Transition path computation using the string method

Requirements: User must be well versed in the Gromacs to run molecular dynamics simulations, Plumed software for performing advance sampling, Bash scripting, and basic python coding.

Follow the below flow chart to generate initial string and to decide the parameters for further string evolution.

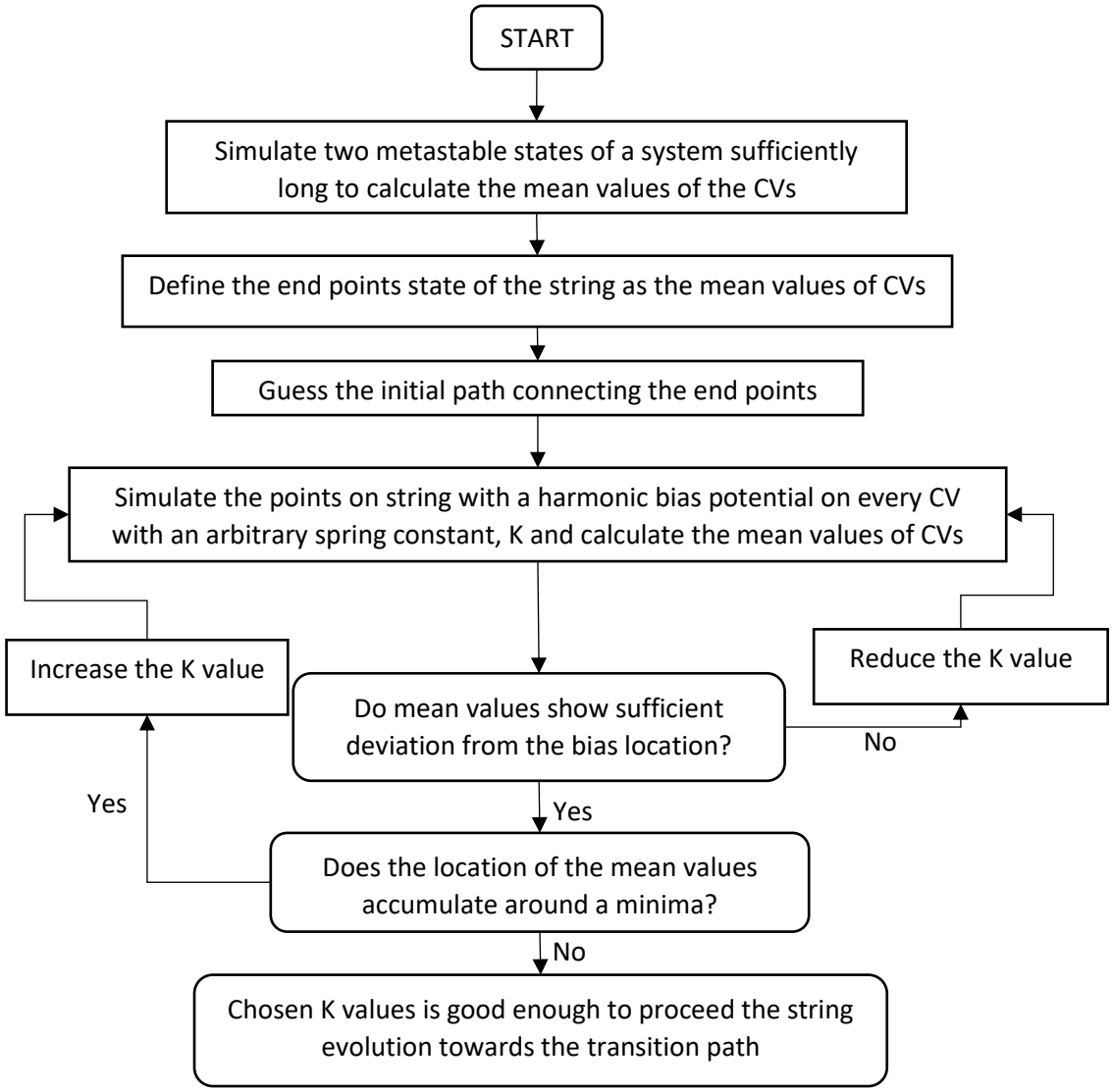

Figure 1: Flow chart-1 to decide the K value and setting up initial string.

Generate the end points using conventional MD simulations and initial guess of path connecting the end states. Initial guess of path is written in “phi-psi-ite-1” file column wise. Then using the below Bash script (*initial.sh*) simulate the initial path with harmonic potential in each CV. Following the script would generate N number of folders (N is number of points on string) containing Plumed output “COLVAR” files and other MD simulations output.

You must have topology files, force field files, structure files (in md0 folder named as md0), phi-psi-ite-1 file, and mdp parameter files in the working directory.

Initial.sh

```
say=0
add=1
for i in {1..27} #Alalnie dipeptide with 27 points on string
do
mkdir md$i
cd md$i
cp ../phi-psi-ite-1 ./
gmx grompp-f ../grompp.mdp-c ../md$say/md$say.gro-p ../topol.top-o md$i.tpr
say=$(($say+$add))
awk "NR==$say{print;exit}" phi-psi-ite-1 > test.dat
phi=$(awk 'NR == 1 {printf "%f\n",$1}' test.dat)
psi=$(awk 'NR == 1 {printf "%f\n",$2}' test.dat)
rm test.dat

#Following plumed script is for alanine-dipeptide which can be written for any other #collective variables
present in the Plumed software
cat >plumed.dat << EOF
phic: TORSION ATOMS=5,7,9,15
psic: TORSION ATOMS=7,9,15,17
#
# Impose an umbrella potential on CV 1 and CV2
# with a spring constant of 80 kjoule/mol
# at fixed points on the Ramachandran plot
#
restraint-phi: RESTRAINT ARG=phic KAPPA=80 AT=$phi
restraint-psi: RESTRAINT ARG=psic KAPPA=80 AT=$psi
# monitor the two variables and the bias potential from the two restraints
PRINT STRIDE=10 ARG=phic,psic FILE=COLVAR
EOF
gmx mdrun-s md$i.tpr-deffnm md$i-plumed plumed.dat-nsteps 10000-c md$i.gro-e md$i.edr-g md$i.log-v

cd ..
done
```

After executing the initial.sh, analyse the COLVAR files in each md folders for finalising the K value (follow flow chart-1). For alanine-dipeptide, 80 was found to be good enough.

Once you decide the K value, follow the below flow chart-2.

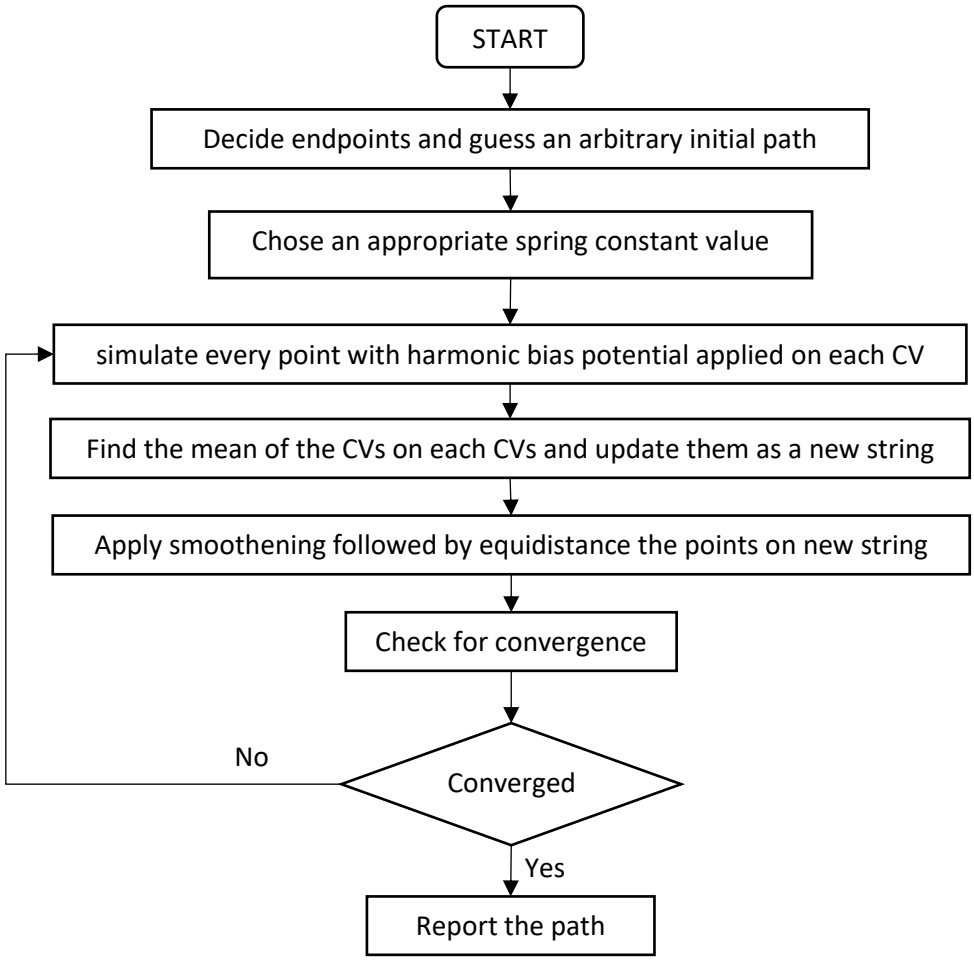

Figure 2: Flow chart-2 for evolving the string towards MFEP.

Once K value has been decided, we perform our first-string evolution using “firsh.sh” bash script. This requires a python script “new.py” which would calculate the mean and write them in “phi-psi-ite-2” files.

New.py

```
from __future__ import division
import numpy as np

f=open("COLVAR","r")
lines=f.readlines()
phi=[]
psi=[]

for x in lines:
   phi.append(x.split(' ')[2])
   psi.append(x.split(' ')[3])
f.close()
```

```
phi=np.array(phi).astype(np.float)
psi=np.array(psi).astype(np.float)

phi_new=np.mean(phi)
psi_new=np.mean(psi)

with open("/WORKING-DIRECTORY/phi-psi-ite-2", "a") as myfile:
   myfile.write("%.11f \t"%phi_new)
   myfile.write("%.11f \n"%psi_new)
myfile.close()
```

First.sh

```
#!!---------------------------------!!#
#Finding new CVs for first iteration
#At inner points on string
for AT in {2..26} #Loop on the points of string except the end points as they are fixed
do

cp new.py ./md$AT/.
cd ./md$AT/.

sed-i '/^#/d' COLVAR  #To remove first line containing string so that python can read the file

python2 new.py

cd ..
done
```

First.sh would generate a second set of CVs in phi-psi-ite-2 file to simulate. Before proceeding, you must add the first and last string point (first and last row of phi-psi-ite-1 respectively) to the phi-psi-ite-2 file, and make the following changes in new.py file of 2nd and N-1 folder.

MAKE THE FOLLOWING CHANGES (in last section of new.py)

```
with open("/WORKING-DIRECTORY/phi-psi-ite-2", "a") as myfile:
   myfile.write("%.11f \t"%phi_new)
   myfile.write("%.11f \n"%psi_new)
myfile.close()
```

TO

FOR 2nd FOLDER

```
with open("/WORKING-DIRECTORY/phi-psi-ite-2", "a") as myfile:
   myfile.write("phi-1\tpsi-1\n")  #where phi-1 and psi-1 are the first data points
   myfile.write("%.11f \t"%phi_new)
   myfile.write("%.11f \n"%psi_new)
myfile.close()
```

FOR (N-1)th FOLDER

```
with open("/WORKING-DIRECTORY/phi-psi-ite-2", "a") as myfile:
```

```
    myfile.write("%.11f \t"%phi_new)
    myfile.write("%.11f \n"%psi_new)
    myfile.write("phi-N\tpsi-N\n")  #where phi-N and psi-N are the last (N^th) data points
myfile.close()
```

phi-psi-ite-2 files only contains the mean of sampling on each point; however, we need to apply smoothening and make points equidistant. Also, check for the convergence with a tolerance value. We would need “equispacing.py” file to do the job. We will call equispacing.py python script in second.sh bash script.

Equispacing.py

```
import numpy as np

tol=0.000001 #Define tolerance for the convergence here

f1=open("phi-psi-ite-1","r") #reading old phi-psi-ite
lines1=f1.readlines()
phio=[]
psio=[]
#print(lines)
for x in lines1:
    phio.append(x.split('\t')[0])
    psio.append(x.split('\t')[1])
f1.close()

phio=np.array(phio).astype(np.float)
psio=np.array(psio).astype(np.float)

f=open("phi-psi-ite-2","r") #resing new generated phi-psi-ite
lines=f.readlines()
phi=[]
psi=[]
#print(lines)
for x in lines:
    phi.append(x.split('\t')[0])
    psi.append(x.split('\t')[1])
f.close()

phi=np.array(phi).astype(np.float)
psi=np.array(psi).astype(np.float)

#***************SMOOTHING*****************#
count=len(phi)
phi_n=np.zeros_like(phi)
psi_n=np.zeros_like(phi)
phi_n[0]=phi[0]
phi_n[-1]=phi[-1]
psi_n[0]=psi[0]
psi_n[-1]=psi[-1]
for i in range (count-2):
    j=(i+1)
    k=(i+2)
    phi_n[j]=((phi[i]+phi[j]+phi[k])/3)
```

```
    psi_n[j]=((psi[i]+psi[j]+psi[k])/3)
phi=phi_n
psi=psi_n
#**************END OF SMOOTHING************#

#********* Equidistance the points**************#
dx=(phi-np.roll(phi,1))
dy=(psi-np.roll(psi,1))

#print(dx)
dx[0]=0
dy[0]=0

lxy=np.zeros_like(phi)
count=len(phi)
for i in range (count):
    lxy[i]=lxy[i-1]+(np.sqrt(pow(dx[i],2)+pow(dy[i],2)))

for i in range (count):
    lxy[i]=lxy[i]/lxy[count-1]

inter=np.linspace(0,1,count)

phi=np.interp(inter,lxy,phi)
psi=np.interp(inter,lxy,psi)

#*********END of equidistance the points**************#

with open("/WORKING-DIRECTORY/phi-psi-ite-2", "w") as myfile:
    for i in range (count):
        myfile.write("%.11f \t"%phi[i])
        myfile.write("%.11f \n"%psi[i])
myfile.close()

#********Tolerance check************#
diffx=0
diffy=0
diffx += pow((phi-phio),2)
diffy += pow((psi-psio),2)
diffx=sum(diffx)
diffy=sum(diffy)

tol_check=(np.sqrt(diffx+diffy))/count

if(tol_check<=tol):
    binary=1
else:
    binary=0

with open("/WORKING-DIRECTORY/tol_check", "w") as my:
    my.write("%d\n"%binary)
my.close()
```

second.sh

```
python2 equispacing.py #Making point equispaced and smoothen the string
```

```
mkdir phi-psi-ite #Directory to store the string positions at every iteration
mv phi-psi-ite-1 ./phi-psi-ite/
rename -v 's/phi-psi-ite-2/phi-psi-ite-1/' phi-psi-ite-2 phi-psi-ite-1

#!!-----------------------------------------------!!#
#First Iteration at every point on string
ite=1
say=0
add=1
for AT in {2..26}
do

mkdir ./md$AT/iteration-$ite
cp topol.top ./md$AT/iteration-$ite/. #Copying topology file
cp grompp.mdp ./md$AT/iteration-$ite/. #Copying mdp parameters
cp minim.mdp ./md$AT/iteration-$ite/.  #Copying mdp parameters for minimisation
cp phi-psi-ite-1 ./md$AT/iteration-$ite/. #Copying phi psi values to applying restraint on

cd ./md$AT/
cp md$AT.gro ./iteration-$ite/. #Copying structure file from previous iteration
cp new.py ./iteration-$ite/.
cd ./iteration-$ite/

#gmx grompp -f minim.mdp -c md$AT.gro -p topol.top -o em.tpr #In case of energy minimization is required, generally this step is not necessary

#gmx mdrun -deffnm em #If simulations crashes due to bad contacts, one can energy minimise the structure, this was not required for alanine-dipeptide (it's just a fail-safe)

gmx grompp -f grompp.mdp -c md$AT.gro -p topol.top -o md$ite.tpr
#If performing energy minimisation use -c em.gro instead of -c md$AT.gro

say=$(($say+$add))
say=$(($say+$add))
awk "NR==$say{print;exit}" phi-psi-ite-1 > test.dat
phi_old=$(awk 'NR == 1 {printf "%f\n",$1}' test.dat)
psi_old=$(awk 'NR == 1 {printf "%f\n",$2}' test.dat)
rm test.dat

cat >plumed.dat << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

restraint-phi: RESTRAINT ARG=phi KAPPA=80 AT=$phi_old
restraint-psi: RESTRAINT ARG=psi KAPPA=80 AT=$psi_old

PRINT STRIDE=1 ARG=phi, psi, FILE=COLVAR
EOF

gmx mdrun -s md$ite.tpr -deffnm md$ite -plumed plumed.dat -nsteps 100000 -c md$ite.gro -e md$ite.edr -g md$ite.log -v

FILE="md$ite.gro"  #To check if the simulation was completed, if not, you can add some condition in else statement

if [ -f "$FILE" ];
```

```
then
  echo "File $FILE exist."
else                            #Look for what can be changed
  echo "File $FILE does not exist" >&2
  cd ../..
  cp grompp2.mdp ./md_phi$AT/iteration-$ite/.
  cd ./md_phi$AT/iteration-$ite/
  gmx grompp-f grompp2.mdp-c em.gro-p topol.top-o md$ite.tpr
  gmx mdrun-s md$ite.tpr-deffnm md$ite-plumed plumed.dat-nsteps 100000-c md$ite.gro-e md$ite.edr-g
md$ite.log
fi

sed-i '/^#/d' COLVAR

say=$(echo $(( say-add )))

python2 new.py

cd /WORKING-DIRECTORY/

done
```

If second.sh runs smoothly, we can proceed for “third.sh” bash script which would automatically do rest of the iterations without user interference. third.sh is essential same as second.sh with an additional loop for number of iterations start from second.

Third.sh

```
for ite in {2..N} #loop over number of iterations “N”
do
tol_check=$(awk 'NR == 1 {printf "%f\n",$1}' tol_check)
echo "$tol_check"

#Checking tolerance coming from equispacing.py
if (( $(echo "$tol_check == 1" |bc-l) )); then
    echo "Hurrah! Converged."
    break
fi

say=0
add=1
gro=$(echo $(( ite-add )))
python2 equispacing.py

mv phi-psi-ite-1 phi-psi-ite-new
mv phi-psi-ite-new ./phi-psi-ite/
cd ./phi-psi-ite/.
mv phi-psi-ite-new phi-psi-ite-$ite
cd ..
rename-v 's/phi-psi-ite-2/phi-psi-ite-1/' phi-psi-ite-2 phi-psi-ite-1

for AT in {2..26}
do

mkdir ./md$AT/iteration-$ite/
```

```
cp topol.top ./md$AT/iteration-$ite/.
cp minim.mdp ./md$AT/iteration-$ite/.
cp grompp.mdp ./md$AT/iteration-$ite/.
cp phi-psi-ite-1 ./md$AT/iteration-$ite/.

cd ./md$AT/
cp ./iteration-$gro/md$gro.gro ./iteration-$ite/.
cp new.py ./iteration-$ite/.
cd ./iteration-$ite/

#gmx grompp-f minim.mdp-c md$gro.gro-p topol.top-o em.tpr

#gmx mdrun-deffnm em

gmx grompp-f grompp.mdp-c md$gro.gro-p topol.top-o md$ite.tpr  #Note, we are using the structure file from the previous iteration

say=$(($say+$add))
say=$(($say+$add))
awk "NR==$say{print;exit}" phi-psi-ite-1 > test.dat
phi=$(awk 'NR == 1 {printf "%f\n",$1}' test.dat)
psi=$(awk 'NR == 1 {printf "%f\n",$2}' test.dat)
rm test.dat

cat >plumed.dat << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

restraint-phi: RESTRAINT ARG=phi KAPPA=80 AT=$phi
restraint-psi: RESTRAINT ARG=psi KAPPA=80 AT=$psi

PRINT STRIDE=1 ARG=phi, psi, FILE=COLVAR
EOF

gmx mdrun-s md$ite.tpr-deffnm md$ite-plumed plumed.dat-nsteps 10000-c md$ite.gro-e md$ite.edr-g md$ite.log-v

FILE="md$ite.gro"

if [-f "$FILE" ];
then
  echo "File $FILE exist."
else
  echo "File $FILE does not exist" >&2
  cd ../..
  cp grompp2.mdp ./md_phi$AT/iteration-$ite/.
  cd ./md_phi$AT/iteration-$ite/
  gmx grompp-f grompp2.mdp-c em.gro-p topol.top-o md$ite.tpr
  gmx mdrun-s md$ite.tpr-deffnm md$ite-plumed plumed.dat-nsteps 10000-c md$ite.gro-e md$ite.edr-g md$ite.log
fi

sed-i '/^#/d' COLVAR

say=$(echo $(( say-add )))

python2 new.py
```

```
cd /WORKING-DIRECTORY/
done
done
```

third.sh bash script would stop as soon as the string converges to a solution. However, most of the time, it will not converge if the tolerance value in "equispacing.py" is small. In that case, one can get into "phi-psi-ite" folder and see the evolution of each CV. CVs start to fluctuates on MFEP due to thermal energy. When you see them fluctuating around a value, you can take average of last few iterations to report a final converged string.

NOTE: phi-psi-ite-(N-1) file in phi-psi-ite folder is corresponding to the iteration-N in each string points folders (mdN).

To generate the free energy surface, one can perform WHAM analysis using existing COLVAR files generated during the string evolution. And, report the free energy surface along with the 1D free energy profile along the converged string.

To generate the structure on the path, one can use a very high spring constant and generate structure on the path or one can pick up the corresponding structure files from the last iteration.

Python code to check the convergence:

```
import numpy as np
import matplotlib.pyplot as plt

num=7     #Number of iterations
rows=12 #Number of points on the string
phi=[]; psi=[];
for count in range(1,(num+1)):
   f=open("phi-psi-ite-" + str(count), 'r')
   lines=f.readlines()
   for x in lines:
      phi.append(x.split('\t')[0])
      psi.append(x.split('\t')[1])
   f.close()

phi=np.array(phi).astype(np.float)
psi=np.array(psi).astype(np.float)

sigma_x=[]
sigma_y=[]

for j in range(rows,(num*rows),rows): #loop over the iterations (first value of phi-psi)
   sigma_xx=0
   sigma_yy=0
   for i in range(j,(j+rows)): #loop over a point on string (tartuing from first value to the last value)
      sigma_xx=sigma_xx+np.abs(phi[i]-phi[(i-rows)])
      sigma_yy=sigma_yy+np.abs(psi[i]-psi[(i-rows)])

   sigma_x.append((sigma_xx)/rows)
   sigma_y.append((sigma_yy)/rows)
```

```
sigma_x=[(i) for i in sigma_x]
sigma_y=[(i) for i in sigma_y]

with open("convergence.dat", "a") as myf:
    for i in range (0,len(sigma_x),1):
        myf.write("%.11f \t"%sigma_x[i])
        myf.write("%.11f \n"%sigma_y[i])
myf.close()

fig=plt.figure(figsize=(9,10))
plt.subplot(2,1,1)
plt.plot(sigma_x, 'ko-', label='ALPHARMSD')
plt.xlabel("Iteration Number",fontsize=20)
plt.ylabel("Convergence", fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.legend()

plt.subplot(2,1,2)
plt.plot(sigma_y, 'ro-', label='$d_z$')
plt.xlabel("Iteration Number",fontsize=20)
plt.ylabel("Convergence", fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.legend()
plt.savefig('convergence_absolute_deviation.png', dpi=300)
```

I used the Alan gross filed WHAM code to get the free energy landscape, please check the link for documentation (http://membrane.urmc.rochester.edu/?page_id=126)

The code is required the input of files in a specific format which is cumbersome to write. I have written a code to automatically generate the input file which has to be executed in the phi-psi-ite folder.

WHAM input generator:

```
import numpy as np
import matplotlib.pyplot as plt

myfile= open("adp-data.dat", "a")

for count in range(3,11):
    phi=[]; psi=[]
    f=open("phi-psi-ite-" + str(count), 'r')
    lines=f.readlines()
    for x in lines:
        phi.append(x.split('\t')[0])
        psi.append(x.split('\t')[1])
    f.close()

    phi=np.array(phi).astype(np.float)
    psi=np.array(psi).astype(np.float)

    for i in range(1,(len(phi)-1)):
```

```
        myfile.write('/home/avijeet/Documents/data/work/SM_maxima/adp/charmm22/272k/tue/'+'md'+str(i+1)+
'/iteration-'+str(count-1)+'/COLVAR\t'+str(phi[i])+'\t'+str(psi[i])+'\t80\t80\n')
    myfile.write('\n')
myfile.close()
```