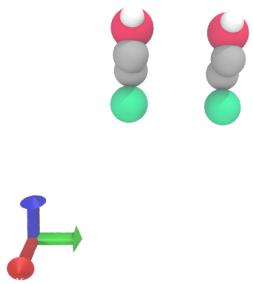
Sampling the Pair Potential Space

May 26, 2017

In this case we have two MeOH molecules separated by 5Å in the y direction.



Our MeOH molecules have the following coordinates:

S $(0.138, 0.0, -1.654)$	S' $(0.138, 5.0, -1.654)$
$C1$ $(0.474, 0.0, 1.059)$	$C1'$ $(0.474, 5.0, 1.059)$
$C2$ $(-0.621, 0.0, -0.007)$	$C2'$ $(-0.621, 5.0, -0.007)$
O $(-0.125, 0.0, 2.357)$	O' $(-0.125, 5.0, 2.357)$
H $(0.598, 0.0, 2.999)$	H' $(0.598, 5.0, 2.999)$

```
S=c(2,0.0,0.138,0.0,-1.654)
C1=c(1,0.0,0.474,0.0,1.059)
C2=c(1,0.265,-0.621,0.0,-0.007)
O=c(3,-0.700,-0.125,0.0,2.357)
H=c(4,0.435,0.598,0.0,2.999)
S_2=c(2,0.0,0.138,5.0,-1.654)
C1_2=c(1,0.0,0.474,5.0,1.059)
C2_2=c(1,0.265,-0.621,5.0,-0.007)
```

```
O_2=c(3,-0.700,-0.125,5.0,2.357)
H_2=c(4,0.435,0.598,5.0,2.999)
```

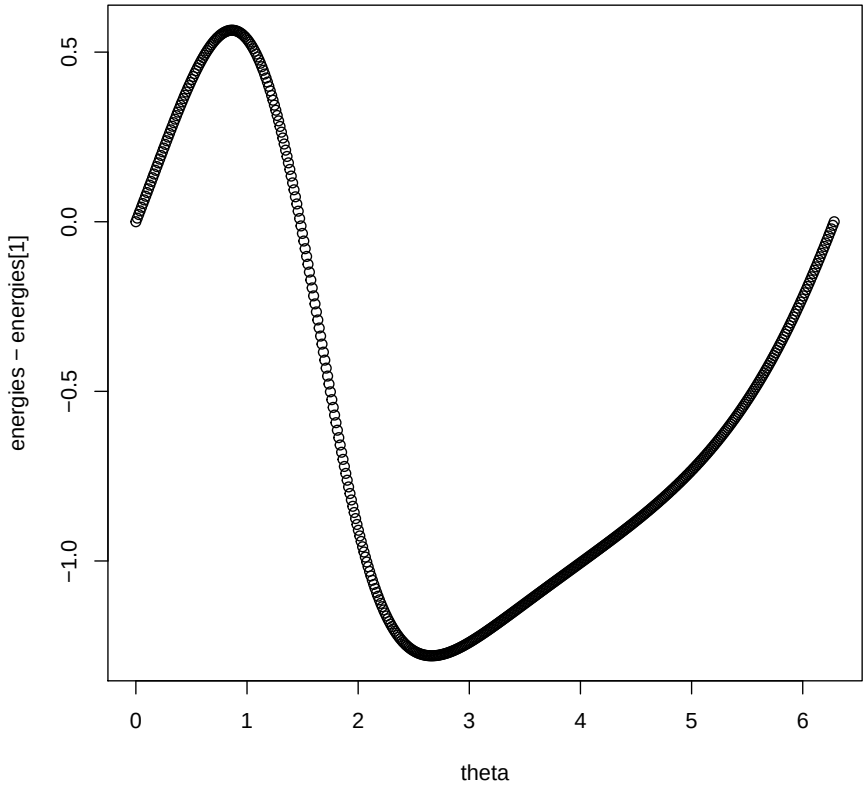
Rotating the *SCCO* dihedral we get:

```
library(onion)
axis = C2[3:5]-C1[3:5]
normaxis = axis/sqrt(sum(axis^2))
numpoints = 500
theta = seq(0,2*pi,length.out = numpoints)
#q = sapply(theta,function(x) cos(x/2)+sin(x/2)*normaxis[1]*Hi+sin(x/2)*normaxis[2]*Hj+si
q = cos(theta/2)+sin(theta/2)*normaxis[1]*Hi+sin(theta/2)*normaxis[2]*Hj+sin(theta/2)*norm
#print(q)
```

```
v1 = O[3:5] - C2[3:5]
v2 = H[3:5] - C2[3:5]
v1_q = 0 + v1[1] * Hi + v1[2] * Hj + v1[3] * Hk
v2_q = 0 + v2[1] * Hi + v2[2] * Hj + v2[3] * Hk
v1_rot_q = q * v1_q * Conj(q)
v2_rot_q = q * v2_q * Conj(q)
v1_rot = data.frame(x = i(v1_rot_q), y = j(v1_rot_q), z = k(v1_rot_q))
v2_rot = data.frame(x = i(v2_rot_q), y = j(v2_rot_q), z = k(v2_rot_q))
# print(v1_rot)
Onew = v1_rot + matrix(rep(C2[3:5], times = numpoints), ncol = 3, byrow = TRUE)
Hnew = v2_rot + matrix(rep(C2[3:5], times = numpoints), ncol = 3, byrow = TRUE)
# print(Onew) print(Hnew)
```

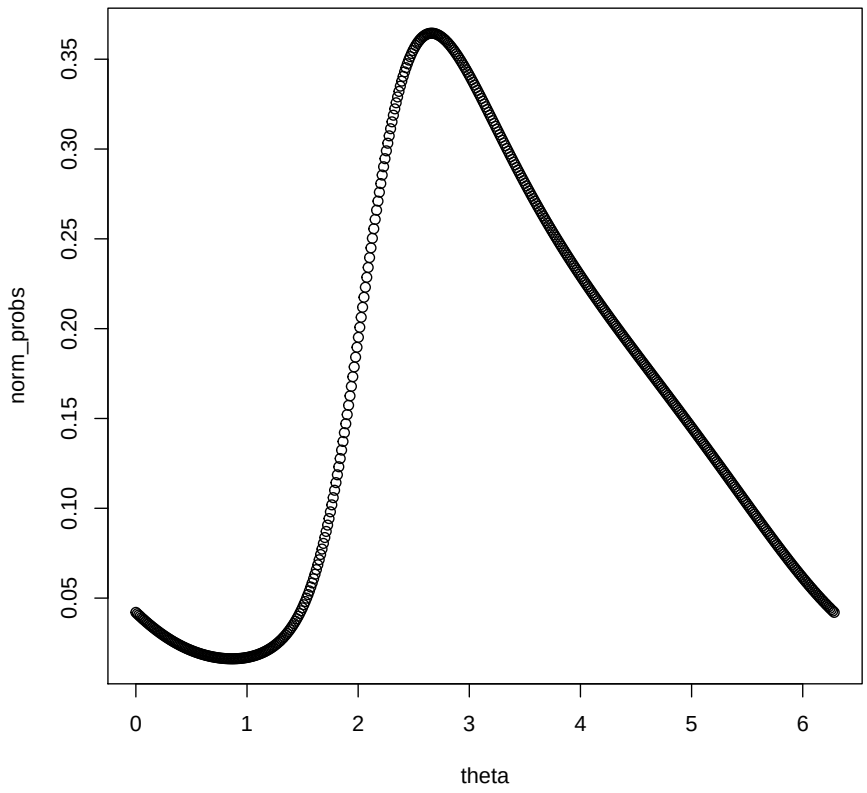
```
epsilons=c(0.118,0.39743,0.20,0.000)
sigmas=c(3.905,4.25,2.85,1.78)
distance=function(atom1,atom2){
  return(norm(atom1[3:5]-atom2[3:5],type="2"))
}
energy=function(atom1,atom2){
  r=distance(atom1,atom2)
  sigma = (sigmas[atom1[1]]+sigmas[atom2[1]])/2
  epsilon = sqrt(epsilons[atom1[1]]*epsilons[atom2[1]])
  return(4*epsilon*((sigma/r)^12-(sigma/r)^6))
}
coul_energy=function(atom1,atom2){
  r=distance(atom1,atom2)
  k=0.2
  return(332.0636*((atom1[2]*atom2[2])/r)*exp(-k*r))
}
```

```
#atoms = list(S,C1,C2,O,H,S_2,C1_2,C2_2,O_2,H_2)
atoms = list(S,C1,C2,O,S_2,C1_2,C2_2,O_2,H_2)
atom_combos = combn(atoms,2)
distances=mapply(distance,atom_combos[1,],atom_combos[2,])
indices=distances>=5
total_energy=function(atoms){
  atom_combos = combn(atoms,2)
  distances=mapply(distance,atom_combos[1,],atom_combos[2,])
  vdw_energies = mapply(energy,atom_combos[,indices][1,],atom_combos[,indices][2,])
  coul_energies = mapply(coul_energy,atom_combos[,indices][1,],atom_combos[,indices][2,])
  return(sum(vdw_energies)+sum(coul_energies))
}
apply_energy = function(Onew,Hnew){
  #print(Onew[1:3])
  #print(atoms[[4]][3:5])
  atoms[[4]][3:5] = c(Onew$x,Onew$y,Onew$z)
  #print(atoms[[4]][3:5])
  #atoms[[5]][3:5] = c(Hnew£x,Hnew£y,Hnew£z)
  atom_combos = combn(atoms,2)
  distances=mapply(distance,atom_combos[1,],atom_combos[2,])
  vdw_energies = mapply(energy,atom_combos[,indices][1,],atom_combos[,indices][2,])
  coul_energies = mapply(coul_energy,atom_combos[,indices][1,],atom_combos[,indices][2,])
  return(sum(vdw_energies)+sum(coul_energies))
}
#print(total_energy(atoms))
energies = sapply(seq(1,numpoints),function(i) apply_energy(Onew[i,],Hnew[i,]))
plot(theta,energies-energies[1])
```



Calculating the probabilities we get:

```
Temp = 298.15
kb = 0.0019872041
beta = 1./(kb*Temp)
dtheta = theta[2]-theta[1]
probs = exp(-beta*(energies-energies[1]))
norm_probs = probs/(sum(probs)*dtheta)
plot(theta,norm_probs)
```




```
  write.csv(norm_probs,file = "pair_pe_pdf.csv",row.names = FALSE,col.names = FALSE)

## Warning in write.csv(norm_probs, file = "pair_pe_pdf.csv", row.names
= FALSE, :  attempt to set 'col.names' ignored
```