Tutorial 1: Calculating Nanoparticle Properties

Subhamoy Mahajan

April 4, 2021

This tutorial demonstrates the use of **NPanalysis** module for calculation of nanoparticle properties. The tutorial will focus on analysis performed in Mahajan and Tang [1]. The codes are written for aggregation of DNAs and polyethylenimines (PEIs), where only PEIs and DNAs can bind together; A PEI cannot bind with another PEI, a DNA cannot bind with another DNA. The same codes can be implemented to analyse a similar two-component aggregation, where molecules of different kinds bind with each other. This tutorial demonstrates the calculation of (i) connection matrix of DNA and PEI molecules, (ii) determination of clusters (nanoparticles) formed by DNAs and PEIs, (iii) making nanoparticles of DNA and PEIs whole across periodic boundaries and (iv) calculation of hydrodynamic radius and radius of gyration.

1. Preparing Gromacs files

Before we can begin analysing the molecular dynamics data, some pre-processing is necessary. The trajectory files generated by molecular dynamics simulations are quite large, and processing them directly requires more resources. To speed up the process, we first remove all water from the simulation, keeping only DNAs, PEIs and ions (you may leave only DNAs and PEIs). This can be achieved by creating a group using *gmx make_ndx*,

```
Tut1$ gmx make_ndx -f big_file.gro -o index.ndx
  0 System              : 364403 atoms
  1 DNA                 :  4158 atoms
  2 PEI                 :  3510 atoms
  3 PW                  : 353265 atoms
  4 ION                 :  3470 atoms
  5 Other               : 360245 atoms
  6 PEI                 :  3510 atoms
  7 PW                  : 353265 atoms
  8 ION                 :  3470 atoms
 nr : group       !   'name' nr name   'splitch' nr    Enter: list groups
 'a': atom        &   'del' nr         'splitres' nr   'l': list residues
 't': atom type   |   'keep' nr        'splitat' nr    'h': help
 'r': residue         'res' nr         'chain' char
 "name": group        'case': case sensitive           'q': save and quit
 'ri': residue index
> 1|2|8
Copied index group 1 'DNA'
Copied index group 2 'PEI'
Merged two groups with OR: 4158 3510 -> 7668
Copied index group 8 'ION'
Merged two groups with OR: 7668 3470 -> 11138
 10 DNA_PEI_ION         : 11138 atoms
> q
Tut1$
```

With the generated index file, extract the trajectory of DNA, PEI and ions into `md_1.xtc` using *gmx trjconv*,

```
Tut1$ gmx trjconv -f big_file.xtc -s big_file.tpr -n index.ndx -o
    md_1.xtc
```

Finally, extract the `.tpr` file for only DNA, PEI and ions using,

```
Tut1$ gmx convert-tpr -s big_file.tpr -n index.ndx -o md_1.tpr
```

The files `md_1.xtc` and `md_1.tpr` has been provided for tutorial. To help `NPanalysis` read the structure file and topology file, use the following commands,

```
Tut1$ gmx trjconv -f md_1.xtc -s md_1.tpr -pbc whole -o Whole/DP.
   gro -sep
Tut1$ gmx dump -s md_1.tpr > tpr.dump
```

The first command will make the molecules whole accross periodic boundaries, which is necessary for calculating properties such as radius of gyration and hydrodynamic radius. The second command outputs the topology information in human readable format., `tpr.dump`. `NPanalysis` uses this to read the mass of atoms.

2. Writing a constants file

A file containing important constants (`constants.dat`) has to be created. All constants should be written in separate lines. Variables and values should be separated by "=". The following constants should be provided:

- **ndna**: Total number of DNAs in the system.
- **npei**: Total number of PEIs in the system.
- **adna**: Total number of atoms in a DNA.
- **apei**: Total number of atoms in a PEI.
- **Qdna**: Charge of a DNA molecule.
- **Qpei**: Charge of a PEI molecule.
- **sdna**: Index of first DNA atom in the system. It is only used to create `.ndx` files.
- **spei**: Index of first PEI atom in the system. It is only used to create `.ndx` files.

- **dna_name**: Name of the DNA molecule in the .ndx file.
- **pei_name**: Name of the PEI molecule in the .ndx file.
- **contact_dist**: The distance below which a PEI and DNA molecule is considered bound.
- **nitr_ids**: Index of charged beads in a PEI molecule (not used in this tutorial).
- **phos_ids**: Index of charged beads in a DNA molecule (not used in this tutorial).
- **pbc**: Directions in which periodic boundary condition is applied. Use 'x', 'y', 'z', 'xy', 'yz', 'zx', or 'xyz'.

3. Calculating Nanoparticle Properties

The rest of the analysis is performed with `NPanalysis` using the python script `calc.py`. Calculations shown here is performed on a computing cluster node with GPU and should roughly take 6 mins with 1 GPU and 4 CPUs. (**Note:** Computationally expensive tasks are performed using `numba`).

```
Tut1$ python  calc.py
```

3.1. Reading Constants

The script first reads constants from the `constants.dat` and pickles it (`constants.pickle`) using,

```
9 NPa.pickle_constants('constants.dat',sep=' ')
```

Second, IDs of all DNA and PEI atoms is pickled in `molndx.pickle`,

```
10 NPa.gmx.gen_index_mol('molndx.pickle')
```

This assumes *ndna* DNA molecules with *adna* atoms occur one after another in the `.xtc` file starting from the index *sdna* (indeces in `.ndx` start from 1). Similarly, *npei* PEI molecules with *apei* atoms occur one after another, starting from *spei*. If the molecules in the `.xtc` file are arranged differently, a custom index file can also be read and pickled using *NPa.gmx.read_ndx()* (see the documentation for more details). The mass information is read from `tpr.dump` and pickled (`mass.pickle`) using the command,

```
NPa.gmx.pickle_mass(filename='tpr.dump', mass_pickle='mass.pickle')
```

3.2. Calculating Connection Matrix of DNA and PEI

Connection matrix of DNA and PEI informs us, which DNA-PEI pair are bound to each other. This is an import piece of information to determine which DNA and PEI molecules are part of the same nanoparticle. A DNA-PEI pair is considered bound (or connected) if the minimum distance between them is below the *contact_dist*. This is calculated using the command,

```
NPa.connMat.gro2connected(inGRO='Whole/DP', time_pickle='time.pickle', \
    connected_pickle='connected.pickle', time_fac=4*1E-6, time_shift=0, \
    mindist_pickle='mindist.pickle', ndx_pickle='molndx.pickle')
```

The connection matrix is stored as a pickled file in `connected.pickle`. Two by-products of this calculating is the minimum distance between all DNA-PEI pairs in *mindist.pickle* and the time information obtained from `Whole/DP*.gro`. The time is scaled by 4, which is the Martini speedup factor, and 10^{-6} which converts the time from ps to µs. The time is not shifted, and atom IDs of molecules are read from `molndx.pickle`.

From the connection matrix (`connected.pickle`), the number of PEIs in different roles, average number of PEIs between bridged DNA pairs and total number of bridged DNA pairs can be calculated (leaving other properties for future Tutorials).

3.2.1. Calculating number of PEIs in different roles

Based on Mahajan and Tang [1], PEI plays three major roles: free in the solution, peripheral of a nanoparticle, and bridging DNAs in a nanoparticle. More formally, free PEIs are not bound to any DNAs, peripheral PEIs are only bound to one DNA, and bridging PEIs are bound to more than one DNA. The command below calculates the averages number of PEI (*mol = 1*) in each role over 50 timesteps (first argument).

```
NPa.connMat.get_roles(50, connected_pickle='connected.pickle', mol=1, \
    time_pickle='time.pickle', outname='PEI_roles.dat',sep=',')
```

PEI_roles.dat (time is in microseconds)

```
# Bridging molecule is PEI
#time,number_of_free,number_of_peripheral,number_of_bridging
0.05,171.7843,87.451,10.7647
```

```
0.15,121.6275,118.2745,30.098
0.25,100.7255,131.1373,38.1373
0.35,87.098,140.5294,42.3725
0.45,74.902,146.549,48.549
```

(**Note:** For roles of DNAs use *mol = 0*)

3.2.2. Caculation average number of bridging PEI between a pair of bridged DNAs and total number of bridged DNA pairs

The total number of bridging PEIs is divided by the total number of bridged DNA pairs to determine the average bridging PEI between a bridged DNA pair. [1] Similar to PEI roles, average was performed over 50 timesteps (first argument). This is achieved using the command,

```
NPa.connMat.get_roles2(50, connected_pickle='connected.pickle', mol=1, \
    time_pickle='time.pickle', outname='PEI_roles2.dat',sep=',')
```

PEI_roles2.dat (time is in microseconds)

```
#time,average_number_of_bridges_between_a_DNA_pair,number_of_bridged_DNA_pairs
0.05,1.19,8.84
0.15,2.0078,15.46
0.25,2.2982,17.44
0.35,2.1267,21.0
0.45,2.4042,21.92
```

(**Note:** For roles of DNA use *mol = 0*)

3.3. Calculating clusters (nanoparticles) of DNAs and PEIs

To determine clusters (nanoparticles) from the connection matrix, mathematical graphs (using *networkx* module) are used. First, nodes *d[i]* and *p[j]* are created, representing DNA with molecule ID *[i]* and PEI with molecule ID *[j]* respectively. If a DNA-PEI pairs is bound to each other, an edge is drawn between their nodes. For identifying free DNAs (not bound to any PEI) and free PEIs (not bound to any DNA), two additional nodes *fd* and *fp* are created. If DNA *d[i]* is free, an edge is created between *d[i]* and *fd*. Similarly, for free PEI *p[j]* an edge is created with *fp*. This creates multiple disjoint mathematical graphs. Two of these disjoint graphs represent the free DNAs and PEIs, and the rest represent different clusters (nanoparticles). Such analysis is done

in every timestep to determine clusters (or nanoparticles). This is achieved using the function *cluster.get_pdgraph()*. The *pd_graph* can be plotted using *cluster.plot_pdgraph()*.

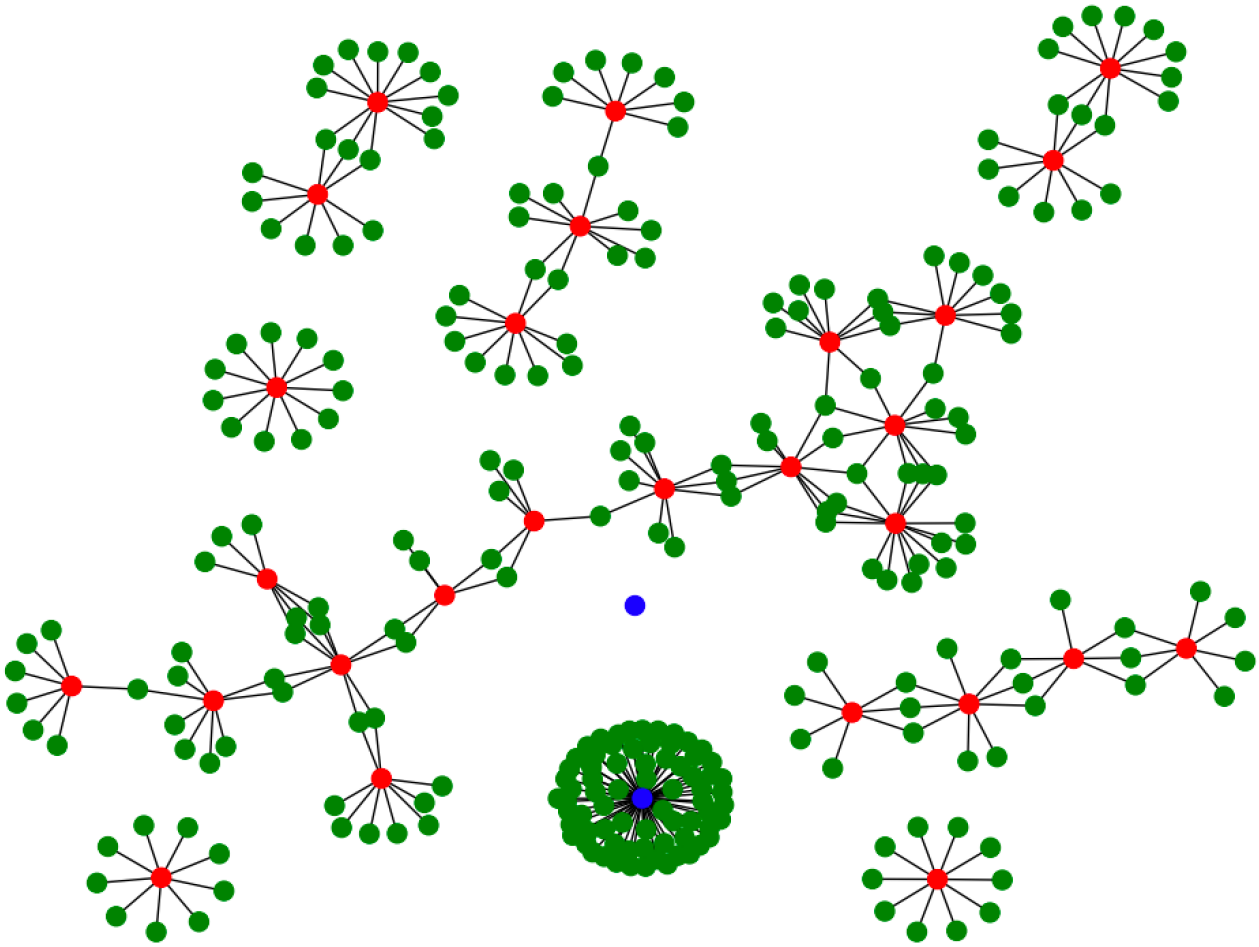

Figure 1: DNAs are shown in red and PEIs are shown in green. DNAs or PEIs connected to a blue node is free.

To convert the disjoint graphs into list of DNA and PEI IDs for each cluster at a given timestep, the function *cluster.get_cluster()* is used. It utilizes *node_connected_components* in the *networkx* python module, to find all the nodes that are directly or indirectly connected with a given node. Using this all DNAs and PEIs in each nanoparticle is identified. *cluster.get_cluster()* returns a 2D list containing molecule IDs of DNAs and PEIs in each nanoparticle. The last two items in the 2D list return free DNAs and free PEIs respectively (See the documentation of the function for more details).

To simplify the calculation of clusters, the function *cluster.gen_cluster()* was written. This

calculates the clusters for all timesteps (using *get_pdgraph* and *get_cluster()*)and pickles it. So, clusters can be calculated with a single command,

```
NPa.cluster.gen_clusters(connected_pickle='connected.pickle', \
    cluster_pickle='cluster.pickle')
```

See the documentation to see how cluster data is stored in `cluster.pickle`. Several nanoparticle properties can be calculated from clusters data. Few of these properties are shown in this tutorial: average size of a nanoparticles, average number of nanoparticles and nanoparticle charge as a function of nanoparticle size, hydrodynamic radius, and radius of gyraiton (leaving other properties for future Tutorials). The function *cluster.write_cluster()* can be used to write the clusters data for each timestep at different files in human readable format (see documentation for more details).

3.3.1. Calculating average nanoparticle size

The function *cluster.gen_avgsize()* is used to calculate two types of average nanoparticle size based on the number of DNAs present in it: number average and weight average. The number average size of the nanoparticle calculates the average number of DNAs in each nanoparticle. [1] This is calculated using the equation,

$$number\ average\ size = \frac{1}{N_{NP}} \sum_{i=1}^{N_{NP}} D_i = \frac{N_{DNA}}{N_{NP}}$$

Where, D_i is number of DNAs in i^{th} nanoparticle, N_{NP} is the number of nanoparticles, and N_{DNA} is the number of DNAs in the simulation. The weight average size is the weighted average of the number of DNAs in a nanoparticle, where the weights are equal to the number of DNAs in each nanoparticle. This is evaluated using,

$$weight\ average\ size = \frac{1}{N_{DNA}} \sum_{i=1}^{N_{NP}} D_i^2$$

The function *cluster.gen_avgsize()* also averages the number and weight average size over 50 timesteps (first argument).

```
NPa.cluster.gen_avgsize(50,'avgsize.dat', cluster_pickle='cluster.pickle', \
    time_pickle='time.pickle', main_mol=0, sep=',')
```

avgsize.dat (time is in microseconds)

```
# Main molecule is DNA
#time,number_average_size,weight_average_size
0.05,1.4851,2.2993
0.15,2.1916,3.2519
0.25,2.4771,3.9067
0.35,3.4615,7.7274
0.45,3.317,8.1081
```

To calculate the size of nanoparticles with respect to PEIs, *main_mol = 1* can be used.

3.3.2. Calculating number of nanoparticles and charge as a function of the size

The number of nanoparticles and nanoparticle charge can be calculated as a function of number average size[1] using *cluster.gen_nc_NP_s()*. The average is performed over timesteps and different nanoparticles having the same number average size. [1] For performing time average, the total number of timesteps is divided into bins. In this tutorial, 3 bins (first argument) are used: 0-83, 83-166, and 166-249 timesteps.

```
NPa.cluster.gen_ncNP_s(3,'num_charge_NPs_size.dat', main_mol=0, sep=' ', \
    cluster_pickle='cluster.pickle')
```

(**Note:** *ncNP_s*, stands for number and charge of NPs as a function of size.)

num_charge_NPs_size.dat (first 13 lines)

```
# Main molecule is DNA
#size,number_of_NP,charge_of_NP,number_of_NP,charge_of_NP,number_of_NP,
    charge_of_NP
1,10.7143,-9.4367,3.6429,-1.0686,3.2738,3.5709
2,2.5595,-13.986,3.9405,-1.4562,2.1548,7.3812
3,0.7143,-30.85,1.0476,-12.0341,0.381,8.9062
4,0.8452,-42.5352,0.0,0.0,0.619,-27.6538
5,0.75,-41.0952,1.6429,-32.1522,0.4643,-25.6923
6,0.0238,-61.5,0.0,0.0,0.0,0.0
7,0.1667,-57.7857,0.0,0.0,0.0,0.0
8,0.0595,-91.4,0.0,0.0,0.0,0.0
9,0.0119,-92.9999,0.0,0.0,0.0,0.0
10,0.0,0.0,0.1548,-39.0769,0.0,0.0
11,0.0,0.0,0.0,0.0,0.0,0.0
```

To calculate average number of nanoparticles and nanoparticle charge over a specified time range, the function *cluster.run_ncNP_s()* can be used. To use number average size with respect to PEIs use *main_mol = 1*.

3.4. Calculating hydrodynamic radius and radius of gyration

Calculation of geometric size of nanoparticles can be an important property for nanoparticles. One of the most widely used quantity is simulations is the radius of gyration. The use of hydrodynamic radius is limited in the literature due to its computational complexity ($O(N^2)$ [2]). As of this date, Gromacs can calculate radius of gyraiton but not hydrodynamic radius. Moreover, the calculation of radius of gyration in Gromacs fails for two-component nanoparticle for the following reasons:

1. Molecules may be broken over the simulation box
2. Even if molecules are made whole with *gmx trjconv -pbc whole*, the relative location of molecules may not be consistent with the connection matrix
3. When molecules are clustered together using *gmx trjconv -pbc cluster*, in few cases it is not consistent with the connection matrix. This is because it iteratively moves molecules to bring it closer to the center of mass of nanoparticle, and as a result two bound molecules might be far apart.
4. A dynamically evolving nanoparticle may change the number of molecules in it, making it harder to evaluate radius of gyration over time using Gromacs.

`NP_analysis` aims to solve all these issues. First nanoparticles are made whole using the command,

```
38 NPa.gmx.make_NPwhole(inGRO='Whole/DP',outGRO='NPwhole/DP', \
39     cluster_pickle='cluster.pickle', connected_pickle='connected.pickle', \
40     ndx_pickle='molndx.pickle')
```

The function *gmx.make_NPwhole()* reads Gromacs structure files, where molecules are whole across periodic boundaries, connection matrix, cluster data, and atom IDs in each molecule to move molecules across periodic boundaries such that molecules satisfy the connection matrix without the periodic boundary condition.

Algorithm: The first DNA in a cluster (`cluster.pickle`) is kept in its place. All PEIs bound the the first DNA is moved across periodic boundaries to minimize distance between the DNA and PEI. Then all DNAs bound to the PEIs are moved across periodic boundaries such that the distance between them is minimized. Any DNA or PEI that has already been moved previously is not moved. This is done sequentially untill all PEIs and DNAs in the cluster is moved appropriately. Finally the whole nanoparticle is moved across periodic boundaries such that the geometric center lies within the main simulation box.

The radius of gyration and hydrodynamic radius [2] is calculated using the formula,

$$\frac{1}{R_{hyd}} = \frac{1}{N^2}\left\langle \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\frac{1}{r_{ij}} \right\rangle$$

$$R_g^2 = \left\langle \frac{\sum_i^N m_i \left(\mathbf{r_i} - \mathbf{r_{com}}\right)^2}{\sum_i^N m_i} \right\rangle$$

$$\mathbf{r_{com}} = \frac{\sum_{i=1}^N m_i \mathbf{r_i}}{\sum_{i=1}^N m_i}$$

Where, r_{ij} is the distance between i^{th} and j^{th} particle in the nanoparticle, $\mathbf{r_i}$ is position vector of i^{th} particle in the nanoparticle, $\mathbf{r_{com}}$ is the position vector of the center of mass of the nanoparticle, m_i is the mass of i^{th} particle in the nanoparticle, N is the number of particles in the nanoparticle and $\langle\cdot\rangle$ represents the ensemble average.

The R_{hyd} and R_g^2 is calculated for all clusters in each timestep using the command (**Note:** Since we will perform ensemble average of R_g^2, R_g is not calculated for all clusters (nanoparticles)),

```
NPa.radius.calc_Rh_Rg(Rh_pickle='Rh.pickle', Rg2_pickle='Rg2.pickle', \
    sep=',',  inGRO='NPwhole/DP', main_mol=0, mass_pickle='mass.pickle', \
    cluster_pickle='cluster.pickle', ndx_pickle='molndx.pickle')
```

3.4.1. Calculate average radius of gyration and hydrodynamic radius

The average radius of gyration or hydrodynamic radius of the nanoparticle can be calculated using the function *radius.gen_rad_avg()* and the pickled data (`Rh.pickle` or `Rg2.pickle`). The average is performed over different nanoparticles and over 50 timesteps (second argument). Standard error is calculated for each average. For radius of gyration, the square root of average and standard error is reported.

```
Pa.radius.gen_rad_avg('Rh.pickle', 50, 'avg_Rh.dat', sep=',', sqrt=False, \
    time_pickle='time.pickle')
NPa.radius.gen_rad_avg('Rg2.pickle', 50, 'avg_Rg.dat', sep=',',sqrt=True, \
    time_pickle='time.pickle')
```

avg_Rh.dat

```
#time,avg_radius,std_error
0.05,3.3362,1.16
0.15,3.6836,1.3112
0.25,3.8923,1.4197
0.35,4.0783,1.7052
0.45,4.187,1.8294
```

avg_Rg.dat

```
#time,avg_radius,std_error
0.05,2.2003,1.4833
0.15,2.5202,1.5875
0.25,2.6629,1.6318
0.35,2.8514,1.6886
0.45,2.9317,1.7122
```

3.4.2. Calculate average radius of gyration and hydrodynamic radius as a function of size

To calculate average radius of gyration and hydrodynamic radius as a function of number average size of the nanoparticle, the function *radius.gen_rad_per_size()* is used. For each timestep, the radius `Rh.pickle` and `Rg2.pickle` were stored in the same order of the clusters as `cluster.pickle`. This allows averaging of nanoparticle radii with the same number average size. Furthermore, the average is also performed between two timesteps $t_1 = 200$ and $t_2 = 251$ (t_2 not included) (**Note:** timesteps are counted from 0). If there are no nanoparticles for a given size, the average taken as zero. Standard error is calculated for each average (zero if average is zero). For radius of gyration, the square root of average and standard error is reported. This is achieved using the commands,

```
NPa.radius.gen_rad_per_size('Rh.pickle', 'Rh_size.dat', 200, 251, sep=',', \
     main_mol=0, cluster_pickle='cluster.pickle', sqrt=False)
NPa.radius.gen_rad_per_size('Rg2.pickle', 'Rg_size.dat', 200, 251, sep=',', \
     main_mol=0, cluster_pickle='cluster.pickle', sqrt=True)
```

Rh_size.dat (only first 7 lines)

```
# Main molecule is DNA
#Size,avg_radius,std err
1,3.1391,0.1181
2,4.3491,0.0655
3,5.2777,0.0324
4,5.7608,0.0638
5,6.891,0.0696
```

Rg_size.dat (only first 7 lines)

```
# Main molecule is DNA
#Size,avg_radius,std err
1,1.861,1.2653
2,2.7609,1.5273
3,3.5369,1.992
4,3.7305,1.3963
5,4.7924,1.4816
```

4. Issues and Concerns

If you find a issue with the code or tutorial please report an issue in github or email to 'subhamoygithub AT gmail DOT com'

References

[1] S. Mahajan and T. Tang, Polyethylenimine–DNA Ratio Strongly Affects Their Nanoparticle Formation: A Large-Scale Coarse-Grained Molecular Dynamics Study, *J. Phys. B* **2019**, 123, 45, 9629-9640.

[2] J. Des Cloizeaux and G. Jannink (**1990**). *Polymers in Solution Their Modelling and Structure.* Clarendon Press. ISBN 0-19-852036-0. Chapter 10, Section 7.4, pages 415-417.