Welcome to asmd2

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

September 5, 2013

1 Quickstart

This assumes that a correctly equilibrated structure and correct namd files are located in:

```
asmd/00.maindir/namd/struc/da/01.vac/00.pdb
asmd/00.maindir/namd/struc/da/0.vac/00.pdb

asmd/00.maindir/namd/struc/da/02.imp/00.pdb
asmd/00.maindir/namd/struc/da/02.imp/00.pdb

asmd/00.maindir/namd/struc/da/03.exp/00.pdb
asmd/00.maindir/namd/struc/da/03.exp/00.pdb

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

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

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

1.1 Command Line

Command line: This creates all working directories used in asmd2

```
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.

```
[da]                    :Name of molecule which is used in the command line
jobid = decaalanine     :Job description, Names the working directories
dircounts= 5            :Number of directories per a velocity
mol = da                :Name of molecule
zcrd = 13.0             :Distance from the first CA to the last CA
envdist = 01.vac:zcrd,02.imp:zcrd,03.exp:zcrd   :Distance in each enviornment
dist = 20.0             :Distance protein is pulled
ts   = 2.0              :Time Step
n    = 2., 3.           :Pulling velocity, i.e, 2 = 100A/ns, 3 = 10A/ns
gate = ggatecpu2        :Specify computer cluster
environ = 01.vac        :Specify the enviornment to make working directories
howmany = 100,50,25,10,1   :How many trajectories per a directory, Based on
                            velocity,i.e., 100 corresponds to the 1000 A/ns,
                            50 to the 100 A/ns, 25 to the 10 A/ns, 10 to
                            the 1 A/ns, and 1 to the 0.1 A/ns
cn   = 2                :Specify number of nodes
wallt= lwt              :Specify the wall time, how long the job will run
queue= standby          :queue
ppn_env = 01.vac:1,02.imp:cn,03.exp:cn           :Specify nodes for each
                                                  enviornment
wt_env  = 01.vac:wallt,02.imp:mwt,03.exp:mwt     :Specify wall time for
                                                  each enviornment
q_env   = 01.vac:queue,02.imp:queue,03.exp:queue :Specify queue for each
                                                  enviornment
path_seg = 0.1          :How many stages, to determine number of stages it
                         must add up to 1
```

```
path_svel= 1.0          :Stage velocity, can reduce the stage velocity by
                         adjusting the value between 0 and 1
langevD = 5             :Langevin damping, damping coeffeicient (gamma)
temp = 300              :Temperature used for simulation
```

2 Description

2.1 Steered Molecular Dynamics

Steered Molecular Dynamics, pulling a peptide with an harmonic potential to obtain a potential of mean force (PMF). A 'pseudoatom' is connected by a harmonic potential to a specific site of the peptide is directed along an axis or through a configuration space at a constant velocity, during which time the force applied on the peptide by the 'pseudoatom' is recorded, thereby allowing one to calculate the work for a single trajectory and the PMF for multiple trajectories by boltzmann weighting the work and taking the average of that result.

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. After each stage the structure is relaxed and then pulled again with the starting coordinates from the last stage. This causes the work trajectory to not be as spread out as it would be if it was steered molecular dynamics causing the calculation of the PMF to be quicker and thus using fewer trajectories.

3 Setup for NAMD

Download a new molecule from the PDB(Protein Data Bank). If the peptide is missing residues, Modeller, a python program can be used to predict the missing residues. For more information on Modeller, visit their website at http://salilab.org/modeller/. If your peptide you want to work with, i.e. poly peptides, is not found on PDB, a VMD program, Molefacture, can be used to generate a protein structue. For more information on Molefacture, visit their website at http://www.ks.uiuc.edu/Research/vmd/plugins/molefacture/. Generate a .pdb file with added hydrogen using VMD and a protein structure file (.psf) with psfgen. For more information on creating a .pdb and .psf files, download the NAMD tutorial. 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.

3.1.2 go

The go.py script runs steered molecular dynamics any number of times. This script is called by the job.sh script.

3.1.3 hb

The hb.py script generates the pickle describing the bonding in a trajectory. Can edit the hb.py script to change the parameters of hydrogen bonding.

```
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(' ','')

#Parameters for H-Bonds
        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.

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 and can select which atom is connected to the psuedoatom and which atom is fixed, 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:
```

```
# Atoms selected for force application

set id1 [atomid U 1 N]     #U is the segment, 1 is the residue, N is the atom fixed
set grp1 {}
lappend grp1 $id1
set a1 [addgroup $grp1]

set id2 [atomid U 20 NT] #U is the segment, 20 is the residue, NT is the atom
                         #connected to psuedoatom
set grp2 {}
lappend grp2 $id2
set a2 [addgroup $grp2]

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
```

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
```

3.2 Adding a new molecule

A few key template files must be put into place! Equilibrated structures need to be in place, i.e.. .psf and .pdb files in the struc directory. The smd continue, initial, and force files need to be in place in the mol.conf directory. The smd force file contains the force constant of the harmonic potential and which atom is fixed.

```
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

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

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

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

3.2.1 Starting Structure

Equilibrated structures are are used for 00.pdb and 00.psf. 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

.pdb file
CRYST1    0.000    0.000    0.000   0.00   -NaN-154243993072403881666760839735915053O
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 file
       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. Also a forces file is needed. 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.tcl/da/01.vac/smd_initial.namd
asmd2/00.maindir/namd/mol.conf.tcl/da/01.vac/smd_continue.namd
asmd2/00.maindir/namd/mol.conf.tcl/da/01.vac/smd_force.tcl

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

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

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

Code:

# Atoms selected for force application

set id1 [atomid U 1 N]     #U is the segment, 1 is the residue, N is the atom fixed
set grp1 {}
lappend grp1 $id1
set a1 [addgroup $grp1]

set id2 [atomid U 20 NT] #U is the segment, 20 is the residue, NT is the atom
                         #connected to psuedoatom
set grp2 {}
lappend grp2 $id2
set a2 [addgroup $grp2]

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
```

4 py_gen

4.1 del.py

This is used to delete jobs that are running. Need to edit the user name. In the command line run del.py and give an argument with the numbers of the job id you want to cancell. Can put a range to delete multiple jobs.

```
./del.py arg1 arg2
```

Code:

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

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

os.system('qstat -u USER > tmpjobs.txt') #Change to specific user name
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

Pipe.py submits the jobs using a pbs resource manager. It also runs the jobs in parallel. The only part of the script that needs editing are the velocities and solvents.

```
    #_____________________________________________________________________
    def get_folder(f):
        try:
            return int(f)
        except:
            return ''

    dirs = [str(d) for d in sorted([get_folder(f) for f in os.listdir(my_dir) \
                          if os.path.isdir(f)])]
    print dirs

    velocities = ['02', '03']     #Edit velocity to run different velocities at once
    #velocities = ['02','03','04','05']
    solvents   = ['vac']          #Edit solvents to run different solvents at once
    #solvents   = ['vac','imp','exp']
    stages     = [str(x).zfill(2) for x in range(1,51)]
    # stages    = ['01','02','03','04','05','06','07','08','09','10']
    # alternatively, limit stages to ['01','02','03']
    # MAIN SUBMISSION CALL
    # alternatively, qsub_job('01','vac')
    [find_job(dirs[0],v,s,stages) for s in solvents for v in velocities]
    #[find_job(f,v,s,stages) for f in dirs for v in velocities for s in solvents]
    #_____________________________________________________________________
```

```
\subsection{run\_on\_laptop.py}

In order to run asmd2 on laptop: \\
 1) Make sure correct namd2 path is in .bashrc file.
    For example: \\
    \begin{verbatim}
    export NAMDHOME2=/Users/usr/Documents/NAMD/NAMD_2.9_MacOSX-x86_64-multicore
    export PATH=$PATH:$NAMDHOME2
```

2) Generate templates and folders by using './gen.py engine molecule'. In order to add new molecule, edit namd.gconf and use the template at top of the file.
For example:
./gen.py namd da
3) Once working directories are created, the top level of the directory is

where the script 'run_on_laptop.py' is located. Make neccessary modifications to the velocity and stages in run_on_laptop.py.
4) Run ./run_on_laptop.py on the same directory level as it is located when generated.

4.3 plotpkl.py

This plots the work and potential of mean force for the stages. In the command line run plotpkl.py and will graph all the stages. If the stages are currently running you can grpah the data that you have up to that stage just have to give it an argument of the stage in the command line.

In the command line, All stages finished: ./plotpkl.py

In the command line, Up to stage 6 finished ./plotpkl.py 06

```
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)
```

4.4 plothb.py

This plots the hydrogen bonds for the stages. In the command line run plothb.py and will graph all the stages. If the stages are currently running you can grpah the data that you have up to that stage just have to give it an argument of the stage in the command line.

In the command line, All stages finished: ./plothb.py

In the command line, Up to stage 6 finished ./plothb.py 06

```
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)
```

4.5 mpmf.py

This script allows you to graph multiple PMF's on the same plot.

```
    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)
```

4.6 weighthb.py

This plots the weighted hydrogen bonds.

```
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)
```