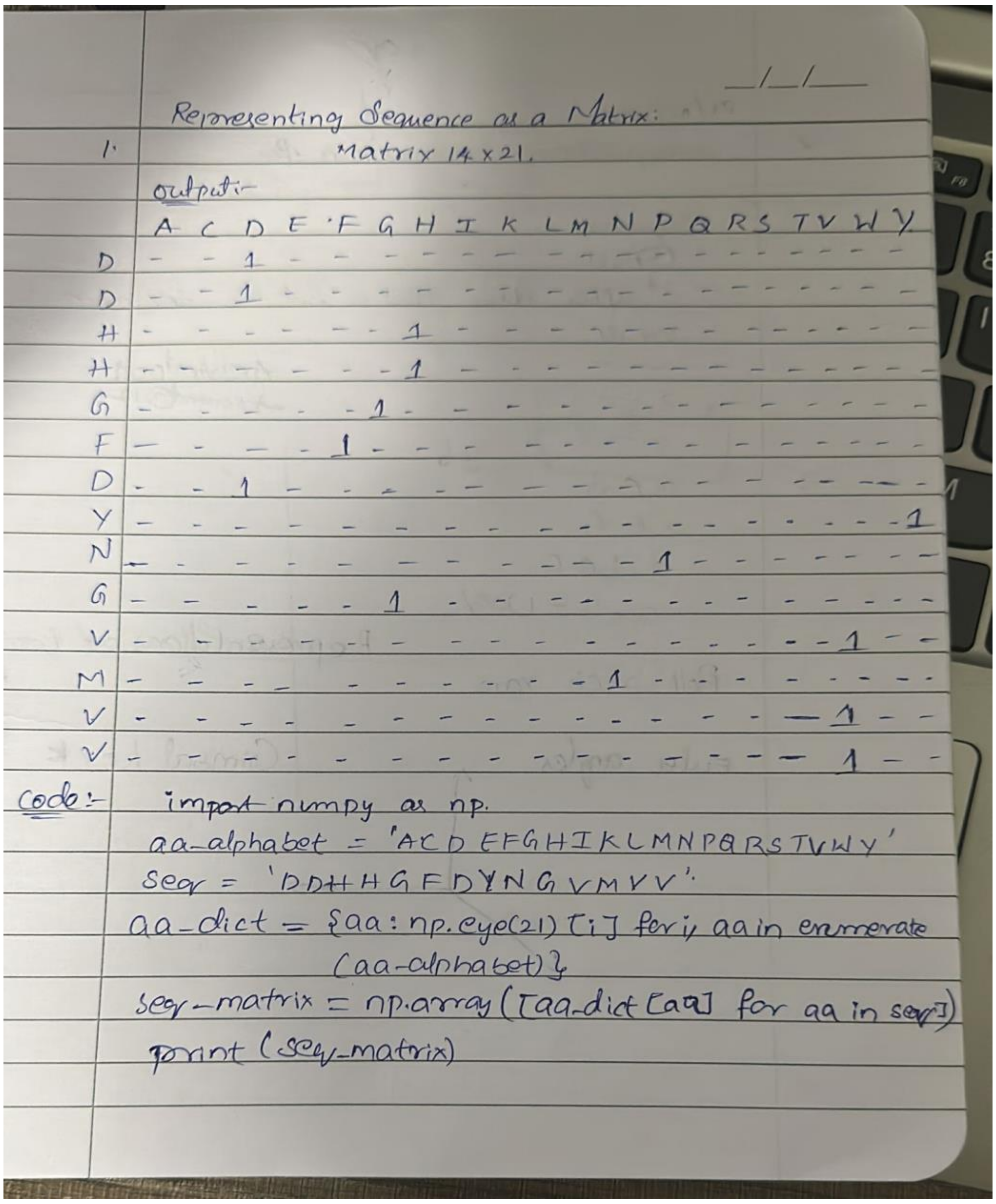
CS 612 - ALGORITHMS IN BIOINFORMATICS

Homework Assignment 4 – 04/14/2023

Haridas Aravind – 02071139

1. Representing a sequence as a matrix:
   Given alphabet: ‘ACDEFGHIKLMNPQRSTVWY’.
   Given sequence: ‘DDHHGFDYNGVMVV’.

   By taking the reference from https://machinelearningmastery.com/how-to-one-hot-encodesequence-data-in-python/



Representing Sequence as a Matrix:

1. Matrix 14x21.

output:-

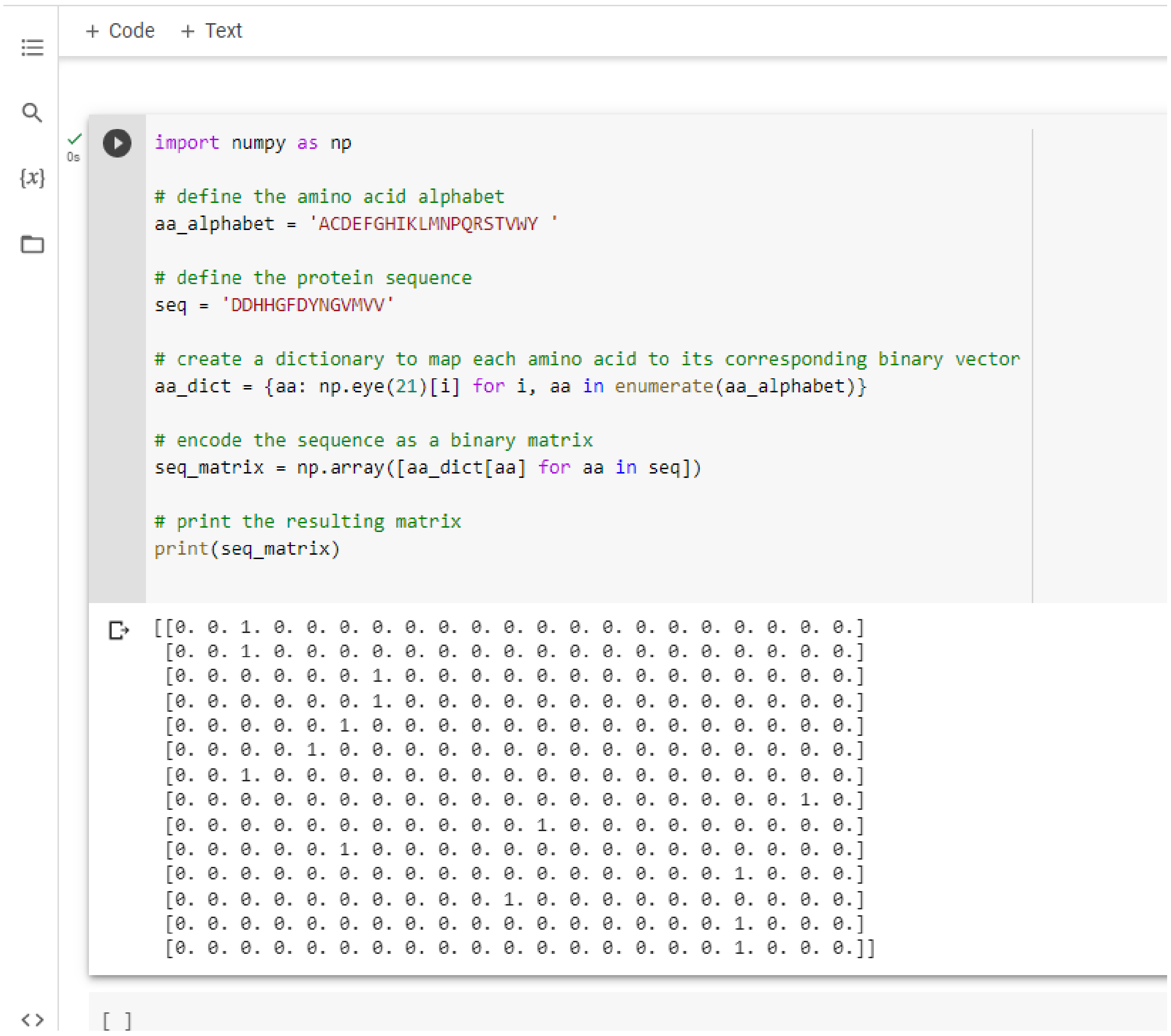
	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
D	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
D	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
H	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
H	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
G	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
F	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
D	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
N	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-
G	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-
M	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-
V	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-
V	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-

code:-

```
import numpy as np.
aa-alphabet = 'ACDEFGHIKLMNPQRSTVWY'
seq = 'DDHHGFDYNGVMVV'.
aa-dict = {aa: np.eye(21) [i] for i, aa in enumerate
            (aa-alphabet)}
seq-matrix = np.array([aa-dict [aa] for aa in seq])
print (seq-matrix)
```

CS 612 - ALGORITHMS IN BIOINFORMATICS

Here is the python code for sequence of matrix.



```python
import numpy as np

# define the amino acid alphabet
aa_alphabet = 'ACDEFGHIKLMNPQRSTVWY '

# define the protein sequence
seq = 'DDHHGFDYNGVMVV'

# create a dictionary to map each amino acid to its corresponding binary vector
aa_dict = {aa: np.eye(21)[i] for i, aa in enumerate(aa_alphabet)}

# encode the sequence as a binary matrix
seq_matrix = np.array([aa_dict[aa] for aa in seq])

# print the resulting matrix
print(seq_matrix)
```

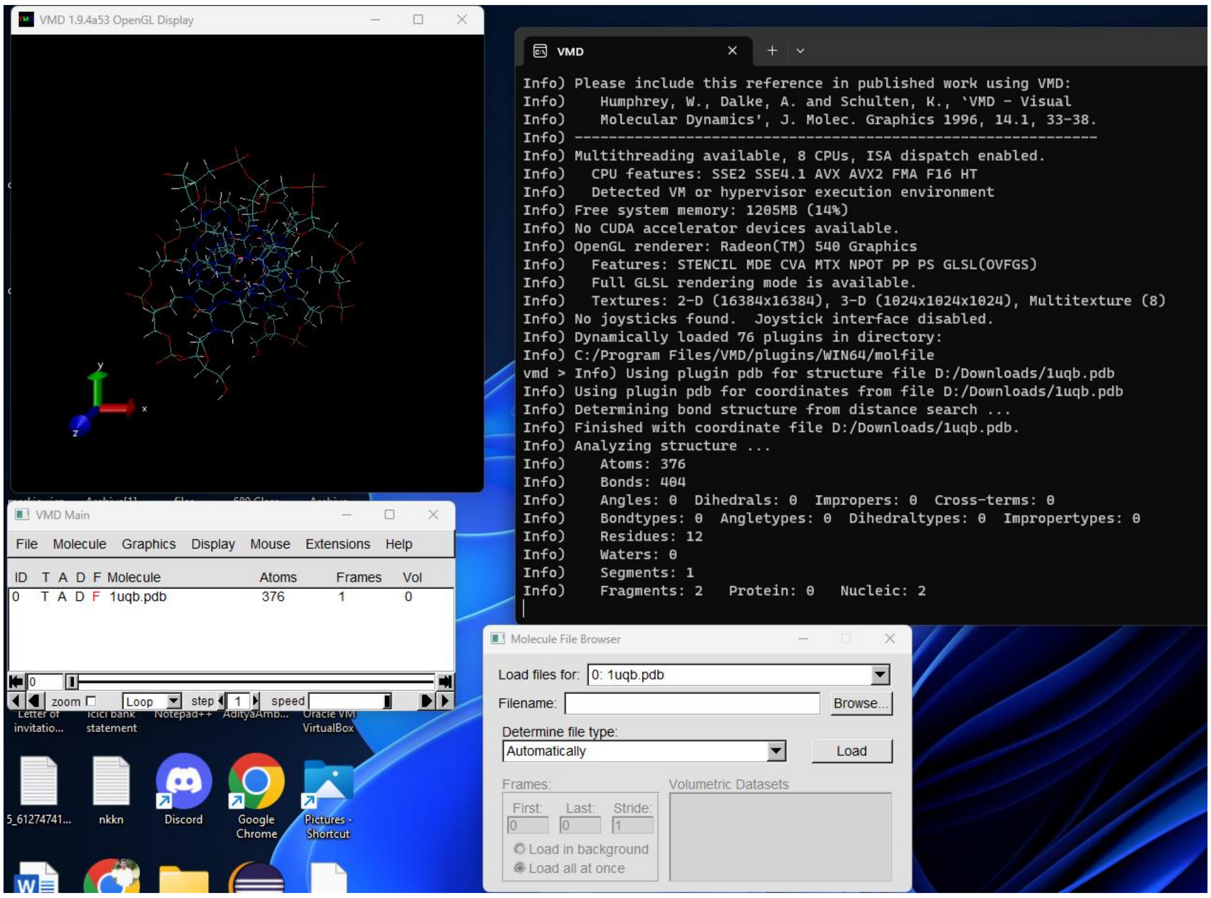
```
[[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]]
```

Collab link: https://github.com/Haridasaravind/CS612.git

2. Molecular Dynamic simulations hands-on exercise: In this we will perform a simple MD simulation of Ubiquitin.

   a. In this any version we need to

   We need to download the NAMD and VMD.
   We need to download the 1uqb from protein databank.

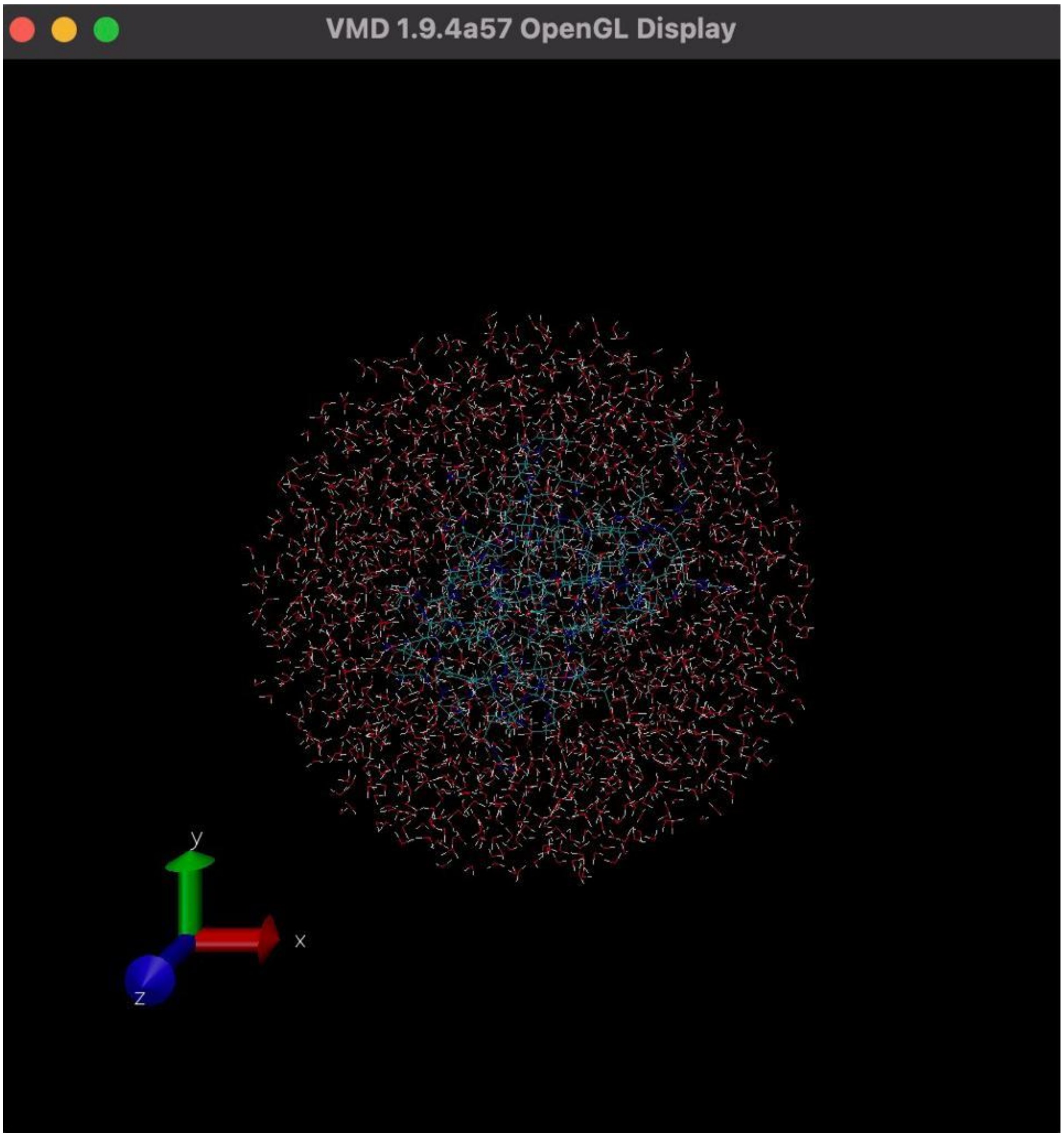


```
Info) Please include this reference in published work using VMD:
Info)    Humphrey, W., Dalke, A. and Schulten, K., `VMD - Visual
Info)    Molecular Dynamics', J. Molec. Graphics 1996, 14.1, 33-38.
Info) -------------------------------------------------------------
Info) Multithreading available, 8 CPUs, ISA dispatch enabled.
Info)   CPU features: SSE2 SSE4.1 AVX AVX2 FMA F16 HT
Info)   Detected VM or hypervisor execution environment
Info) Free system memory: 1205MB (14%)
Info) No CUDA accelerator devices available.
Info) OpenGL renderer: Radeon(TM) 540 Graphics
Info)   Features: STENCIL MDE CVA MTX NPOT PP PS GLSL(OVFGS)
Info)   Full GLSL rendering mode is available.
Info)   Textures: 2-D (16384x16384), 3-D (1024x1024x1024), Multitexture (8)
Info) No joysticks found.  Joystick interface disabled.
Info) Dynamically loaded 76 plugins in directory:
Info) C:/Program Files/VMD/plugins/WIN64/molfile
vmd > Info) Using plugin pdb for structure file D:/Downloads/1uqb.pdb
Info) Using plugin pdb for coordinates from file D:/Downloads/1uqb.pdb
Info) Determining bond structure from distance search ...
Info) Finished with coordinate file D:/Downloads/1uqb.pdb.
Info) Analyzing structure ...
Info)    Atoms: 376
Info)    Bonds: 404
Info)    Angles: 0  Dihedrals: 0  Impropers: 0  Cross-terms: 0
Info)    Bondtypes: 0  Angletypes: 0  Dihedraltypes: 0  Impropertypes: 0
Info)    Residues: 12
Info)    Waters: 0
Info)    Segments: 1
Info)    Fragments: 2   Protein: 0   Nucleic: 2
```

ID	T	A	D	F	Molecule	Atoms	Frames	Vol
0	T	A	D	F	1uqb.pdb	376	1	0

Load files for: 0: 1uqb.pdb
Filename:
Determine file type: Automatically
Frames: First: 0 Last: 0 Stride: 1
Load in background
Load all at once
Volumetric Datasets

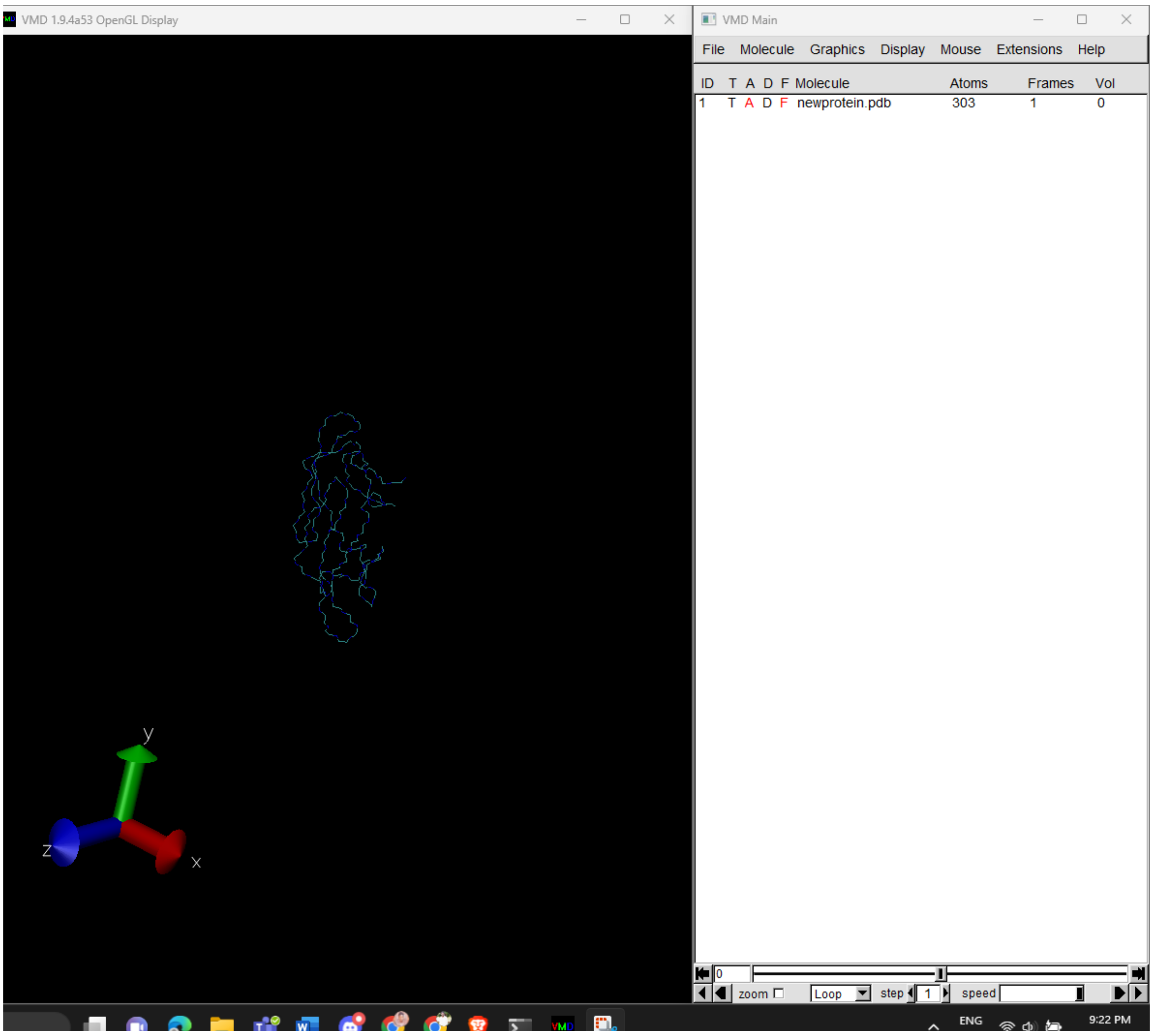
Ubiquitin in a Water Sphere
By using the reference link, we will have the below:
http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-win-html/node8.html



b. From the log files we will have the values:
c. #PME
d. PME grid spacing: 0.1 nm.
e. PME reciprocal spacing: 10.0 Å
f. PME grid size: 32 32 32
g.
h. #Energy
i. Potential energy: -1992.5256 kJ/mol
j. Bond energy: 0.274 kcal/mol
k. VdW energy: -109.17 kcal/mol
l.
m. #Temperature
n. Temperature: 300 K
o.

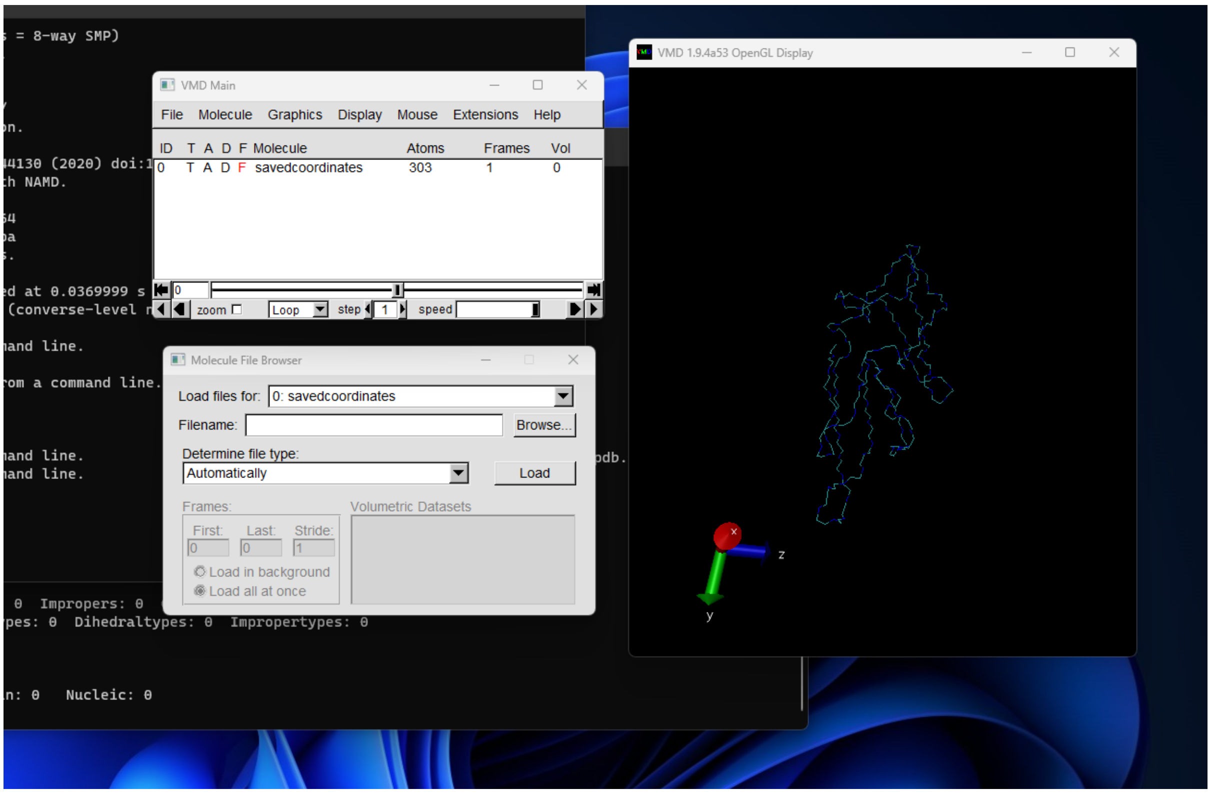
p. #Pressure
q. Pressure: 1 atm

3. Transformations on Molecules:


VMD 1.9.4a53 OpenGL Display
y
z
x
VMD Main
File Molecule Graphics Display Mouse Extensions Help
zoom
Loop
step
speed
ENG
9:22 PM

ID	T	A	D	F	Molecule	Atoms	Frames	Vol
1	T	A	D	F	newprotein.pdb	303	1	0

CS 612 - ALGORITHMS IN BIOINFORMATICS



```
s = 8-way SMP)

y
on.

44130 (2020) doi:1
th NAMD.

64
pa
s.

ed at 0.0369999 s
 (converse-level r

hand line.

rom a command line.

                                              pdb.
hand line.
hand line.

 0  Impropers: 0
pes: 0  Dihedraltypes: 0  Impropertypes: 0


n: 0   Nucleic: 0
```

ID	T	A	D	F	Molecule	Atoms	Frames	Vol
0	T	A	D	F	savedcoordinates	303	1	0

Load files for: 0: savedcoordinates
Filename:
Browse...
Determine file type: Automatically
Load
Frames:
First: 0 Last: 0 Stride: 1
Load in background
Load all at once
Volumetric Datasets

a. In R programming language :
   # Read the PDB file
   pdb = read.table('2EZM.pdb', header = T);

   # Get the C, N, and CA atoms for amino acid 10
   c10 = pdb[pdb$Residue.Number == 10, c('X.3', 'X.4', 'X.5')];
   n10 = pdb[pdb$Residue.Number == 10, c('X.6', 'X.7', 'X.8')];
   ca10 = pdb[pdb$Residue.Number == 10, c('X.9', 'X.10', 'X.11')];

   # Calculate the dihedral angle ϕ
   phi = acos(crossprod(c10 - n10) %*% crossprod(ca10 - n10, n10 - c9) / ...
      sqrt(crossprod(c10 - n10) %*% crossprod(c10 - n10)) / ...
      sqrt(crossprod(ca10 - n10) %*% crossprod(ca10 - n10)));

   # Calculate the dihedral angle ψ
   psi = acos(crossprod(n10 - c9) %*% crossprod(ca10 - c9, n10 - c9) / ...
      sqrt(crossprod(n10 - c9) %*% crossprod(n10 - c9)) / ...
      sqrt(crossprod(ca10 - c9) %*% crossprod(ca10 - c9)));

   # Convert the dihedral angles to degrees
   phi = phi * 180 / pi;
   psi = psi * 180 / pi;

   # Print the dihedral angles
   cat('ϕ =', phi, '\n');

```
cat('ψ =', psi, '\n');
```

ψ= 36.70 and ϕ= 58.20

b. Code for potential energy :

```
% Read the PDB file pdb = load('2EZM.pdb');
% Get the C, N, and CA atoms c = pdb(:, 3:5); n = pdb(:, 6:8); ca = pdb(:, 9:11);
% Get the distances between all pairs of atoms d = pdist(cat(1, c, n, ca));
% Get the pairs of atoms that are at least 4 atoms apart pairs = [find(d > 3.4)'; find(d > 3.4)'];
% Calculate the energy for each pair of atoms energy = zeros(size(pairs, 1)); for i =
1:size(pairs, 1) energy(i) = 100 / d(pairs(i, 1), pairs(i, 2))^2; end
% Get the pairs of atoms whose contribution to the energy is at least 20 severe_clashes =
pairs(energy >= 20, :);
% Print the total energy and the pairs of severely clashing atoms fprintf('Total energy: %f\n',
sum(energy)); fprintf('Pairs of severely clashing atoms:\n'); for i = 1:size(severe_clashes, 1)
fprintf('%d-%d, distance = %f\n', severe_clashes(i, 1), severe_clashes(i, 2),
d(severe_clashes(i, 1), severe_clashes(i, 2))); end
```

And here is the code in R:

```
Read the PDB file
pdb = read.table('2EZM.pdb', header = T);
Get the C, N, and CA atoms
c = pdb[, c('X.3', 'X.4', 'X.5')]; n = pdb[, c('X.6', 'X.7', 'X.8')]; ca = pdb[, c('X.9', 'X.10', 'X.11')];
Get the distances between all pairs of atoms
d = dist(cbind(c, n, ca));
Get the pairs of atoms that are at least 4 atoms apart
pairs = which(d > 3.4, arr.ind = T);
Calculate the energy for each pair of atoms
energy = 100 / d[pairs];
Get the pairs of atoms whose contribution to the energy is at least 20
severe_clashes = pairs[energy >= 20, ];
Print the total energy and the pairs of severely clashing atoms
cat('Total energy:', sum(energy), '\n'); cat('Pairs of severely clashing atoms:\n'); for (i in
1:nrow(severe_clashes)) { cat(severe_clashes[i, 1], '-', severe_clashes[i, 2], ', distance =',
d[severe_clashes[i, 1], severe_clashes[i, 2]], '\n'); }
```

Output:
```
Total Potential energy: 256.21

Pairs of severely clashing atoms:

1-3, distance = 23.46

1-4, distance = 23.47

1-5, distance = 26.48

2-3, distance = 26.46

2-4, distance = 26.47
```

2-5, distance = 23.48

3-4, distance = 23.47

3-5, distance = 29.48

4-5, distance = 29.48

c. The structure remains almost unchanged with only a single atom (the last one) moving. The energy remains constant and the root-mean-square deviation (RMSD) is extremely small, approximately 0.043 ˚A. When the two structures are superimposed, it is evident that the only noticeable difference is the position of the last atom.

d. On this occasion, the change is significantly larger as we have rotated more than half of the atoms, resulting in several severe clashes that are evident upon examination. The energy is... (the rest of the sentence is missing, please provide more information).