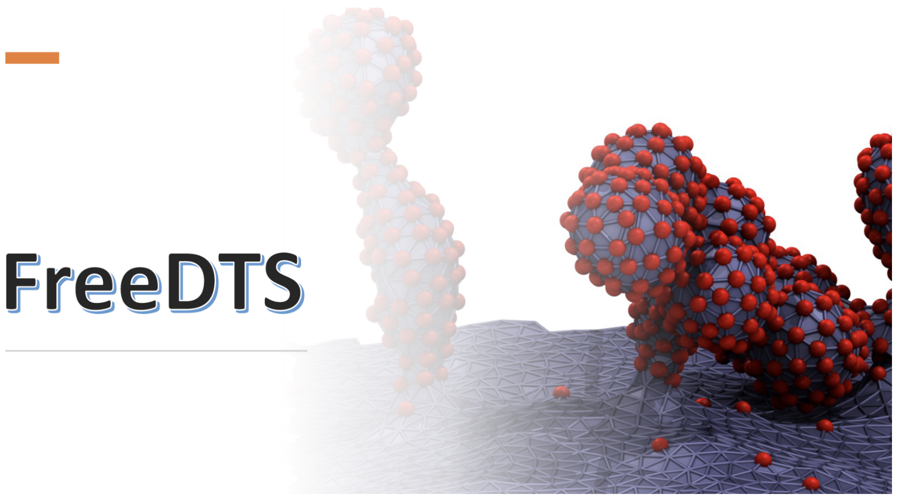
FreeDTS

Version 2 Manual



Simulation of complex surfaces and biomembranes

Weria Pezeshkian

December 4, 2024

The online and latest version of the manual can be found on the official FreeDTS GitHub Wiki at the following link: https://github.com/weria-pezeshkian/FreeDTS/wiki/User-Manual-for-version-2.

Contents

FreeDTS

FreeDTS is software to perform computational research on biomembranes at messocpic length-scale. 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. Several algorithms are included into the software that allow for simulation of framed membrane with constant tension, vesicles with various fixed volume or constant pressure difference, confined membranes into the fixed region of the space, constant fixed global curvature and application for external forces on regions of the membrane. In addition, the software allows one to turn off the shape evolution of the membrane and only explore inclusions organization. This allows to take realistic membrane shapes obtained from Cryo-ET and obtain heterogeneous organization of biomolecules which can be backmapped to finer simulations models.

About FreeDTS version 2

Development of this version commenced in March 2024. The structure of the code differs significantly from Version 1, enabling easier extension and maintenance of the software.

Latest Updates

Version 2 of FreeDTS introduces several enhancements and structural improvements over its predecessor. These changes facilitate a more streamlined development process and ensure greater flexibility for future updates. The core features of FreeDTS Version 2 include:

- **Enhanced Algorithm Diversity:** Includes algorithms for treating surfaces with open edges, dynamic topology changes, and **importantly** surfaces with multiple layers of vector fields.
- **User-Friendly Customization:** Users can easily create customized versions tailored to their specific needs and applications.
- **Improved Performance:** Significant improvements have boosted performance, and many features are now multithreaded using OpenMP.
- **Flexible Energy and Curvature Calculation:** Methods for calculating energy and curvature can be expanded without extensive modifications.

Compiling The Source Code

The second version requires the C++11 library and can be compiled with or without OpenMP support. Additionally, MPI parallelization is available for parallel tempering. To compile the FreeDTS source code, simply run the `compile.sh` script.

```
./compile.sh
```

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

Running a Simulation

To run a simulation, you need an input file and a topology file. The input file can be empty, allowing the software to use default settings for the simulation. However, the topology file must define a valid triangulated structure. This structure can be generated using the GEN script or obtained from other software tools. The topology can either consist of multiple `.q` files listed in a `.top` file, or a single file. For details on the structure and format of these files, refer to the `Topology File` section. A simplest command line to start a simulation must be as:

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

Command Line Options

With `$PATH/DTS -help` or `$PATH/DTS -h`, all the command line options can be seen.

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 string for labeling a specific run
`-ndx`	string	`Index.inx`	Index file name
`-restart`	string	`NO`	restart file name
`-nt`	int	1/all	number of threads for OpenMP

Table 1: Command Line Options for FreeDTS.

dts file and input parameters and option

The input file must have a `.dts` extension. Below are the most common options specified in this file, though it may also contain additional details such as inclusions and their interactions. Additional options are explained in their respective sections.

Integrator Type

This will define which kind of integration scheme should be used.

```
Integrator_Type = MC_Simulation
```

Default integrator `MC_Simulation`

Standard simulation input parameters

Simulation steps

```
; Set_Steps = initial_step final_step
 Set_Steps = 1 5000
```

In the Temperature section, the parameters β and $\Delta\beta$ are defined. A higher value of $\Delta\beta$ corresponds to a lower temperature. The parameter $\Delta\beta$ introduces a constant shift, altering the acceptance probability of metropolis algorithm by a fixed amount. This shift drives the simulation out of equilibrium, i.e., $(P_{acc} = exp(\beta\Delta E + \Delta\beta)$

```
;Temperature  = beta delta_beta
 Temperature = 1 0
```

Each vertex in the membrane model represents either a surface element (if the vertex is part of the surface) or a curve element (if the vertex lies on an open edge of the triangulated surface). In version 2, the surface no longer needs to be closed. Algorithms are also provided to convert surface vertices to edge vertices as needed.

For each vertex, specific mechanical parameters must be defined:

Kappa: Defines the bending rigidity, Gaussian modulus, and spontaneous curvature.

```
 ;Kappa = kappa kappa_g C_0
 Kappa = 20 0 0
```

Edge_Parameter: Sets parameters for line tension, geodesic curvature, and normal curvature constants.

```
Edge_Parameters = 0 0 0
```

VertexArea: Couples each vertex to a stretching energy and requires four values: the first two apply to surface vertices ($e_s = \frac{k_a}{2}(a_\nu - a_0)^2$), while the second two apply to edge vertices ($e_s = \frac{k_e}{2}(l_\nu - l_0)^2$). Where $a_0 = (1+2a)\frac{\sqrt{3}}{2}$ and $l_0 = \sqrt{1+2b}$

```
 ;VertexArea = ka a kl b
 VertexArea = 0 0 0 0
```

Frequency of centering the mesh inside the box

```
 Box_Centering_F = 0
```

Mesh constraint during simulations: It is recommended to use these values (they are also the default) unless different settings are needed for a specific purpose.

```
 ;Min_Max_Lenghts = a_min^2 a_max^2
 Min_Max_Lenghts = 1 3
 ;MinfaceAngle = min_angle
 MinfaceAngle = -0.5
```

Vertex Position Update Schemes

Initialising and defining an algorithm for updating vertex positions. Note that in FreeDTS, there are two types of vertices: those on the surface of the triangulated structure and those on the edges.

Options

```
;VertexPositionIntegrator = MetropolisAlgorithm x y R
VertexPositionIntegrator = MetropolisAlgorithm 1 1 0.05
```

A serial algorithm that updates vertices using the Metropolis Algorithm. In each step, `x` represents the percentage of surface vertices to be updated, `y` represents the percentage of edge vertices to be updated, and `R` specifies the maximum displacement allowed for a vertex. **This is the Default algorithm.**

```
;VertexPositionIntegrator = MetropolisAlgorithmOpenMP x y R
VertexPositionIntegrator = MetropolisAlgorithmOpenMP 1 1 0.05
```

A Multithreading algorithm (using OpenMP) that updates vertices using the Metropolis Algorithm. In each step, `x` represents the percentage of surface vertices to be updated, `y` represents the percentage of edge vertices to be updated, and `R` specifies the maximum displacement allowed for a vertex.

Edge Flip (Alexander Move)

Initialising and defining an algorithm for edge flipping.

Options

```
;AlexanderMove = MetropolisAlgorithm x
AlexanderMove = MetropolisAlgorithm 1
```

A serial algorithm that flips the edges using the Metropolis Algorithm. In each step, `x` represents the percentage of edges (links) to be attempted to flip. **This is the Default algorithm.**

```
;AlexanderMove = MetropolisAlgorithm x
AlexanderMove = MetropolisAlgorithmOpenMP 1
```

A Multithreading algorithm (using OpenMP) that flips the edges using the Metropolis Algorithm. In each step, `x` represents the percentage of edges (links) to be attempted to flip. **This is the Default algorithm.**

Inclusion Pose Updates

Initializing and defining an algorithm for inclusion (protein) pose, which includes both Kawasaki moves and changes in the orientation of inclusions.

```
;InclusionPoseIntegrator = MetropolisAlgorithm   x y
InclusionPoseIntegrator = MetropolisAlgorithm   1 1
```

A serial algorithm that updates protein pose using the Metropolis Algorithm. In each step, `x` represents the percentage of inclusions attempting Kawasaki moves, while `y` represents the percentage of inclusions attempting orientation changes. **This is the Default algorithm.**

Box Size Update

Box Size Update In FreeDTS, by default, the simulation box remains constant. However, it can be coupled with dynamic box algorithms to capture various physical behaviors. These dynamic box algorithms are meaningful only when the triangulated surface is periodic in at least one direction.
Single thread isotropic coupling to a frame tension

```
;Dynamic_Box =  IsotropicFrameTension tau tension XY
Dynamic_Box =  IsotropicFrameTension 5 3 XY
```

Multithread isotropic coupling to a frame tension

```
;Dynamic_Box =  IsotropicFrameTensionOpenMP tau tension XY
Dynamic_Box =  IsotropicFrameTensionOpenMP 5 3 XY
```

Single thread anisotropic coupling to a frame tension

```
;Dynamic_Box =  AnisotropicFrameTension tau f1 f2 f3 XY
Dynamic_Box =  AnisotropicFrameTension 5 3 1 0 XY
```

A harmonic potential can be used to maintain the box size around a specific value while allowing it to fluctuate within the bounds of the potential.

```
;Dynamic_Box =  HarmonicPotential tau K A0 XY
Dynamic_Box =  HarmonicPotential  5 100 10000 XY
```

Coupling Globalvariables

volume coupling using polynomial potentials. The energy will be coupled to

$$E_V = -\Delta PV + K/2(v-\gamma)^2$$

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

```
VolumeCoupling = SecondOrder delta_p K target_v
```

Osmotic pressure: Based on the Jacobus van 't Hoff equation $\Pi = icRT$

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

```
VolumeCoupling = OsmoticPressure gamma P0
```

total area coupling: The system energy will be coupled to a function as below:

$$E_A = \frac{K}{2} \times N_T \times (\frac{A}{A_0} - 1)^2$$

where

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

```
TotalAreaCoupling = HarmonicPotential K gamma
```

global curvature coupling: This will couple the system energy to a function as

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

```
GlobalCurvatureCoupling = HarmonicPotential K C_g0
```

Force between two groups

A harmonic potential will be inserted on two groups. Each group consist of a list of vertices that it should be defined in an index file.

```
ConstraintBetweenGroups = HarmonicPotentialBetweenTwoGroups 100 10 G1 G2 0 0 1
```

External Field On Vector Fields

```
ExternalFieldOnVectorFields = ConstantFieldOnVectorFields 20 1  0  0  20 1 1 0
```

Boundary conditions

FreeDTS also allows the simulation of membranes in confined spaces, where part of the space is excluded. Four types of confined spaces are defined in FreeDTS, such as TwoFlatParallelWall. To apply one of these confinement types, one of the commands below must be added to the input file.

```
;Boundary = TwoFlatParallelWall thickness direction
Boundary = TwoFlatParallelWall 2 Z
```

```
;Boundary = EllipsoidalShell thickness R a b c
Boundary = EllipsoidalShell 2 10 1 1 1
```

```
;Boundary = EllipsoidalCore R a b c
Boundary = EllipsoidalCore 10 1 1 1
```

Output management

```
VisualizationFormat = VTUFileFormat VTU_F 100
NonbinaryTrajectory = TSI TrajTSI 100
```

Surface Boundary Edge Update

```
OpenEdgeEvolution =   EvolutionWithConstantVertex 1 1
```

Dynamic Topology

```
DynamicTopology = Three_Edge_Scission 1
```

Energy Calculation method

```
 EnergyMethod = FreeDTS1.0_FF
```

Additional forces

```
VectorFieldsForceOnVertex = Constant_NematicForce 2 2
```

Non equilibrium Commands

Nonequilibrium Commands are small functions in **FreeDTS** that accept inputs, and changing specific simulation parameters. Users can include as many Nonequilibrium Commands as desired. Currently, **FreeDTS** has the following list of commands. Each command receives an integer (indicating the frequency, in steps, at which the command is executed) and a value representing the rate of the change. Please note, developing a new Nonequilibrium command is rather easy and straight forward.

- Changing Temperature, in principle *Beta (1/kT)*

```
NonequilibriumCommands ChangeTemperatureConstantRate 100 0.1
```

- Changing the equilibrium distance of the Harmonic Potential that is defined between two groups. Note, only valid if the `HarmonicPotentialBetweenTwoGroups` is applied (see above).

```
NonequilibriumCommands IncrementHarmonicPotentialBetweenTwoGroups 100 0.1
```

- Changing the targeted volume in `SecondOrder` algorithm for volume coupling. Note, only valid if the `SecondOrder` is applied (see above).

```
NonequilibriumCommands IncrementVolumeCouplingSecondOrder 100 0.1
```

- Changing the radius of the corresponding rigid wall (see above).

```
NonequilibriumCommands ExpandEllipsoidalCoreWall 100 0.1
```

- Changing the thickness of the corresponding rigid wall (see above).

```
NonequilibriumCommands ThinningEllipsoidalShell 100 0.1
```

Inclusion (proteins and vector fields)

```
INCLUSION
Define 2 Inclusions
SRotation   Type   K   KG  KP  KL  C0     C0P  C0L
1           Pro1   0   0   0   0   0.0   0   0    0   0  0  0.0
2           Pro2   0  0   0   0   0.0   0   0    0    0  0  0.0
```

Generating inclusions

```
GenerateInclusions
Selection_Type Random
TypeID       2
Density      0.1
```

Inclusion Inclusion Interactions

```
Inclusion-Inclusion-Int
1    1    1   2  0   -10
```

Topology File Overview

The topology file provides information about the positions of all vertices and how they are linked. There are two types of files that you can provide as a topology file: top and tsi file formats.

top File

Extension: `.top` Description: Contains a list of TS files in the `q` (see below for its format) format. Advantages: Allows multiple TS files to be fed into the system. Disadvantages: Does not include inclusions information.

tsi File

Extension: `.tsi` Description: A single file that represents a single frame of FreeDTS trajectories. Advantages: Includes inclusions information. TS File Formats TS Files Triangulated surface files that can be read by FreeDTS can be in either the q format or the tsi format. The Generate script can create files in both formats (see the Generate script section).

tsi Format Files

The following shows a part of a .tsi file with all necessary keywords highlighted in bold. Each .tsi file starts with a line specifying version 1.1. The next line defines the box size (x, y, and z) of the system in nm. The subsequent three sections describe the TS mesh. Each section starts with a keyword (vertex, triangle, and inclusion) and their corresponding count.

- Vertices: The file includes 130 vertices, each with an index and a position in x, y, and z.
- Triangles: The 130 vertices are connected via 256 triangles. Each triangle has an index and is defined by the vertices it connects. For example, triangle 0 connects vertices 11, 55, and 43.
- vector_fields
- Inclusions: A .tsi file can include a section for (protein) inclusions. In this example, there are three inclusions of two different types. Each inclusion has an index followed by the inclusion type (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 (the sum of both numbers must be one) defining the orientation of the inclusion with respect to the bilayer normal. The Generate script can produce these files, ensuring compatibility and ease of use within the FreeDTS system.

```
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 `.q` format files are another way to represent triangulated surfaces in FreeDTS. Below is a detailed description of the `.q` format structure:

- Line 1: Box information (3 double numbers).
- Line 2: Number of vertices (1 integer number; let's call it NV).
- Lines 3 to NV+2: Vertex ID and coordinates (1 integer number and 3 double numbers per line).
- Line NV+3: Number of triangles (1 integer number; let's call it NT).
- Lines NV+4 to NV+NT+3: Triangle ID and IDs of its vertices (4 integer numbers per line).

```
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
```

Index file

An example of index file as follows.

```
Group1  1
673
Group2  1967
0   1   2   3   4   5   6   7   8   9   10   11
```

The first string is the group name and the second is number of the vertices within the group.

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`

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`