Introduction

TS2CG is used to build coarse-grained membrane models with user defined shape and compositions. Initially it was developed for backmapping dynamically triangulated simulation structure into its corresponding molecular model. This gives us the possibility to incorporate experimentally obtained membrane shape and compositions and generate CG membranes initial structure.

In Figure 1 the general workflow of TS2CG is exemplified for a vesicle containing a single protein (shown as yellow bead). The initial triangulated surface is rescaled to the desired system size and the two monolayers are generated. In order to have enough points for the subsequent lipid placement, the number of vertices in both monolayers is increased using a pointillism operation, i.e. each triangle is divided into four new triangles thereby increasing the number of vertices by a factor of four. In the last steps, proteins and lipids are placed on the respective vertices. For more details on the method, please refer to the original paper: Pezeshkian, W., König, M., Wassenaar, T.A. *et al.* Backmapping triangulated surfaces to coarse-grained membrane models. *Nat Commun* **11,** 2296 (2020) (https://doi.org/10.1038/s41467-020-16094-y)

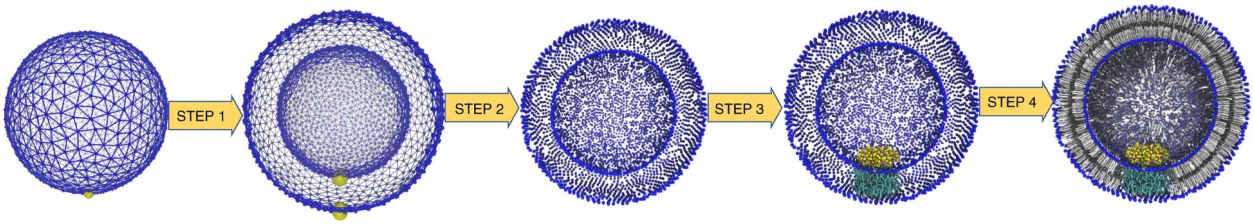


Fig. 1. Steps in backmapping a triangulated surface (TS) mesh using TS2CG. (Step 1) A TS structure of a vesicle containing one protein (yellow bead) is rescaled and two TS structures corresponding to the two monolayers that are generated. (Step 2) Using a Pointillism operation, the number of vertices is increased. (Step 3) The CG protein structure together with a membrane segment is placed at the appropriate TS position. (Step 4) Lipids are placed at the remaining positions and the configuration is ready for subsequent MD simulation.

Currently, TS2CG version 1.1 can utilize two types of TS input file formats: Files with *.q* and *.tsi* extension. While both are quite similar, *.tsi* files are more generic and future developments will be based on this file format. For more details about the *.tsi* file format see the end of this file. In tutorials 1 to 5 we will use a *.tsi* file called *Sphere.tsi* to build a simple vesicle and subsequently add different membrane domains with and without protein insertions.
Additionally, TS2CG can be used to create well-defined (analytical) shapes from scratch. In tutorial 6 we will demonstrate how to build a curved bilayer and maintain the curvature using a shape-preserving wall.

Download and install TS2CG

Download the latest version of the TS2CG from
https://github.com/marrink-lab/TS2CG1.1

For compiling, **gcc version 8.3.0 or above** is needed.
In the source code folder, execute the script “compile.sh” as

```
./compile.sh
```

In this folder, two binary files will be generated: **PLM** and **PCG** (and a **SOL** script for adding solvent to the systems). PLM performs pointillism (Step 1 and 2) and PCG performs Membrane builder (Step 3 and 4).

Tutorials

From the download TS2CG folder, you can find all the files needed for these tutorials in the tutorials folder.

Tut1: Creating a simple vesicle

In this tutorial we will use a simple TS file (sphere) to create a vesicle. We choose this shape because it is small and can be run on your local machines. However, the same scheme can be used for any other TS files.
From the *files* folder select the *Sphere.tsi* file. Use a text editor to open this and familiarize yourself with the structure. See *.tsi file format* at the end of this document for more information about the file format. You can also use pymol or paraview to visualize the structure (see *Visualizing .tsi or .q files*).

The first in backmapping any TS file to a CG structure is to increase the number of vertices using a pointillism operation (done by **PLM**). In the same step we also generate the two monolayers.

```
Path_to_TS2CG/PLM -TSfile Sphere.tsi -bilayerThickness 3.8  -rescalefactor 4 4 4
```

In the *pointvisualization_data* folder, you will find gromacs compatible structure files (*.gro*) for the upper and lower monolayer including a topology file (.top) each as well as paraview compatible *.vtu* files for both monolayers. You can have a look at the created points using vmd or paraview.

The second step to create a vesicle is two place lipids on the generated points using **PCG**. For this you need to write a *.str* file defining the lipid composition of both monolayers. Using any text editor, create an *input.str* file and write the following text in it:

```
[Lipids List]
Domain 0
POPC  1       1       0.64
End
```

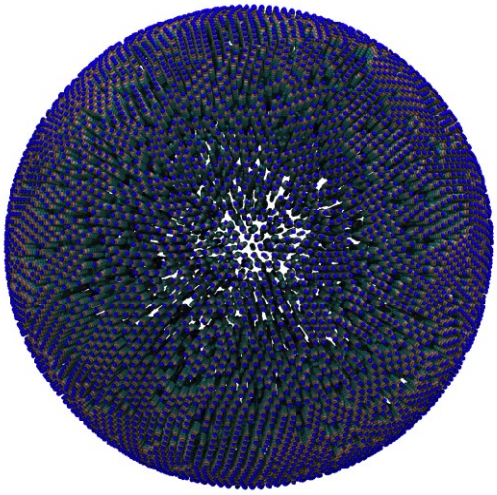
This implies that your system should contain only one lipid domain with POPC in both monolayers using an area per lipid (APL) for POPC of 0.64. To know more about the *.str* file format and other options see the User Manual.

The other thing we need is a lipid structure file (*.LIB*). This file simply defines the lipid connectivity for placing the lipid beads on the previously generated points. Making this file is easy but might be time consuming for many different lipids. (See the User Manual for the exact file format). Luckily, we already have a file that contains all Martini3 lipids called *Martini3.LIB*, you can find it in the *files* folder.

Using these two files now you can execute PCG:

```
Path_to_TS2CG/PCG  -str input.str -Bondlength 0.2 -LLIB Martini3.LIB -defout system
```

The outputs are *system.gro* and *system.top*:



```
 ;This file was generated by TS Back Mapping
 [ system ]
 Expect a large membrane
 [ molecules ]
; domain 0
 ;  in the upper monolayer
        POPC  6256
; domain 0
 ;  in the lower monolayer
        POPC  3876
```

Fig. 2. Snapshot of the simple POPC vesicle made with vmd (left) and the corresponding topology file (right).

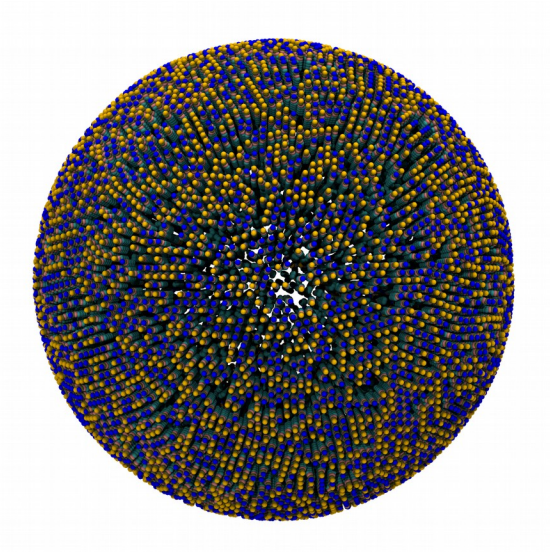
By including the force field headers to the *system.top* file, you can run the system using gromacs.

Tut2: Creating lipid Mixture

Repeating the previous tutorial but with the *.str* file shown below, you will create a 50/50 mixture of POPC and DOPC in both monolayers.

```
[Lipids List]
Domain 0
POPC  0.5     0.5     0.64
DOPC 0.5     0.5     0.67
End
```

Executing PCG will generate two output files *system.gro* and *system.top*:



```
;This file was generated by TS Back
Mapping
 [ system ]
 Expect a large membrane
 [ molecules ]
; domain 0
 ;  in the upper monolayer
        POPC  3056
        DOPC  3056
; domain 0
 ;  in the lower monolayer
        POPC  1893
        DOPC  1893
```

Fig. 3. Snapshot of the mixed POPC (blue) / DOPC (orange) vesicle made with vmd (left) and the corresponding topology file (right).

Tut3: Membrane domains

To make a membrane containing two or more different lipid domains, we need to modify the TS file. These changes can be made manually or by using a script. Here we only try it manually. First use the command below to obtain a *.gro* file containing all the vertices

```
Path_to_TS2CG/PLM -TSfile  Sphere.tsi -bilayerThickness 0  -rescalefactor 0.2
0.2 0.2 -Mashno 0
```

This time, we reduced the size of the original *.tsi* file (using -rescalefactor 0.2 0.2 0.2) and kept the initial number of vertices (using -Mashno 0). This allows for an easier selection of points when done manually (e.g. by click and select in vmd). Now, open the file *Upper.gro* from the *pointvisualization_data* folder using vmd. You will see all the vertices of the triangulated

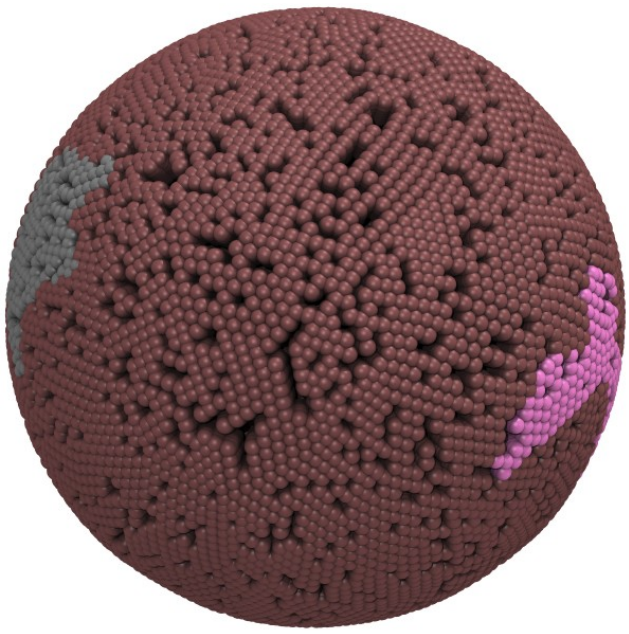
surface. Find a few beads for which you want to create a new domain of lipids. I selected vertices [113, 117, 124, 48, 112] for domain number 1 and [74, 82, 81, 45] for domain number 2 and the rest remains as domain 0.

Next, open the *.tsi* file using a text editor and find the line of selected vertices. Add the domain ID to the end of the line (here: add a 1 for vertices with index 113, 117, 124, 48, 112 and a 2 for vertices with index 74, 82, 81, 45). Note: Lines that do not have any domain ID belong to domain 0 per default. After doing so, you need to modify the *.str* file to define the lipid composition of each domain:

```
[Lipids List]
Domain 0
POPC 1  1  0.64
End
Domain 1
DOPC 1  1  0.64
End
Domain 2
POPE 1  1  0.64
End
```

Then execute PLM and PCG using commands below (as done in the previous tutorials). This will generate a vesicle with three lipid domains.

```
Path_to_TS2CG/PLM -TSfile Sphere.tsi -bilayerThickness 3.8 -rescalefactor 4 4 4
Path_to_TS2CG/PCG -str input.str -Bondlength 0.2 -LLIB Martini3.LIB -defout system
```



Tut4: Adding proteins to a membrane

In this tutorial we are going to add two types of proteins to a vesicle containing POPC lipids. These proteins are named P1 and P2 in which we need their corresponding gro file that can be found in the tutorial folder. Open these files and change the first line to the name of the proteins. Lets say, protein1 and protein2. Also these files should be included in your str file as

```
include  P1.gro
include  P2.gro
```

Next, we need to perform some tricks on the TS and str file. These changes can be made manually or by using a script. Here we only try it manually.

First use the below command to obtain a gro file of vertices positions.

```
Path_to_TS2CG/PLM -TSfile  Your_TS_File -bilayerThickness 0  -rescalefactor
0.2 0.2 0.2 -Mashno 0
```

Then in the pointvisualization_data folder, open the Upper.gro using vmd. What you see is all the vertices of the TS surface. Find some vertices at which you would like to place proteins. We selected vertices 22 and 5 to place two copies of protein1 and vertex 30 to place one copy of protein1. Using a text editor, open your TS file and go to the bottom of the file in the inclusion section. As we want to only add 3 proteins, change the zero to 3 and then in the next line, add the proteins information. For each protein, you need 3 integer numbers and 2 float numbers. The first number is the protein id that should start from 0 and incrementally get increased for each next the protein. The second number is the protein type id. Which for two of them is 1 and for 1 of them is 2. The third number is the id of the vertex that the protein will set on. The last two numbers are the orientation of the protein in the local coordinate of the vertex that should be a unit two dimensional vector. So the inclusion section of this TS file should be something like.

```
inclusion      3
 0      1  22  0       1
 1      1  5   0       1
 2      2  30  0       1
```

The last step is to define the proteins in the str file. In addition to including the protein gro file names in the header, there should be some extra information. For this case we add following lines.

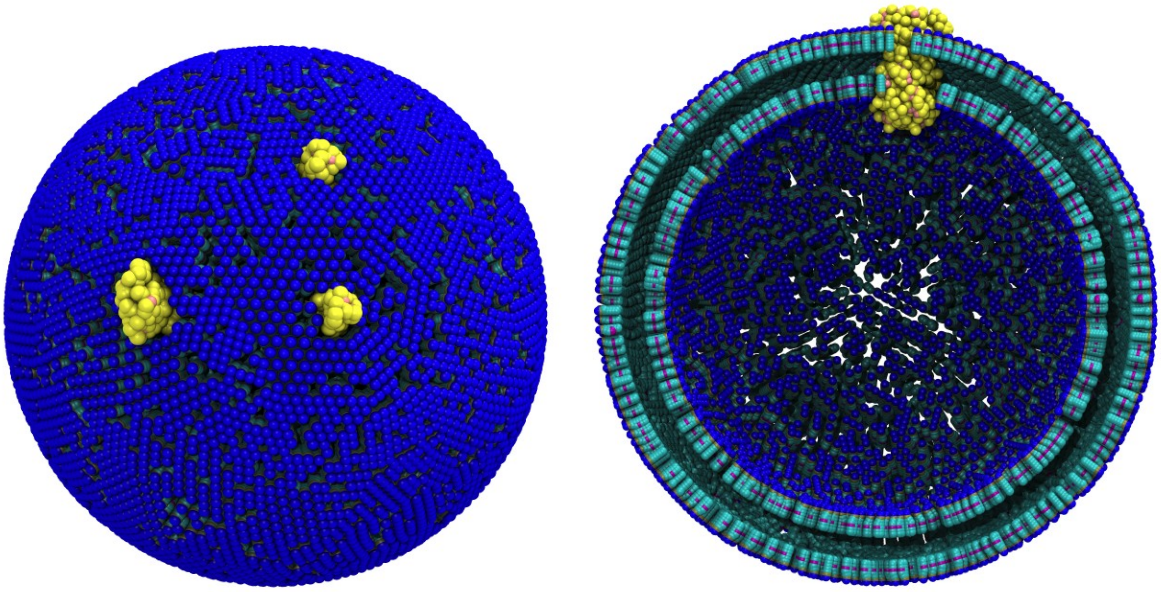
```
[Protein List]
protein1 1 0.01 0 0 -3.7
```

```
protein2  2 0.01 0 0 -2.4
End Protein
```

The first line indicates that we want to define proteins and should be always written as it is. The same is true for the fourth line. The lines in between the first and the last depend on the amount of unique proteins you want to include. Here we got two, therefore we only have two lines in between. The first letter is the protein name in the gro file. The second number indicates the protein type ID (this ID was used in the TS file in the inclusion section). The last number indicates how much we should move the proteins in the normal direction of the membrane surface. The remaining three numbers do nothing in the current approach.

Since your TS, str files are ready, you can execute PLM and PCG using the commands below (or as previous tutorials).The result should be a POPC membrane with three proteins.

```
Path_to_TS2CG/PLM -TSfile  Sphere.tsi  -bilayerThickness 3.8  -rescalefactor 3 3 3
Path_to_TS2CG/PCG  -str input.str -Bondlength 0.2 -LLIB Martini3.LIB -defout system
```



Tut5: Proteins with a specific domain

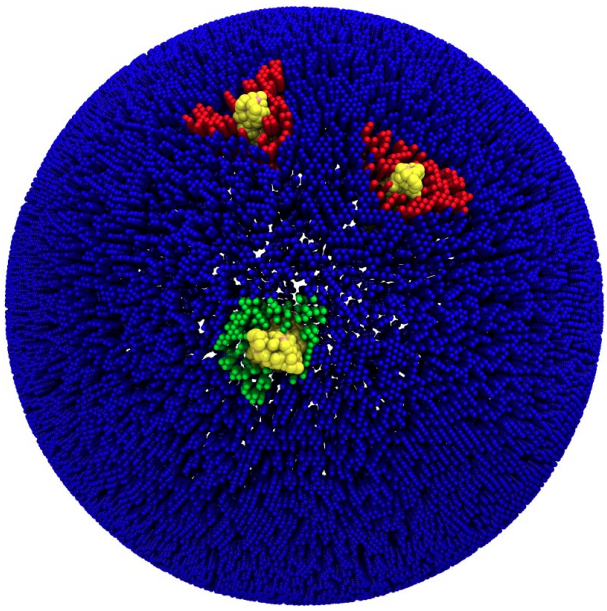
In the previous tutorials, we could have defined a specific lipid domain for the proteins, somehow combining Tut3 and 4. So let's change the domain ID of vertices 22 5 30 that are supposed to be the location of the proteins in the TS file as demonstrated in Tut3. We set 22 and 5 to domain 1 and 30 to domain 30. We also change the lipid section in the str file to:

```
[Lipids List]
```

```
Domain 0
POPC 1 1   0.65
End
Domain 1
DOPC 1 1   0.65
End
Domain 2
POPE 1 1   0.65
End
```

Now perform PLM and PCG again:

Path_to_TS2CG/PLM -TSfile Sphere.tsi -bilayerThickness 3.8 -rescalefactor 6 6 6
*Path_to_TS2CG/*PCG -str input.str -Bondlength 0.2 -LLIB Martini3.LIB -defout system



You may not be happy about the lipid domain around the proteins. Since the resolution of the TS file is low, the shape of the lipid domain around the proteins gets strongly affected by the discretization. Nevertheless this can be improved with a little bit of effort.
Write the previous protein information in the *Sphere.tsi* file without adding the domains. Meaning add the block below at the end of the tsi file.

```
inclusion              3
        0       1   22          0.00   1.000
        1       1   5   0.000           1.000
```

```
2       2   30          0.000          1.000
```

We are going to add the domains using a small script. Using the command below we will extend the number of the triangles and global size of the system without affecting the shape of the TS file.

```
Path_to_TS2CG/PLM -TSfile Sphere.tsi -bilayerThickness 0 -rescalefactor 8 8 8
```

The output tsi file is called extended.tsi. Change the name of this file to Sphere_2.tsi.

Then compile a script (tsi_format.cpp which can be found in the file folder of the tutorial folder) for creating domains using c++ as

```
g++ -o domain tsi_change.cpp
```

And then execute this command as

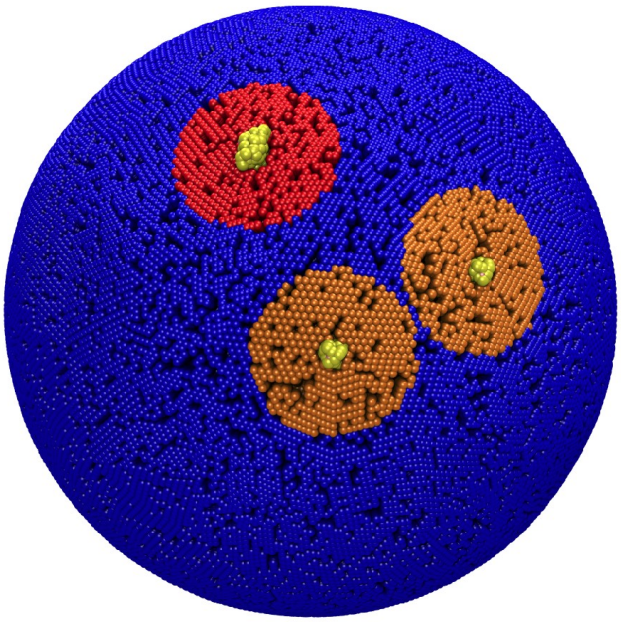
```
./domain Sphere_2.tsi Sphere_domain.tsi 8
```

The Sphere_domain.tsi file is the new tsi file which contains the Sphere_2.tsi surface and protein information and also a domain with size of 8 nm,created around each protein where the domain id and the protein id are the same. Next, we are going to perform **PLM** and **PCG** using the Sphere_2.tsi file. However, instead this time we ask PLM to not to extend the surface, only creating the monolayer surfaces, and an input folder for PCG.

```
Path_to_TS2CG/PLM -TSfile Sphere_domain.tsi -bilayerThickness 4 -rescalefactor 1 1 1 -Mashno 0
```

```
Path_to_TS2CG/PCG -str input.str -Bondlength 0.2 -LLIB Martini3.LIB -defout system
```

You should get an output as shown below figure.



Tut6: Fixed shapes

Without a need for a TS file, **PCG** also allows for creating membranes with certain fixed shapes such as flat bilayers, sphere, cylinder and other periodic shapes that can be written as a sum of fourier 1D modes. This is a new feature of **PCG** and therefore some limitations exist. There is also a wall option that allows you to keep the shapeduring both the equilibration and production run.

> *Path_to_TS2CG/*PCG -str input.str -Bondlength 0.2 -LLIB Martini3.LIB -defout system -function analytical_shape

In the str file the shape information is defined as below for creating different shapes.
Note: Not all the options need to be specified in the str file. The shape type option is needed but you could use the default values for the rest.

Cylinder

```
[Shape Data]
ShapeType  Cylinder
Box 40 40 40
Density 2
Thickness  4
WallDensity  1 1
Radius     12
End
```

1D Fourier Shape

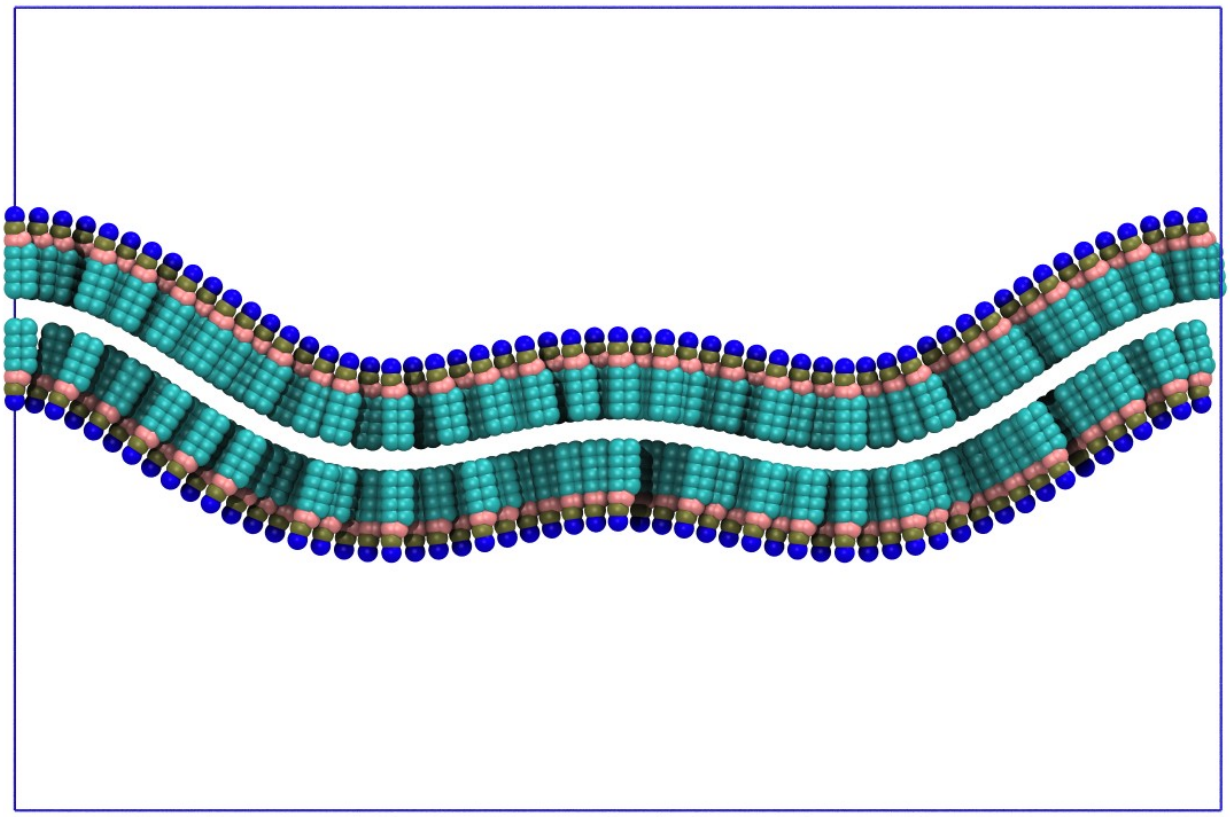
```
[Shape Data]
ShapeType 1D_PBC_Fourier
Box 20 10 20
WallRange 0 1 0 1
Density 3 1
Thickness 4
Mode 1.5     1   0
Mode 0.5     2   0
End
```

Sphere

```
 [Shape Data]
ShapeType  Sphere
Box 40 40 40
Density 2
WallDensity  1 1
Thickness  4
DL  0.2
Radius     15
End
```

Flat

```
 [Shape Data]
ShapeType  Flat
Box 40 40 40
Density 0.4  1
Thickness  4
WallRange 0 1 0 1
End
```



Creating the Wall

To generate the wall bead, you need to add the following line to the PCG command line:

-Wall -WallBName WL

Note, it is all on you how to make the wall bead to interact with the bilayer for keeping the shape. However, the best option is to create a repulsive LJ interaction between the wall beads and lipid chain beads while the wall beads are invisible to other beads in the system

General information about how to run the outputs from TS2CG

1) A short (100 steps) energy minimization with softcore potentials while restraining the lipid headgroups and protein backbones.
2) Normal energy minimization without solvent
3) Short equilibration without solvent
4) Solvation
   Note: Before solvation, always make sure the box size is what you need
   ./SOL -in input.gro -tem water.gro -unsize 3 -ion 0 0 -o solvated.gro -Rcutoff 0.4
5) energy minimization
6) Equilibration
7) Production run.

.tsi file Format

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** of TS2CG. 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). Additionally, a vertex can have a domain id, e.g., vertices 1, 126 and 127 belong to domain 1. The default domain is 0. 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 1

```
   2    26.9921761232    25.5136587223    28.0195776981
   3    23.3273229896    26.2315165676    28.0075875808        2
   4    26.2722773116    26.3271061222    28.1420707299
   5    22.0396876425    23.6080597437    26.8858740866        2
   .
   .
   .
 125    21.5556280860    25.5595098219    26.5363425272
 126    23.2182025326    26.8060871266    21.5195141902        1
 127    25.3199303865    24.3519379911    20.6752314764        1
 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
```

Visualizing *.tsi* or *.q* files

From the TS folder in the tutorial folder copy a TS file in your working folder. These files can be visualized using pymol, vmd or paraview.

pymol

You need PyMOL2q.py from the TS2CG folder then do the following steps

1) Load pymol
2) On pymol window, execute "**run PyMOL2q.py**"
3) Then execute "**loadq *Your_TS_File***"

paraview or vmd visualization of the generated points

Path_to_TS2CG/PLM -TSfile Your_TS_File -bilayerThickness 0 -rescalefactor 1 1 1 -Mashno 0

Then in “*pointvisualization_data*” folder, you can find several files and it is the gro file which can be used for visualization.

How to obtain a TS file

TS files can be obtained from different sources. 1) Form DTS simulations, 2) converting density points to mashes for example using Geogram and GAMer.