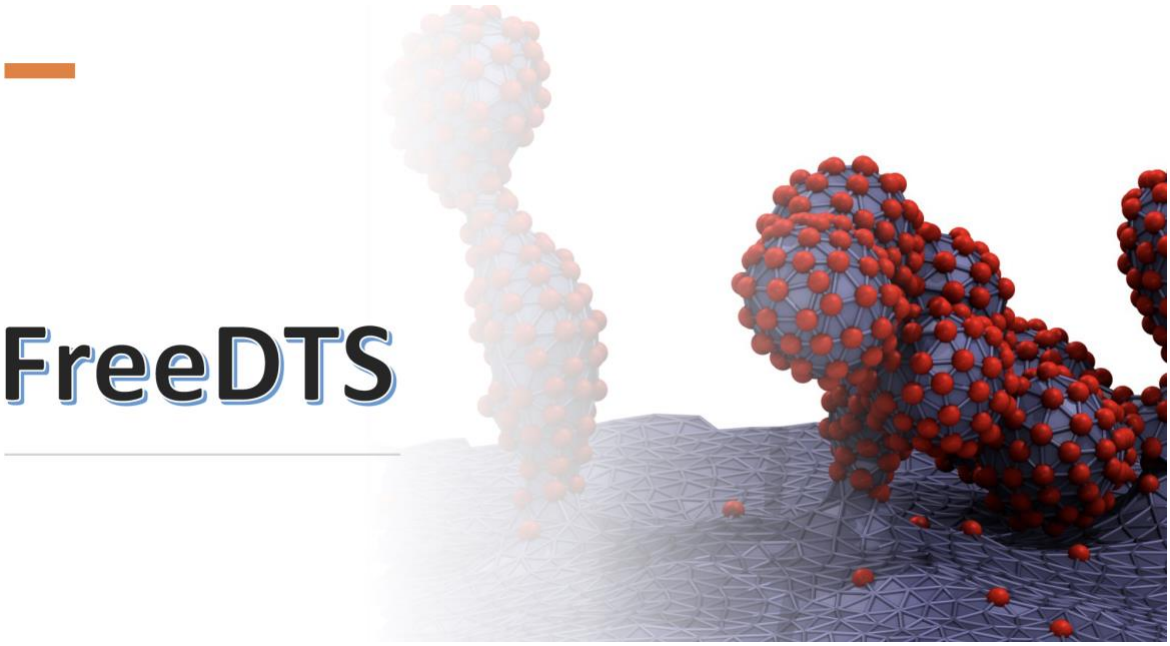
User Manual FreeDTS



About FreeDTS

FreeDTS is a software for performing computational research on biomembranes at the mesoscale. In FreeDTS, a membrane is represented by a dynamically triangulated surface equipped with vertex-based inclusions to integrate the effects of integral and peripheral membrane proteins.

Main developer

Weria Pezeshkian
Niels Bohr International Academy,
Niels Bohr Institute, University of Copenhagen,
Blegdamsvej 17, 2100 Copenhagen, DENMARK

Citations

Weria Pezeshkian, John H. Ipsen
Mesoscale simulation of biomembranes with FreeDTS
doi: https://doi.org/10.1101/2023.05.05.539540

W. Pezeshkian, J.H. Ipsen
Fluctuations and conformational stability of a membrane patch with curvature inducing inclusions
Soft matter **15**, 9974-9981 (2019)

Table of Contents

Compiling the source code

To compile the OPENDTS source code just execute “compile.sh” script.

```
./compile.sh
```

This will create three binary files: **DTS**, **CNV**, and **GEN**. The **DTS** is for running simulations; **GEN** is for generating triangulated files (TS) and **CNV** is for converting different file formats.

How to run a simulation using FreeDTS

To run a simulation using **OPENDTS**, we need to feed at least two files to the **DTS** binary. An input file (with *dts* extension) that includes all the simulation inputs (see the “Input File” section) and a topology file that contains information about the topology of the TS mesh. The topology file could be in *top* or *tsi* format (see the Topology File sections).

With these two files a simulation can be started as.

```
$PATH/DTS -in input.dts -top top.top
```

Some of the parameters that are defined in the input file could be overwritten in the command line. Note: running a restart simulation is only possible by providing the restart file name in the command line as: `-restart restart_file_name.res`

Simulation Output files

There are a number of output files for each DTS simulations that are listed below.
vtu files: This is a list of files that shows the evolution of the system and can be opened by paraview.
tsi files: A list of files that shows the evolution of the system. These files can be converted to different file format using the Convert Script.
restart file: A file to restart a simulation. This file has “res” extension.
log file: Stores some information of about the current simulation. This file has “*log”* extension.
energy file: This file end with *“-en.xvg*” and contains system the bending energy, box size, volume and etc., at each step.

Command line options

All the option that can be defined in the command line. Also by executing `./DTS -h` you can see this list.

Identifier	Type	Default value	Description
-in	string	Input.dts	input file name
-top	string	topology.top	Topology file name
-b	int	1	initial time step
-e	int	100	final time step
-seed	int	36723	Random number seed
-defout	string	output	A general string for label a specific run
-ndx	string	Index.inx	Index file name
-restart	string	NO	restart file name
-angle	double	-0.5	minimum of cos of the angle between two faces
-minDist	double	1	Square of minimum distance between two vertices

-maxDist	double	3	Square of maximum link length

Input file option and format

The input file must have a dts extension. Below, you can find the most common options defined in this file. However, this file could contain more information, including inclusions and their interactions. Other options are defined in their corresponding sections.

```
Integrator                      = MC
MC_Moves                        = 1 1 1
Initial_Step                    = 1
Final_Step                      = 1000000
Display_periodic                = 1000
OutPutEnergy_periodic           = 1000
Restart_periodic                = 10000
Kappa                           = 15 0
Spont_C                         = 0
Seed                            = 37382
OutPutTRJ_TSI                   = 1000  10  TrjTSI
GeneralOutputFilename           = output
Cell_Size                       = 2.5 2.6 2.5
```

Integrator: The algorithm for updating the configuration of the system. Currently only “MC” available.
MC_Moves: An option for activating or disactivating the MC moves. First one is vertex move, second is the link flip and the last one is the inclusion moves. 1 1 1 means all the moves are active.
Initial_Step: Initial step of the simulation, usually it should be one, unless something else is intended.
Final_Step: The final step of the simulations
Display_periodic: Frequency of coordinate file in vtu file format.
OutPutEnergy_periodic: Frequency of energy file. Energy file contains several other information depending on global constrains applied on the system.
Restart_periodic: Frequency of restart file recording.
Kappa: Membrane bending and gaussian rigidity.
Spont_C: spontaneous curvature of the membrane
Seed: Random number seed.
OutPutTRJ_TSI: Frequency for writing trajectory frames in tsi file, the precision of the coordinates and the name of the folder that the files will be stored in.
GeneralOutputFilename: A general string to label all the generated files of a specific run with that string.
Cell_Size: The size of the unit cells for domain decompositions, should be larger than 1
Box_Centering_F: To canter the system inside a box after every m steps. The m should be provided as an input.
FreezingAGroup: Freezes a group of vertices. Requires an index file in which the name of the group should have been defined in the index file.
OutPutTRJ_BTS: Simulation trajectory output in a binary file format (bts extension). Requires 3 variables. (1) Frequency of saving the coordinates; (2) Precision for file saving (3) the name of the file.
Min_Max_LinkLenghtsSquare: Square of the maximum and minimum allowed edge size. Two double numbers. Default values are 1 and 3, corresponding to 1 and $\sqrt{3}$ in length.
MinfaceAngle: The minimum allowed dihedral angle between two neighbouring triangles. This number will indicate the minimum value for the cosine of the angle between the two triangle normals.

Applying Constraints

Certain commands can be added to the input file for simulation with constraints such as fixed global curvature or/and fixed volume.

Fixed frame tension simulations

For doing this, one needs to add below line to the input file. The last two numbers are the frame tension in the unit of [k_BT/a^2] and inverse frequency of applying the box size check algorithm to maintain the fixed membrane tension respectively. Note: this algorithm only makes sense for predoc membranes.

```
Frame_Tension                    = on Position_Rescale 0 5
```

Pressure deference or fixed volume simulation

```
;Volume_Constraint               = on eqtime    DP     K      gamma
Volume_Constraint                = on 100      -0.1    0      0
```

This command will link the system energy to a potential as

$$E_V = -\Delta PV + \frac{K}{2}(v-\gamma)^2 {V_0}^2$$

Where $v = \frac{V}{V_0}$ and $V_0 = \frac{\sqrt{A^3}}{6\sqrt{\pi}}$

Coupling to Fixed Global Curvature

To apply a constraint on the global curvature below command can be added to the input file.

```
;CouplingtoFixedGlobalCurvature = state kr m0
CouplingtoFixedGlobalCurvature = on 60 0.1
```

This will couple the system energy to a function as

$$E_s = \frac{k_r}{2A}(M - m_0 A)^2$$

Harmonic potential between two groups of vertices

To apply a harmonic potential between two groups of vertices, one need to add below command to the input file.

```
; HarmonicPotentialBetweenTwoGroups = on K L0 eqstep  Group1 Group2 nx ny nz
HarmonicPotentialBetweenTwoGroups = on 10 100 2000000 Group1 Group2 0 0 1
```

You also need to provide an index file for the group definition. This file should have *"inx"* extension. An example of index file is as below:

```
Group1 1
24
Group2 3
12 16 33
```

Applying Constant Surface Area

Constant total area based on the triangles area

```
;Apply_Constant_Area             = state eq_time gamma  Ka
Apply_Constant_Area              = on eq_time gamma Ka
```

Eq_time is number of the steps in which the energy of this command reaches its final value
Gamma is a number between zero and 1 that fixes the reference area. 1 provides the maximum possible area and zero is the minimum.

$$E_A = \frac{K}{N_T}(A - A_0)^2$$

$$A_0 = N_T(1+2\gamma)\sqrt{3}/4$$

Parallel Tempering

Adding below command allows for additional MC moves, by running simulations at multiple temperatures. The number of the bins will depends on the max and min beta and the provided threads for the openMP

```
Parallel_Tempering = on steps minbetta maxbeta
```

In the command line with -nt the number of the replicas will be defined.

Membranes in confined spaces

OPENDTS also allows simulating membranes while they are in confined regions. If the initial membrane is not within the defined region, bthe surface is forced to enter this region by defining an equilibrium time. However, if the eq time is short, it may fail to do so. There are four types of confided space defined in OPENDTS: TwoFlatParallelWall, Cuboid, Ellipsoid and EllipsoidalShell. To apply one of these confinements, one of the below commands must be added to the input file.

```
CoupleToRigidWalls              = on TwoFlatParallelWall  10000  X
```

“X” is a double number which represent half the distance between the two sandwiching walls.

```
CoupleToRigidWalls              = on Cuboid  10000  10 10 10
```

```
CoupleToRigidWalls              = on Ellipsoid  10000  10 10 10
```

```
CoupleToRigidWalls              = on EllipsoidalShell  10000  10 10 10  0.01
```

Osmotic pressure

Based on the Jacobus van 't Hoff equation $\Pi = icRT$.

$$\Delta E(\Delta V) = -P_0\left(V_0 \ln\left[1+\frac{\Delta V}{V}\right] - \Delta V\right)$$

Add below line to the input.dts file

```
Osmotic_Pressure = Type   time  gamma  P0
```

$$V_0 = \frac{\gamma}{6}\left(\frac{{A_0}^3}{\pi}\right)^{1/2}$$

$$V_0 = \frac{1}{6\pi^{1/2}} A^{3/2}$$

Where $A_0 = 3N_T \frac{3\sqrt{3}}{4}$

Inclusions (protein model)

To have inclusion in the simulations, we need to define different type of inclusions. Inclusion type must be defined at the end of the input file as

```
INCLUSION
Define 4 Inclusions
SRotation Type   K   KG  KP  KL  C0  C0P C0L
3         Pro1   10  0   0   0   0   0   0
3         Pro2   20  0   0   0   0   1   0
3         Pro3   20  5   0   0   0   0   0
2         Pro4   20  5   0   0   0   0   0
```

The Inclusion-inclusion interactions should be always placed at the end of the file as

```
Inclusion-Inclusion-Int
 1    1    1   2    0.0     0.0
```

`Type 1  n  A  B`

$$e_{i,j} = -A_{i,j} + B_{i,j}(1 + \cos[n\Theta])$$

`Type 2 A B n` Θ_0 `C` γ_0

$$e_{i,j} = -A_{i,j} - B_{i,j}\cos[n(\Theta - \Theta_0)] - C_{i,j}C_{i,j}(\gamma - \gamma_0)^2$$

Defining inclusions type inside the input file does not means that they are present in the simulation. To create inclusion in the simulations, we need to define them either inside the "*tsi*" file or tell the input file to generate them. Currently we can only generate random distribution of inclusions from the input file. However, some can manually or use a script to add such inclusions to the tsi file.

```
GenerateInclusions
Selection_Type Random
TypeID     1     2     3
Density    0.0   0     0
```

Topology file format

Topology file provides information about the position of all the vertices and how they are linked. There are two types of file that you can provide as a topology file; *top* and *tsi* file format.

***top* file:** This file has a "top" extension and contains a list of TS files in q (TS file) file format. An advantage of this topology format is that it allows us to feed multiple TS file into the system. A disadvantage is that it does not include inclusions.

***tsi* file**: tsi topology is single file that is also single frames of OPENDTS trajectories. An advantage of this topology file is that it includes the inclusions information.

TS file

Triangulated surface files that can be read by FreeDTS either has "q" format or "tsi" format. The Generate script can generate files in both formats (see Generate script section).

"tsi" format files

The following shows a part of a .tsi file with all necessary keywords highlighted in bold. Every .tsi file starts with a line calling version 1.1. The next line defines the box size (x, y, and z) of the system in nm. The next three sections describe the TS mesh. Each section starts with a keyword (vertex, triangle and inclusion) and their corresponding number. Here, we have 130 vertices (the numbering starts from 0). Each vertex has an index and a position in x, y and z (in nm). The 130 vertices are connected via 256 triangles. Again, every triangle has an index (starting from 0) and is defined by the vertices the triangle connects, i.e. triangle 0 connects vertices 11, 55 and 43. Furthermore, a .tsi file

can have a (protein) inclusion section. Here, there are three inclusions from two different types. Again, each inclusion has an index. The index is followed by the inclusion type (here: type 1 for inclusions 0 and 1, type 2 for inclusion 2) and the corresponding vertex index. The last two (floating point) numbers describe a unit two-dimensional vector (sum of both numbers must be one!) which defines the orientation of the inclusion with respect to the bilayer normal.

```
version 1.1
box   50.0000000000     50.0000000000     50.0000000000
vertex         130
0     21.1606233083     25.4394806652     25.5960855271
1     27.0284995400     23.2012757654     21.6715285158
2     26.9921761232     25.5136587223     28.0195776981
3     23.3273229896     26.2315165676     28.0075875808
4     26.2722773116     26.3271061222     28.1420707299
5     22.0396876425     23.6080597437     26.8858740866
.
.
.
125   21.5556280860     25.5595098219     26.5363425272
126   23.2182025326     26.8060871266     21.5195141902
127   25.3199303865     24.3519379911     20.6752314764
128   28.0093200458     22.6356946990     23.4685318698
129   21.4000741257     26.5841316766     25.2761757772
triangle       256
0     11     55     43
1     94     75     14
2     64      3     91
3     59     52     40
.
.
.
253   33    109     44
254   53     69     47
255   85      6     74
inclusion        3
0      1     22      0     1
1      1      5      0     1
2      2     30      0     1
```

"q" format files

The previous tsi file in q format can be seen below.
Line 1: Box information (3 double numbers).
Line 2: Number of vertices (1 integer number; Let's call it NV).
Line 3 to NV+2: Vertex ID and coordinate (1 integer number and 3 double numbers).
Line NV+3: Number of triangles (1 integer number; Let's call it NT).
Line NV+4 to NV+NT+3: Triangle ID and ID of its vertices (4 integer number).

```
50.0000000000     50.0000000000     50.0000000000
130
0     21.1606233083     25.4394806652     25.5960855271
1     27.0284995400     23.2012757654     21.6715285158
2     26.9921761232     25.5136587223     28.0195776981
3     23.3273229896     26.2315165676     28.0075875808
4     26.2722773116     26.3271061222     28.1420707299
5     22.0396876425     23.6080597437     26.8858740866
.
.
.
125   21.5556280860     25.5595098219     26.5363425272
126   23.2182025326     26.8060871266     21.5195141902
127   25.3199303865     24.3519379911     20.6752314764
128   28.0093200458     22.6356946990     23.4685318698
129   21.4000741257     26.5841316766     25.2761757772
256
0     11     55     43
1     94     75     14
2     64      3     91
3     59     52     40
.
.
.
253   33    109     44
```

```
254   53     69     47
255   85      6     74
```

Generate Script

Generate (**GEN**) bindery allows you to create triangulated surface files with different topologies in two different file formats, *"q"* or *"tsi"*. Three options exist, flat bilayer periodic in x and y directions, cylinder periodic in the x-direction and a closed sphere. To see all the options, execute `$path/GEN -h`
Some example:
To generate flat bilayers

```
$path/GEN -box 50 50 30 -type flat -o topol.q
```

Generating closed membranes

```
$PATH/GEN -box 50 50 50 -type tetrahedron -N 20 -o topol.q
```

Note: You can force this structure to become sphere by using Ellipspoidalshell command and a short simulation.

```
CoupleToRigidWalls              = on EllipsoidalShell  10000  10 10 10  0.01
```

Convert Script

The convert (**CNV**) bindery allows for converting "*tsi"* and "*q*" files to each other and several other different file format such as "*vtu"* and "*gro*". To see all the options, execute `$path/GEN -h`

Tutorials

T1: Framed membranes

The simplest simulation that can be performed by FreeDTS is a flat membrane (a fluid elastic surface) in a PBC box. To do this, we have to first generate a flat TS file. This can be done by using **GEN** binary. Using this you can generate a TS file in a q file format using the below command line.

```
$path/GEN -box 30 30 100 -type flat -o topol.q
```

This command creates a TS file in *q* file format with a box size of 50*50*100. Next, this file name should be added to a topology file name with top extension (here it is named top.top), and we should add the below line in the top file.

```
topol.q 10
```

The number in front of the file gives an *id* to the entire mesh (the value is not important but needs to be unique if multiple files are included). To run a simulation, an input file (with dts extension) is required to define the simulation parameters (see below box and the input file format section).

```
Integrator                      = MC
MC_Moves                        = 1 1 1
Initial_Step                    = 1
Final_Step                      = 100000
Display_periodic                = 1000
OutPutEnergy_periodic           = 100
Restart_periodic                = 10000
Kappa                           = 20 0
```

```
OutPutTRJ_TSI                    = 1000   10   TrjTSI
GeneralOutputFilename            = output
```

Using these two files we are now able to run DTS simulations as:
`$PATH/DTS -in input.dts -top top.top -seed 76532`

Everything above could be done by running the *./run.sh* script in the T1/dts_tutorials/T1-FramedMembrane_a folder in the source code.

To make a membrane tensionless, we need to add below command to the input file (such the example in the input file section).

```
Frame_Tension                    = on Position_Rescale 0 5
```

In the dts_tutorials/T1/T1-FramedMembrane_b folder run the *./run.sh* script to perform this section.

Each run will give multiple outputs. Folder VTU_Frames contains *paraview* readable files to visualize the evolution of the system. TrjTSI folder provides files in *tsi* format for analysis of the simulation. *output-en.xvg* file contains information about system energy, box size (for constant tensions simulations) membrane volume for constant volume simulations (see the vesicle tutorials) …. as a function of the simulation steps.

T2: Vesicle simulations

Perpetrating a Vesicle

For a simulation of a vesicle, we first need to create a vesicle structure. For this, we use **GEN** script to generate a tetrahedron and run a short simulation with *EllipsoidalShell* confinement to shape the tetrahedron into a vesicle. The below command creates a *TS* file in the shape of tetrahedron. The size of the tetrahedron can be changed by -N option

```
$PATH/GEN -box 50 50 50 -type tetrahedron -N 20 -o topol.q
```

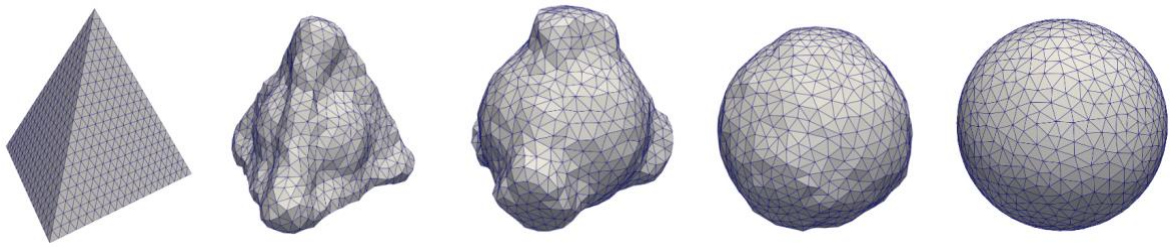
Next, we add the generated TS file to the topology file and run a normal simulation while having below line in the input file (for more see “Membranes in confined spaces” section in the manual).

```
CoupleToRigidWalls               = on EllipsoidalShell  10000  10 10 10  0.01
```

After about 20K steps, you should get a spherical TS file (the run steps may differ for different system)

In the dts_tutorials/*T2-Vesicle* folder run the *./run.sh* script to perform this part. Also see the video in the folder.

***Note**: one can easily create a TS file with a spherical shape. However, we prefer this method because it allow us to create a TS file with regular edge size (between range required for DTS simulations) on many different shapes.*



Constant Volume Vesicle

Using the obtain spherical TS file, we can obtain a shape of a vesicle with a constant volume. For this use the **CNV** script and convert last frame of the previous simulation into a q file for a new simulation.

```
$PATH/CNV -in output20.tsi  -o vesicle.q
```

Remove below line from your input file

```
CoupleToRigidWalls              = on EllipsoidalShell  10000  10 10 10  0.01
```

And add below line to the input file (see “Osmotic pressure or fixed volume simulation” section)

```
Volume_Constraint               = on 1000000     0  0.1     0.4
```

In the dts_tutorials/*T2-Vesicle* folder run the *./run_2.sh* script to perform this part.

T3: Proteins

In this tutorial, we want to make a flat membrane covered by 20% curvature inducing proteins.

Inclusions: Proteins are modeled as inclusions. The type and the model parameters of the corresponding inclusion must be defined in the input file (*.dts file). Below is an example of defining two inclusion types in the input files.

```
INCLUSION
Define 2 Inclusions
SRotation Type   K   KG  KP  KL  C0   C0P C0L
0         Pro1   10  0   0   0   0.4       0   0
0         Pro2   20  0   0   0   -0.4      0   0
```

Interactions: Inclusion-inclusion interactions are also defined in the input file. For interaction type 1 (the tested version) $E_{ij} = -A + B\cos N\Theta$ it will be as:

```
Inclusion-Inclusion-Int
i    j    ftype   N     A     B
```

In this dts file, this will be defined as below. Please note: this section MUST always be the last section of the input file (*.dts file).

```
Inclusion-Inclusion-Int
1    1    1   2     2     0.0
1    2    1   2     0     0.0
2    2    1   2     2     0.0
```

Defining inclusions: The number of the inclusions in a simulation can be defined in two different ways. If the simulation starts from a q file, a random distribution of inclusions must be defined in the input file.

```
GenerateInclusions
Selection_Type Random
TypeID     1     2     3
Density    0.3   0.1     0
```

However, if the simulation started from a tsi file, inclusions will be read from the inclusion section of this file.

In this first step of the tutorial, we will generate the inclusions from an input file and force them to cluster to enhance the aggregation of the inclusions (to accelerate the process). For this, we will turn off the vertex, edge flip and box change moves.

```
MC_Moves                        = 0 0 1
```

In the dts_tutorials/*T3-Membranes_Inclusions* folder run the *./run.sh* script to perform this.

Next, we use the last frame (tsi file) of the previous simulation to start a new simulation with all the required moves.

In the dts_tutorials/*T3-Membranes_Inclusions* folder run the *./run_2.sh* script to perform this.

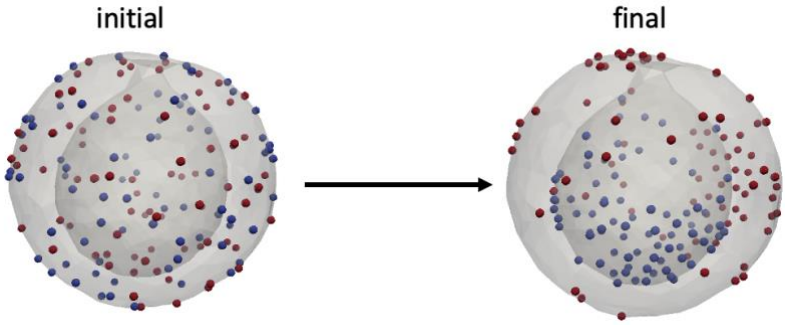
T4: Protein Sorting

With FreeDTS, we can look at the protein sorting without allowing for membrane shape change. In this way, we can talk for example the shape of membrane structure from Cryo-ET and obtain protein organizations on the membrane. For this, we need to turn off the vertex, edge flip moves and only allow for inclusion moves.

```
MC_Moves                          = 0 0 1
```

We will take the stomotosyte shape in the tutorial 2 and add two different protein type.

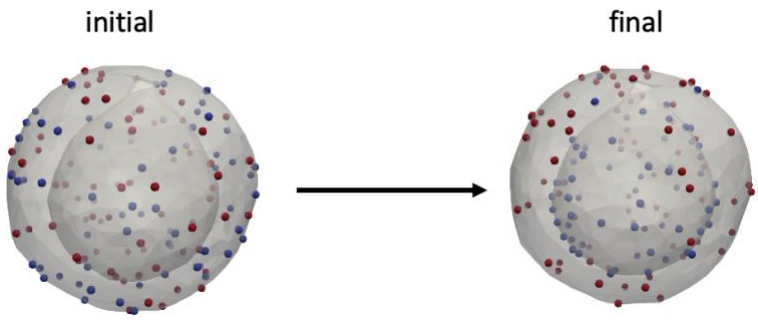
```
INCLUSION
Define 2 Inclusions
SRotation    Type    K    KG   KP   KL   C0      C0P   C0L
1          Pro1    20   0    0    0    0.3       0    0
1          Pro2    20   0    0    0   -0.3       0    0
1          Pro2    20   0   10    0    0    1    0
GenerateInclusions
Selection_Type Random
TypeID      1      2      3
Density     0.1    0.1   0.0
Inclusion-Inclusion-Int
1     1     1    0      1      0.0
2     2     1    0      1      0.0
1     2     1    0      1      0.0
```



In the dts_tutorials/ *T4-protein_sorting* folder run the *./run.sh* script to perform this.

The same simulation can be done a system that inclusion one prefers inclusion 2 but still the get sorted by the membrane curvature. For this

```
Inclusion-Inclusion-Int
1     1     1    0      0      0.0
2     2     1    0      0      0.0
1     2     1    0      1      0.0
```



In the dts_tutorials/ *T4-protein_sorting* folder run the *./run_2.sh* script to perform this.

T5: Pulling a membrane nanotube

To pull a nanotube from a membrane, we need to have a membrane under tension (see previous tutorials for such a simulations). Then we can apply a harmonic potential between two groups, one group can be the single vertex while the other group can be all the other vertices. Using a corresponding index file (See the index file section) and adding below command to the input file we have everything we need to pull a nanotube from a flat membrane.

```
HarmonicPotentialBetweenTwoGroups = on 10 100 1000000 Group1 Group2 0 0 1
```

Next, just run below command and wait for the nanotube to form.

```
$PATH/DTS -in input.dts -top top.top -seed 76532 -ndx index.inx
```

In the dts_tutorials/*T4-tether_pulling* folder run the *./run.sh* script to perform this tutorial.

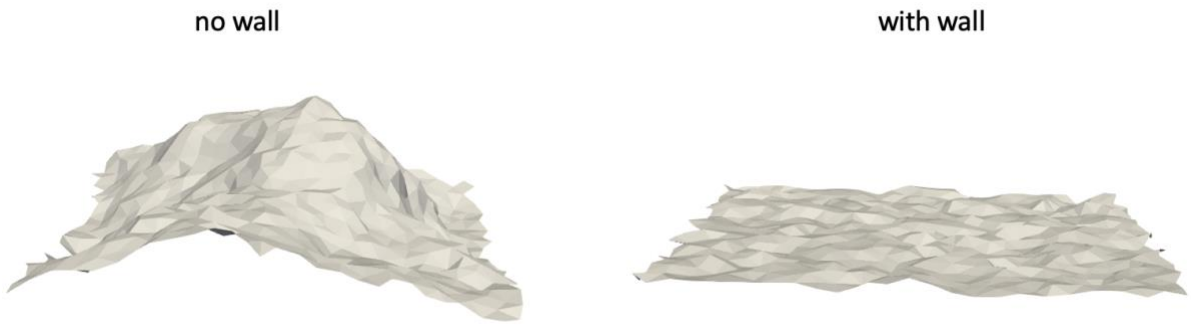
Note: this tutorial might take long.

T6: Confirmed membranes

How membranes behave in a confined space. This can be studied using FreeDTS. In this tutorial we will look how two sandwiching hard walls. For this, we need add below command to the inpot file

```
CoupleToRigidWalls              = on TwoFlatParallelWall  10000  0.5
```

We also set the kappa to 4 to make the



In the dts_tutorials/*T6_confined_membranes* folder run the *./run.sh* script to perform this tutorial.