecule for use in asmd2.

3.1 General Control Templates

Generally, only two of the following sections, mol.conf and struc, require added templates for the continued use of asmd2 to perform full-scale adaptive or simple steered molecular dynamics on new molecules. The rest of the templates, python scripts, and bash scripts are general enough to require no further adjustments. A short description is provided for each in case further development in the algorithm is required.

3.1.1 continue

The continue.py script packs the smdforces.out/tef.dat(time,extension,force) files into 1 pickle per stage. It also carries out the selection and copying of

the daOut.coor and daOut.vel files for use in the following stage using the Jarzynski averaging criterion.

Example:
asmd2/00.maindir/namd/continue/continue.py
Code:

```
def calc_pmf(data,st,w_c):
    phase = int(st)-1
    if phase == 0:
        data[::,::,3] = np.cumsum(data[::,::,3]*path_v_aps[phase]*dt,axis=1)
        data[::,::,3] = data[::,::,3] + w_c[phase]
        deltaf = np.log(np.exp(data[::,::,3]*beta).mean(axis=0))*(1/beta)
        d = np.linspace(spos,spos+domain[phase],deltaf.shape[0])
        JA = deltaf[-1]
        return JA
    else:
        data[::,::,3] = np.cumsum(data[::,::,3]*path_v_aps[phase]*dt,axis=1)
        data[::,::,3] = data[::,::,3] + w_c[phase]
        deltaf = np.log(np.exp(data[::,::,3]*beta).mean(axis=0))*(1/beta)
        d = np.linspace(spos+domain[phase-1],spos+domain[phase],deltaf.shape[0])
        JA = deltaf[-1]
        return JA

class asmd_calcs:
    def __init__(self,wrk_pkl,w_c,d_cp):
        self.wrk = wrk_pkl
        self.w_c = w_c
        self.d_cp = d_cp
    def create_pkl(self,st):
        acc = []
        self.wrk[st]={}
        self.d_cp[st]={}
        stdir = os.path.join(my_dir,st)
        folds=[f for f in os.listdir(stdir) if os.path.isdir(os.path.join( \
                                                    stdir,f))]
        foldp=[os.path.join(stdir,f) for f in folds]
        seeds=[(p.split('/')[-2],p.split('/')[-1].split('.')[2]) for f in foldp \
               for p in glob(os.path.join(f,'*tef.dat*'))]
        seeds=[p.split('/')[-1].split('.')[2] for f in foldp \
                 for p in glob(os.path.join(f,'*tef.dat*'))]
```

```
for path in glob(os.path.join(my_dir,'%s/*/*tef.dat*' % st)):
    folder = path.split('/')[-2]
    seed = path.split('/')[-1].split('.')[2]
    self.wrk[st][seed]={}
for path in glob(os.path.join(my_dir,'%s/*/*tef.dat*' % st)):
    folder = path.split('/')[-2]
    seed = path.split('/')[-1].split('.')[2]
    sample_i = np.loadtxt(path)
    # errcheck
    if len(sample_i)!= xxlenarrayxx:
        os.remove(path)
        del self.wrk[st][seed]
        seeds.remove(seed)
    else:
        acc.append(sample_i)
        data_1=np.array(sample_i)
        tew,wf=calc_work(data_1,st,self.w_c) #sample_i/data => tew
        self.wrk[st][seed]=folder,tew,wf
        os.remove(path)
data=np.array(acc)
JA=calc_pmf(data,st,self.w_c)               # get JA
wf_sd=dict([(self.wrk[st][s][2],s) for s in seeds])
sel_seed=wf_sd.get(JA, wf_sd[min(wf_sd.keys(), key=lambda k: abs(k-JA))])
work_ss=self.wrk[st][sel_seed][2]
self.d_cp[st][sel_seed]=self.wrk[st][sel_seed][0]
print wf_sd
print 'JA:',JA
print 'selected seed',sel_seed
print 'work for selected seed:',work_ss
self.w_c[int(st)]=work_ss
return seeds
```

3.1.2 go

The go.py script runs steered molecular dynamics any number of times.

Examples:
asmd2/00.maindir/namd/go/go-ggatecpu2.py
asmd2/00.maindir/namd/go/go-ggategpu2.py
asmd2/00.maindir/namd/go/go-fgatecpu2.py
Code:

```
def reg_ex(script,subout,subin,n):
    o=open(script,'r+')
    text=o.read()
    text=re.sub(subout,subin,text)
    o.close()
    o=open(script,'w+')
    o.write(text)
    o.close()

def run_namd(i):
    seed = randint(10000,99999)
    while seed in slist:
        seed = randint(10000,99999)
    slist.append(seed)
    cp_file(my_dir,'smd.namd',my_dir,'smd.namd.%s' % (seed))
    script = os.path.join(my_dir,'smd.namd.%s' % (seed))
    reg_ex(script,'xxxxx',str(seed),i)
    st = time.time()
    os.system('namd2 +pxxnodecountxx smd.namd.%s > run.log' % seed)
    tt = time.time()-st
    os.remove(script)
    os.rename('smdforces.out','%d-tef.dat.%s' % (i,seed))
    os.rename('daOut.coor','daOut.coor.%s' % (seed))
    os.rename('daOut.vel','daOut.vel.%s' % (seed))
    os.system('python ../hb.py %d %s' % (i,seed))
    return tt
```

3.1.3 hb

The hb.py script generates the pickle describing the bonding in a trajectory.

Example:
asmd2/00.maindir/namd/hb/hb.py
Code:

```
#!/usr/bin/env python
import MDAnalysis
import MDAnalysis.analysis.hbonds
import MDAnalysis.analysis.distances
from sys import argv
import numpy as np
import os,sys,pickle
```

```
#_____________universe_____________________________________________________
u = MDAnalysis.Universe('../../../../00.struc/xxenvironxx/00.psf','daOut.dcd',\
                        permissive=True)

def analyze_bond(univ,seg1,seg2):
    try:
        name1=seg1.replace(' ','')
        name2=seg2.replace(' ','')
        h = MDAnalysis.analysis.hbonds.HydrogenBondAnalysis(univ,seg1,seg2, \
                                                distance=4.0, angle=140.0)
        results = h.run()
        pickle.dump(h.timeseries,open('%s-hb_%s_%s.pkl.%s' % (sys.argv[1], \
                                            name1,name2,sys.argv[2]),'w'))
    except:
        pass

#__analyze__bonds___________________________________________________________
analyze_bond(u,'protein','protein')
analyze_bond(u,'protein','segid WT1')
```

3.1.4 hb_pkl

The hb_pkl directory contains the hb_pkl.py script. This script pickles all the hydrogen bonding trajectory pickles into 1 pickle for that stage.

Example:
asmd2/00.maindir/namd/hb_pkl/hb_pkl.py
Code:

```
    for path in glob(os.path.join(my_dir,'%s/*/*-hb_pr*pr*.pkl.*' % stage)):
        print path
        seed = path.split('.')[-1]
        sample_i = pickle.load(open(path,'rb'))
        if len(sample_i)==xxlenbpklxx:
            dct_s_hb[seed]=[sample_i]
            os.remove(path)
    if len(dct_s_hb)>0:
        pickle.dump(dct_s_hb,open('%s-sd_hb.pkl' % stage,'w'))
    if my_dir.split('/')[-2].split('.')[1]=='exp':
        for path in glob(os.path.join(my_dir,'%s/*/*-hb_pr*segid*.pkl.*' % stage)):
            print path
```

```
            seed = path.split('.')[-1]
            sample_i = pickle.load(open(path,'rb'))
            if len(sample_i)==100:
                dct_s_whb[seed]=[sample_i]
                os.remove(path)
        if len(dct_s_hb)>0:
            pickle.dump(dct_s_whb,open('%s-sd_wp.pkl' % stage,'w'))
```

3.1.5 job

The job directory contains the bash scripts submitted to the pbs resource manager for controlling the go.py scripts, which run steered molecular dynamics in any given stage.

Example:
asmd2/00.maindir/namd/job/job-ggatecpu2.py
asmd2/00.maindir/namd/job/job-ggategpu2.py
asmd2/00.maindir/namd/job/job-fgatecpu2.py
Code:

```
#!/bin/bash
#PBS -N xxjobnamexx
#PBS -j oe
#PBS -l xxwalltimexx
#PBS -l pmem=220mb
#PBS -l xxnodesxx
#PBS -V

# job properties
NAMD_DIR=/share/apps/NAME_2.9_Linux-x86-64-multicore/
export PATH=${NAMD_DIR}:${PATH}
export LD_LIBRARY_PATH=${NAMD_DIR}:${LD_LIBRARY_PATH}

cd $PBS_O_WORKDIR

# run job
./go.py
```

3.1.6 jobc

The jobc directory contains the bash scripts that are submitted to a pbs resource manager for job control of the continue.py scripts.

```
Example:
asmd2/00.maindir/namd/jobc/job-ggatecpu2.py
asmd2/00.maindir/namd/jobc/job-ggategpu2.py
asmd2/00.maindir/namd/jobc/job-fgatecpu2.py
Code:
#!/bin/bash
#PBS -N xxjobnamexx
#PBS -j oe
#PBS -l walltime=27:00
#PBS -l pmem=310mb
#PBS -l nodes=1:ppn=1
#PBS -V

# job properties
cd $PBS_O_WORKDIR

NUM=xxnumxx

# run job
./$NUM-continue.py $NUM
```

3.1.7 jobhb

The jobhb directory contains the bash scripts that are submitted to a pbs resource manager, specific to the cluster to be used, that controls the pickling of the hydrogen bonding pickles obtained per trajectory into 1 pickle associated with the stage in which they were obtained.

```
Example:
asmd2/00.maindir/namd/jobhb/job-ggatecpu2.py
asmd2/00.maindir/namd/jobhb/job-ggategpu2.py
asmd2/00.maindir/namd/jobhb/job-fgatecpu2.py
Code:
#!/bin/bash
#PBS -N xxjobnamexx
#PBS -j oe
#PBS -l walltime=27:00
#PBS -l pmem=310mb
#PBS -l nodes=1:ppn=1
#PBS -V
```

```
# job properties
cd $PBS_O_WORKDIR

NUM=xxnumxx

# run job
./$NUM-continue.py $NUM
```

3.1.8 mol.conf.tcl - Steering Control

The mol.conf directory is where solvent configuration files by molecule first and solvent second are stored.

The control file, where the velocity of the pseudoatom and force constant of the harmonic potential are set, is placed in the following location.

```
Examples:
asmd2/00.maindir/namd/mol.conf/da/01.vac/smd_force.tcl
asmd2/00.maindir/namd/mol.conf/da/02.imp/smd_force.tcl
asmd2/00.maindir/namd/mol.conf/da/03.exp/smd_force.tcl
Code:
set Tclfreq xxfreqxx
set t 0
#set currentStep yyyyy

# contraint points

set c1x 0.0
set c1y 0.0
set c1z 0.0

set c2x 0.0
set c2y 0.0
set c2z [expr xxzcoordxx+xxcur_zxx]

# force constant (kcal/mol/A^2)
set k 7.2

# pulling velocity (A/timestep)
set v xxvelocityxx
```

```
set outfilename smdforces.out
open $outfilename w
```

3.1.9 psfgen

This directory contains some example pgn scripts used for structure generation and solvation.

```
package require psfgen
psfcontext new delete
topology ../reso/toppar/top_all27_prot_lipid.rtf

# build protein segment
segment PEP {
  pdb eenoh.pdb
  first ACE
  last CT2
}

coordpdb eenoh.pdb PEP
guesscoord

# write psf & pdb
writepdb ee_nw.pdb
writepsf ee_nw.psf

# End of psfgen commands
```

3.1.10 struc

The struc directory houses the structure files by molecule first and solvent second.

```
Examples:
asmd2/00.maindir/namd/mol.conf/da/01.vac
asmd2/00.maindir/namd/mol.conf/ee/03.exp
asmd2/00.maindir/namd/mol.conf/danvt/02.imp
Code:
```

3.1.11 toppar

The toppar directory is for the most commonly used topology and parameter files.

```
Examples:
asmd2/00.maindir/namd/toppar/par_all27_prot_lipid.prm
asmd2/00.maindir/namd/toppar/top_all27_prot_lipid.inp

*>CHARMM22 All-Hydrogen Parameter File for Proteins and Lipids <<
*>>>>> Includes phi, psi cross term map (CMAP) correction <<<<<<<
*>>>>>>>>>>>>>>>>>>>>>   July, 2003   <<<<<<<<<<<<<<<<<<<<<<<<
* All comments to ADM jr. via the CHARMM web site: www.charmm.org
*               parameter set discussion forum
BONDS
!
!V(bond) = Kb(b - b0)**2
!
!Kb: kcal/mole/A**2
!b0: A
!
!atom type Kb          b0
!
!Carbon Dioxide
CST  OST   937.96      1.1600 ! JES
!Heme to Sulfate (PSUL) link
SS   FE    250.0       2.3200 !force constant a guess
           !equilbrium bond length optimized to reproduce
           !CSD survey values of
           !2.341pm0.01 (mean, standard error)
           !adm jr., 7/01
C    C     600.000     1.3350 ! ALLOW ARO HEM
                ! Heme vinyl substituent (KK, from propene (JCS))
```

3.1.12 toppar.all

The toppar.all directory is for the least commonly used but all known topology and parameter files.

3.2 Adding a new molecule

A few key template files must be put into place!

3.2.1 Starting Structure

The same starting coordinates are used for every steered molecular dynamics' trajectory.

Example: To study decaalanine, assigned a label "da", in three solvents, the following "equilibrated structure" files are required:

```
asmd2/00.maindir/namd/struc/da/01.vac/00.pdb
asmd2/00.maindir/namd/struc/da/01.vac/00.psf

asmd2/00.maindir/namd/struc/da/02.imp/00.pdb
asmd2/00.maindir/namd/struc/da/02.imp/00.psf

asmd2/00.maindir/namd/struc/da/03.exp/00.pdb
asmd2/00.maindir/namd/struc/da/03.exp/00.psf

CRYST1    0.000    0.000    0.000   0.00   -NaN-1542439930724038816667608397359150530
ATOM      1  N   ALA B   1       0.166   0.267  -0.304  1.00  0.00      BH
ATOM      2  HT2 ALA B   1      -0.544   0.183   0.437  1.00  0.00      BH
ATOM      3  HT3 ALA B   1       0.949   0.817   0.184  1.00  0.00      BH
ATOM      4  CA  ALA B   1       0.767  -1.116  -0.506  1.00  0.00      BH
ATOM      5  HA  ALA B   1      -0.011  -1.806  -0.508  1.00  0.00      BH
ATOM      6  CB  ALA B   1       1.315  -1.243  -1.914  1.00  0.00      BH
ATOM      7  HB1 ALA B   1       1.652  -2.217  -2.273  1.00  0.00      BH
ATOM      8  HB2 ALA B   1       0.445  -1.015  -2.585  1.00  0.00      BH
ATOM      9  HB3 ALA B   1       2.022  -0.480  -2.148  1.00  0.00      BH
ATOM     10  C   ALA B   1       1.877  -1.479   0.519  1.00  0.00      BH
ATOM     11  O   ALA B   1       2.204  -0.655   1.349  1.00  0.00      BH
ATOM     12  N   ALA B   2       2.563  -2.642   0.294  1.00  0.00      BH
ATOM     13  HN  ALA B   2       2.354  -3.219  -0.488  1.00  0.00      BH


PSF

       1 !NTITLE
 REMARKS original generated structure x-plor psf file
```

```
104 !NATOM
  1 BH   1    ALA  N    NH3   -0.620000       14.0070           0
  2 BH   1    ALA  HT2  HC     0.310000        1.0080           0
  3 BH   1    ALA  HT3  HC     0.310000        1.0080           0
  4 BH   1    ALA  CA   CT1   -0.100000       12.0110           0
  5 BH   1    ALA  HA   HB     0.100000        1.0080           0
  6 BH   1    ALA  CB   CT3   -0.270000       12.0110           0
  7 BH   1    ALA  HB1  HA     0.090000        1.0080           0
  8 BH   1    ALA  HB2  HA     0.090000        1.0080           0
  9 BH   1    ALA  HB3  HA     0.090000        1.0080           0
 10 BH   1    ALA  C    C      0.510000       12.0110           0
 11 BH   1    ALA  O    O     -0.510000       15.9990           0
 12 BH   2    ALA  N    NH1   -0.470000       14.0070           0
 13 BH   2    ALA  HN   H      0.310000        1.0080           0
```

3.2.2 Configuration files

An initial and restart configuration file are needed per solvent per molecule. As an example, in the case of running ASMD (any case with more than 1 stage of SMD), the following template files would be required in the following locations:

Example: To study decaalanine, assigned a label "da", in three solvents, the following configuration files are required:

```
asmd2/00.maindir/namd/mol.conf/da/01.vac/smd_initial.namd
asmd2/00.maindir/namd/mol.conf/da/01.vac/smd_continue.namd

asmd2/00.maindir/namd/mol.conf/da/02.imp/smd_initial.namd
asmd2/00.maindir/namd/mol.conf/da/02.imp/smd_continue.namd

asmd2/00.maindir/namd/mol.conf/da/03.exp/smd_initial.namd
asmd2/00.maindir/namd/mol.conf/da/03.exp/smd_continue.namd

Code:
#################################################################
## JOB DESCRIPTION                                             ##
#################################################################
# SMD simulation (stretching) of deca-alanine in vacuum
# Constant temperature
```

```
#####################################################################
## ADJUSTABLE PARAMETERS                                           ##
#####################################################################
structure          ../../../../00.struc/01.vac/00.psf
coordinates        ../../../../00.struc/01.vac/00.pdb
outputName         daOut
#####################################################################
## SIMULATION PARAMETERS                                           ##
#####################################################################
# Input
seed                xxxxx
paraTypeCharmm     on
parameters          ../../../../toppar/par_all27_prot_lipid.prm
temperature         300

# Force-Field Parameters
exclude             scaled1-4
1-4scaling          1.0
cutoff              12.0
switching           on
switchdist          10.0
pairlistdist        13.5


#####################################################################
## JOB DESCRIPTION                                                 ##
#####################################################################
# SMD simulation (stretching) of deca-alanine in vacuum
# Constant temperature
#####################################################################
## ADJUSTABLE PARAMETERS                                           ##
#####################################################################
structure          ../../../../00.struc/01.vac/00.psf
coordinates        ../00.coor
outputName         daOut
#####################################################################
## SIMULATION PARAMETERS                                           ##
#####################################################################
# Input
seed                xxxxx
```

```
paraTypeCharmm     on
parameters          ../../../../toppar/par_all27_prot_lipid.prm
velocities          ../00.vel

# Force-Field Parameters
exclude             scaled1-4
1-4scaling          1.0
cutoff              12.0
switching           on
switchdist          10.0
pairlistdist        13.5
```

4 py_gen

4.1 del.py

```
#!/usr/bin/env python
import sys,os,glob

low = int(sys.argv[1])
high= int(sys.argv[2])

os.system('qstat -u dmerz3 > tmpjobs.txt')
f = open('tmpjobs.txt','r')
for line in f.readlines()[5:]:
    print line.split('.')[0]
    val=int(line.split('.')[0])
    if (val>=low) and (val<=high):
        print 'match'
        os.system('qdel %d' % val)
f.close()
```

4.2 pipe.py

```
def qsub_job(stage,job_path):
    if stage==stages[0]:
        dep_args = []
        pipe=subprocess.Popen([qsub_path] + dep_args +
                              [job_path],stdin=subprocess.PIPE, \
            stdout=subprocess.PIPE,stderr=subprocess.STDOUT)
```

```
        stout, stderr = pipe.communicate()
        print stout
        print stout.split('.')[0]
        print 'stderr >> ',stderr
        jdict[stage]=stout.split('.')[0]
    elif stage!=stages[0]:
        entry=str(int(stage)-1).zfill(2)
        print cdict[entry]
        job_deps = ':'.join(cdict[entry])
        dep_args = ['-W depend=afterany:%s' % job_deps]
        print 'dep_args',dep_args
        pipe=subprocess.Popen([qsub_path] + dep_args +
                          [job_path],stdin=subprocess.PIPE, \
            stdout=subprocess.PIPE,stderr=subprocess.STDOUT)
        stout, stderr = pipe.communicate()
        print stout
        print stout.split('.')[0]
        print 'stderr >> ',stderr
        jdict[stage]=stout.split('.')[0]
def qsub_jobc(stage,job_path):
    print jdict[stage]
    job_deps = ':'.join(jdict[stage])
    dep_args = ['-W depend=afterany:%s' % job_deps]
    print 'dep_args',dep_args
    pipe=subprocess.Popen([qsub_path] + dep_args +
                      [job_path],stdin=subprocess.PIPE, \
        stdout=subprocess.PIPE,stderr=subprocess.STDOUT)
    stout, stderr = pipe.communicate()
    print stout
    print stout.split('.')[0]
    print 'stderr >> ',stderr
    cdict[stage]=stout.split('.')[0]
def qsub_jobh(stage,job_path):
    print jdict[stage]
    job_deps = ':'.join(jdict[stage])
    dep_args = ['-W depend=afterany:%s' % job_deps]
    print 'dep_args',dep_args
    pipe=subprocess.Popen([qsub_path] + dep_args +
                      [job_path],stdin=subprocess.PIPE, \
        stdout=subprocess.PIPE,stderr=subprocess.STDOUT)
```

```
    stout, stderr = pipe.communicate()
    print stout
    print stout.split('.')[0]
    print 'stderr >> ',stderr
    #cdict[stage]=stout.split('.')[0]

def find_job(vel,solv,st):
    for path in glob(os.path.join(my_dir,'*.%s/%s/%s/*/job.sh'%(solv,vel,st))):
        num=(path.split('/')[-4])
        sol=(path.split('/')[-5]).split('.')[1]
        stg=(path.split('/')[-3])
        jtype=num+sol+stg
        acc.append(jtype)
        root='/'.join(path.split('/')[:-1])
        #root='/'+'/'.join(path.split('/')[2:-1])
        #path='/'+'/'.join(path.split('/')[2:])
        print root
        print path
        os.chdir(root)
        qsub_job(stg,path)
    for path in glob(os.path.join(my_dir,'*.%s/%s/%s-job.sh'%(solv,vel,st))):
        num=(path.split('/')[-2])
        sol=(path.split('/')[-3]).split('.')[1]
        stg=(path.split('/')[-1]).split('-')[0]
        root='/'.join(path.split('/')[:-1])
        #root='/'+'/'.join(path.split('/')[2:-1])
        #path='/'+'/'.join(path.split('/')[2:])
        print root
        print path
        os.chdir(root)
        qsub_jobc(stg,path)
    for path in glob(os.path.join(my_dir,'*.%s/%s/%s-jobh.sh'%(solv,vel,st))):
        num=(path.split('/')[-2])
        sol=(path.split('/')[-3]).split('.')[1]
        stg=(path.split('/')[-1]).split('-')[0]
        root='/'.join(path.split('/')[:-1])
        #root='/'+'/'.join(path.split('/')[2:-1])
        #path='/'+'/'.join(path.split('/')[2:])
        print root
        print path
```

```
        os.chdir(root)
        qsub_jobh(stg,path)

# submitted 02:
# submitted 03:
velocities = ['02']
solvents   = ['vac']   #solvents = ['imp'] #solvents = ['exp'] #one at a time
stages     = ['01','02','03','04','05','06','07','08','09','10']
# MAIN SUBMISSION CALL
# alternatively, qsub_job('01','vac')
[find_job(v,s,st) for v in velocities for s in solvents for st in stages]
```

4.3 lockfile.py

4.4 plotpkl.py

```
def plot_pmf(data,st):
    if st=='01':
        print data.shape[0]
        phase = int(st)-1
        deltaf= np.log(np.exp(data[::,::,3]*beta).mean(axis=0))*(1/beta)
        d = np.linspace(spos,spos+domain[phase],deltaf.shape[0]) # stg 1 specific
        lb = st+' '+str(data.shape[0])
        plt.plot(d,deltaf,'r-',linewidth=4.0,label='PMF')
        plt.plot(d,deltaf,'k--',linewidth=1.4)
    else:
        print data.shape[0]
        phase = int(st)-1
        deltaf= np.log(np.exp(data[::,::,3]*beta).mean(axis=0))*(1/beta)
        d = np.linspace(spos+domain[phase-1],spos+domain[phase],deltaf.shape[0])
        lb = st+' '+str(data.shape[0])
        plt.plot(d,deltaf,'r-',linewidth=4.0)
        plt.plot(d,deltaf,'k--',linewidth=1.4)
```

4.5 plothb.py

```
def pack(stage):
    seed_bond={}
    wght_bond={}
    wrk_pkl={}
    wrk_pkl=pickle.load(open('%s-sfwf.pkl' % stage,'rb'))
```

```
    for path in glob(os.path.join(my_dir,'%s/*/*-hb_pr*pr*.pkl.*' % stage)):
        seed = path.split('.')[-1]
        sample_i = pickle.load(open(path,'rb'))
        bond_clist=[]
        for i in range(len(sample_i)):
            cnt = len(sample_i[i])
            bond_clist.append(cnt)
        seed_bond[seed]=np.array(bond_clist)
    seeds = wrk_pkl[stage].keys()
    for s in seeds:
        sample_w = np.exp(wrk_pkl[stage][s][1][::,3]*beta).astype(float)
        sample_b = (seed_bond[s]).astype(float)
        lenf_w = len(sample_w)/100
        lenf_b = len(sample_b)/100
        print lenf_w,lenf_b
        B_list=[]
        W_list=[]
        for b in range(len(sample_b)):
            wv = int(((b+1)/lenf_b)*lenf_w)
            sum_B=(sample_b[b]*np.exp(beta*sample_w[wv]))
            sum_W=(np.exp(beta*sample_w[wv]))
            print sum_B,sum_W
            B_list.append(sum_B)
            W_list.append(sum_W)
        avg_B=np.array(B_list).cumsum()/np.array(W_list).cumsum()
        wght_bond[s]=avg_B
        #plot_hb_bluedot(avg_B[::2],stage,'b.',0.1)
    wb_vecs = np.array(wght_bond.values()).mean(axis=0)
    plot_hb(wb_vecs,stage,'k-',2)
    print type(wb_vecs)
    print len(wb_vecs)
#_______________________________________________________________________________
def plot_pkl(stage,sel,acc_d,acc_b,index=0,color='k-',b_label='hydrogen bonds'):
    phase=int(stage)-1
    def residue_index(label):
        return int(re.sub("[^0-9]","",label))
    def charac_bond2(trajectory,distance_target):
        acc_count_frames = []
        for frame in trajectory:
            acc_count = 0
```

```
        for bond in frame:
            distance = residue_index(bond[2])-residue_index(bond[3])
            if distance == distance_target:
                acc_count += 1
        acc_count_frames.append(acc_count)
    return acc_count_frames
#_____________________________________________________________________________
if sel != 'ihb':     # sel ==  'wp', 'hb'
    dct_sd_hb=pickle.load(open('%s-sd_%s.pkl' % (stage,sel),'rb'))
    print '%s-sd_%s.pkl' %(stage,sel)     # pkl: sd_hb or sd_wp
    seeds = dct_sd_hb.keys()
    acclens=[]
    for s in seeds:
        acc=[]
        sample_i = dct_sd_hb[s]  # trajectory,dcd-length list with
        for c in range(len(sample_i[0])):     # width of bonds per frame
            hbc=len(sample_i[0][c]) # hbc-hydrogen-bond-count, 1 frame
            acc.append(hbc)      # acc: counts over full trajectory
        acclens.append(acc)      # acclens: all trajectories
    data = np.array(acclens).mean(axis=0)
    if stage=='01':
        d = np.linspace(spos,spos+domain[phase],data.shape[0])
    elif stage !='01':
        d = np.linspace(spos+domain[phase-1],spos+domain[phase],data.shape[0])
    # establish domain, by linspacing - vector same length as data(frames)
    acc_d.append(d)              # acc_d.append(d[2:-2])
    acc_b.append(data)           # acc_b.append(data[2:-2])
else: # sel == 'ihb'
    dct_sd_hb=pickle.load(open('%s-sd_%s.pkl' % (stage,sel[1:3]),'rb'))
    print '%s-sd_%s.pkl' %(stage,sel[1:3])
    seeds = dct_sd_hb.keys()
    b_data = np.array([[charac_bond2(dct_sd_hb[s][0],n) for s in seeds] \
                 for n in [3,4,5]])
    if stage=='01':
        d = np.linspace(spos,spos+domain[phase],b_data.shape[2])
    elif stage !='01':
        d = np.linspace(spos+domain[phase-1],spos+domain[phase], \
                              b_data.shape[2])
    acc_d.append(d)
    acc_b.append(b_data)
```

4.6 mpmf.py

```
        def plot_pmf(data,st,c_lin):
            if st=='01':
                print data.shape[0]
                phase = int(st)-1
                deltaf= np.log(np.exp(data[::,::,3]*beta).mean(axis=0))*(1/beta)
                if phase == 0:
                    d = np.linspace(spos,spos+domain[phase],deltaf.shape[0])
                else:
                    d = np.linspace(spos+domain[phase-1],spos+domain[phase],deltaf.sh
                lb = str(data.shape[0])+' '+method
                plt.plot(d,deltaf,'%s' % c_lin,linewidth=4.0,label=lb)
                #plt.plot(d,deltaf,'k--',linewidth=1.4)
            else:
                print data.shape[0]
                phase = int(st)-1
                deltaf= np.log(np.exp(data[::,::,3]*beta).mean(axis=0))*(1/beta)
                if phase == 0:
                    d = np.linspace(spos,spos+domain[phase],deltaf.shape[0])
                else:
                    d = np.linspace(spos+domain[phase-1],spos+domain[phase],deltaf.sh
                lb = st+' '+str(data.shape[0])
                plt.plot(d,deltaf,'%s' % c_lin,linewidth=4.0)
                #plt.plot(d,deltaf,'k--',linewidth=1.4)
```

4.7 weighthb.py

```
def plot_hb(avgB,st,color,lw):
    phase = int(st)-1
    if (st=='01'):
        d = np.linspace(spos,spos+domain[phase],avgB.shape[0])
        plt.plot(d,avgB,color,label="hydrogen bonds",linewidth=lw)
    elif st !='01':
        d = np.linspace(spos+domain[phase-1],spos+domain[phase],avgB.shape[0])
        plt.plot(d,avgB,color,linewidth=lw)
def plot_hb_bluedot(avgB,st,color,lw):
    phase = int(st)-1
    if (st=='01'):
        d = np.linspace(spos,spos+domain[phase],avgB.shape[0])
```

```
            plt.plot(d,avgB,color,linewidth=lw)
        elif st !='01':
            d = np.linspace(spos+domain[phase-1],spos+domain[phase],avgB.shape[0])
            plt.plot(d,avgB,color,linewidth=lw)

    def pack(stage):
        seed_bond={}
        wght_bond={}
        wrk_pkl={}
        wrk_pkl=pickle.load(open('%s-sfwf.pkl' % stage,'rb'))
        for path in glob(os.path.join(my_dir,'%s/*/*-hb_pr*pr*.pkl.*' % stage)):
            seed = path.split('.')[-1]
            sample_i = pickle.load(open(path,'rb'))
            bond_clist=[]
            for i in range(len(sample_i)):
                cnt = len(sample_i[i])
                bond_clist.append(cnt)
            seed_bond[seed]=np.array(bond_clist)
        seeds = wrk_pkl[stage].keys()
        for s in seeds:
            sample_w = np.exp(wrk_pkl[stage][s][1][::,3]*beta).astype(float)
            sample_b = (seed_bond[s]).astype(float)
            lenf_w = len(sample_w)/100
            lenf_b = len(sample_b)/100
            print lenf_w,lenf_b
            B_list=[]
            W_list=[]
            for b in range(len(sample_b)):
                wv = int(((b+1)/lenf_b)*lenf_w)
                sum_B=(sample_b[b]*np.exp(beta*sample_w[wv]))
                sum_W=(np.exp(beta*sample_w[wv]))
                print sum_B,sum_W
                B_list.append(sum_B)
                W_list.append(sum_W)
            avg_B=np.array(B_list).cumsum()/np.array(W_list).cumsum()
            wght_bond[s]=avg_B
            plot_hb_bluedot(avg_B[::2],stage,'b.',0.1)
        wb_vecs = np.array(wght_bond.values()).mean(axis=0)
        plot_hb(wb_vecs,stage,'k-',2)
        print type(wb_vecs)
```

```
print len(wb_vecs)
```