Implementation of object-in-fluid in ESPResSo

Ivan Cimrák*, Markus Gusenbauer†

* Department of Software Technologies, University of Žilina, Slovakia
† St. Poelten University of Applied Sciences, Austria

1 Object-in-fluid in 5 minutes

The following lines are from the official webpage of ESPResSo:

> *"ESPResSo is a highly versatile software package for performing and analyzing scientific Molecular Dynamics many-particle simulations of coarse-grained atomistic or bead-spring models as they are used in soft-matter research in physics, chemistry and molecular biology. It can be used to simulate systems such as polymers, liquid crystals, colloids, ferrofluids and biological systems, for example DNA and lipid membranes."*

These lines tell the user exactly, what ESPResSo is aimed for. Molecular simulations. Very briefly: given the points in the space, and given the forces between them (and how they depend on many parameters), EPSResSo can compute how these points move in space (and much more, of course).

Probably all current simulations using ESPResSo, work with objects (molecules, atoms, polymers, colloids, crystals, ...) that are physicaly composed of points linked together with bonds. These objects are like skeletons, without inner or outer volume.

Our idea is to use ESPResSo for objects that do have inner volme, for example red blood cells, magnetic beads, capsules, ... In fact, it is easy: the boundary of an object (for example a red blood cell) is covered with triangular mesh. The vertices of the mesh are put into ESPResSo as particles. The edges of the mesh will define elastic forces keeping the shape of the red blood cell. The movement of the red blood cell will be achieved by adding forces to the mesh points and voilá, we have moving object simulated in ESPResSo.

2 Objects in a fluid

Modelling of the flow of a fluid with immersed elastic or rigid objects is a challenging task. The fluid interacts with an elastic object resulting in its deformation; this imediately generates forces acting back on the fluid. The aim is to describe the immersed object using the notion of particles, and to create bonds between these particles representing elastic or rigid forces. Such an object is put in a Lattice-Boltzman flow.

We consider objects composed of a membrane encapsulating the fluid inside the object. For now, the inside fluid must have the same density and viscosity as the outside fluid. (See Section 4.1 with unresolved issues.) The object will be

represented by its membrane (boundary) and the membrane will be discretized using a triangulation. Such a triangulation will define particles distributed on the surface of the immersed object. Next we define different bonded interactions:

- between two particles, corresponding to the edges in the triangulation (modelling the stretching of the membrane),
- between three particles, corresponding to the triangles of the triangulation (local area, or local surface preservation of the membrane),
- between four particles, corresponding to two triangles from the triangulation sharing a common edge (bending of the membrane).

The object immersed in the fluid will move under the influence of the deforming forces, defined through the bonds, and under the influence of the fluid motion. The fluid-particle interaction is described in the user guide of ESPResSo. We use the same approach. This interaction is based on the frictional force between the fluid and the spherical particles. In our case however, we consider the movement of a larger immersed object and the particles are only a virtual discretization points on the surface of the object. Therefore the object will move in the flow only if there is a nonzero difference between the fluid velocity and the particle velocity. In other words, there has to be at least small flow through the membrane, which is in most cases unphysical. However, we believe that on larger scales, this unphysical flow through the membrane is negligible.

Other approach (See Section 4.2 with unresolved issues) is to use the no-slip condition on the boundary of the immersed object. In this case, the influence of the fluid on the immersed object is not transferred through the forces, but through the velocities of the particles on the membrane. This is completely different approach, however the flexibility of ESPResSo allows for implementing it. See unresolved issues.

Membranes

With our approach it is easy to model also elastic sheets, or free membranes that do not necessarily enclose a 3D object. In this case, we do not define area_force_global and volume_force interactions, since these two interactions are ment for closed immersed objects.

Parameters

There are several parameters involved in this model. All of them should be calibrated.

- Mass of the particles. Every particle has its mass and the dynamics is influenced by this parameter.
- Friction coefficient. The main parameter describing the fluid-particle interaction is the `friction` parameter from the ESPResSo command `lbfluid`.
- Parameters of elastic moduli. Elastic behaviour can be described by different eleastic moduli. We show five of them: stretching, bending, local and global area preservation and volume preservation. Each of them has its own scaling parameter, we denote them ks, kb, kal, kag, kv, respectively.

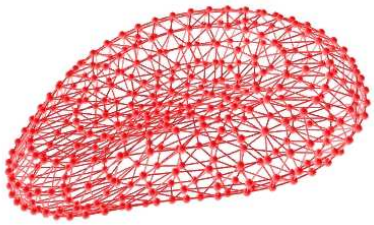

Figure 1: Triangular mesh representing the boundary of a RBC deformed in the fluid flow.

The mass of the particles and the friction coefficient can be calibrated using the drag coefficients of the ellipsoidal objects. These drag coefficients have known analytical values and the mass and friction can be calibrated to fit this values. More details about the calibration is in [1].

The elastic parameters are specific to the immersed onjects. They correspond to their physical values. More details about their mechanical and biological meaning is presented in [2] specifically for red blood cells. However, the proper calibration to fit the experimental data has been performed in [1].

2.1 Geometry

The membrane of the immersed object is triangulated. In Figure 1 you can see an example of such triangulation. Triangulation can be obtained using various software tools. Two files must be put in a directory INPUT: `mesh-nodes.dat` and `mesh-triangles.dat`. The parameters of the mesh are the number of particles on the surface of the immersed object, denoted by `nnode`, and the number of triangular faces in the triangulation, denoted by `ntriangle`.

The `mesh-nodes.dat` contains `nnode` lines with three real numbers separated by blank space, representing three coordinates of the corresponding particle. The membrane is thus discretized into `nnode` particles with IDs starting from 0 to `nnode-1`. The IDs are assigned in the same order as in the `mesh-nodes.dat` file.

The `mesh-triangles.dat` contains `ntriangle` lines with three nonnegative integers separated by blank space. Each line represents one triangle in the triangulation. For algorithmic purposes it is crucial to have defined a correct orientation of the triangle. We define the orientation using the normal vector associated with the triangle. The important rule is that the normal vector of the triangle must points inside the immersed object.

As an example, let us have one line in the file `mesh-triangles.dat` with numbers 4, 0 and 7. This means that particles with IDs 4, 0 and 7 form one triangular face of the triangulation. The orientation is defined as follows: create two vectors v_1 and v_2, such that v_1 is pointing from particle 4 to particle 0, and v_2 is pointing from particle 4 to particle 7. Be carefull, the order of vectors and particles matters!

The normal vector n is computed as a vector product $v_1 \times v_2$. The direction of n can be determined by the rule of right hand: the thumb points in the v_1 direction, the index finger in the v_2 direction and the middle finger in the

n direction. Following this principle, all the lines in the `mesh-triangles.dat` files must be such that the normal vectors of the corresponding triangles must point inside the immersed object.

These two files are sufficient to describe the geometry and topology of the triangulation. For the definition of bonded interactions the following geometric entities are necessary: position of the particles, edges, lengths of the edges, triangles, areas of triangles, angles between two triangles sharing a common edge, surface of the immersed object, volume of the immersed object. All these geometrical entities can be computed using the information from the files `mesh-nodes.dat` and `mesh-triangles.dat` and the computation is done in the script `bondGenerator`.

The script `bondGenerator` compiled from the C source `bondGenerator.c` reads both mesh files, generates list of edges, and computes all geometrical entities needed for definition of bonded interactions. It writes three types of files into directory `TMP`. First it writes file `createPart`, containing `nnode` lines with `part` commands. An example is as follows:

```
part 0 pos 3.0 3.0 6.0 type 1 mol 1 mass 1
```

Next it writes files `bondsStretching`, `bondsBending`, `bondsAreaLocal`, `bondsAreaGlobal`, `bondsVolume`. Each of these files contains information about interactions defining one of five elastic moduli. For example, in the file `bondsStretching` we have as many lines as there are the edges in the triangulation. Each line defines one interaction with its own interaction ID, identificator of the interaction type and parameters defining the properties of this interaction, e.g.

```
inter 106 stretching_force 4.6 5.0
```

Detailed description of the available types of interactions is presented in Section 2.2.

Further, `bondGenerator` writes files `partStretching`, `partBending`, `partAreaLocal`, `partAreaGlobal`, `partVolume`. In these files, the bonds between particles are defined. Each of these files contains lines with one bond definition, e.g.

```
part 313 bond 10006 293
```

Previousle described files generated by `bondGenerator` are directly included in the ESPResSo tcl script.

2.2 Interactions

The following interactions were implemented in order to mimic the mechanics of biological membranes. Their mathematical formulations have been taken from [3].

2.2.1 Stretching force

Syntax

| `inter` *bondid* `stretching_force` L^0_{AB} k_s

Description

This type of interaction is available for closed 3D immersed objects as well as for 2D sheet flowing in the 3D flow.

For each edge of the mesh, L_{AB} is the current distance between point A and point B. By L^0_{AB} we denote the distance between these points in the relaxed state, that is if the edge has the length exactly L^0_{AB} then no forces are added. ΔL_{AB} is the deviation from the relaxed state, that is $\Delta L_{AB} = L_{AB} - L^0_{AB}$. The stretching force between A and B is computed using

$$F_s(A,B) = k_s \kappa(\lambda_{AB}) \frac{\Delta L_{AB}}{L^0_{AB}} n_{AB}.$$

Here, n_{AB} is the unit vector pointing from A to B, k_s is the stretching constant, $\lambda_{AB} = L_{AB}/L^0_{AB}$, and κ is a nonlinear function that resembles neo-Hookean behaviour

$$\kappa(\lambda_{AB}) = \frac{\lambda_{AB}^{0.5} + \lambda_{AB}^{-2.5}}{\lambda_{AB} + \lambda_{AB}^{-3}}.$$

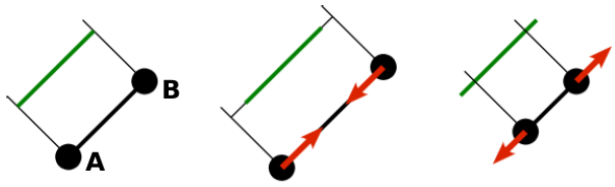

Figure 2: Stretching force between two points of the mesh surface.

The stretching force acts between two particles and is symmetric. Therefore if an interaction is defined by

```
inter 1 stretching_force 2.0 4.0
```

then the following two commands

```
part 42 bond 1 43
part 43 bond 1 42
```

are equivalent.

2.2.2 Bending force

Syntax

| `inter` *bondid* `bending_force` θ^0 k_b

Description

The tendency of an elastic object to maintain the resting shape is governed by prescribing the prefered angles between the neighbouring triangles of the mesh. This type of interaction is available for closed 3D immersed objects as well as for 2D sheet flowing in the 3D flow.

Denote by θ^0 the angle between two triangles in the resting shape. For closed immersed objects, you always have to set the inner angle. The deviation of this angle $\Delta\theta = \theta - \theta^0$ is computed and defines two bending forces for two triangles A_1BC and A_2BC

$$F_{bi}(A_iBC) = k_b \frac{\Delta\theta}{\theta^0} n_{A_iBC}.$$

Here, n_{A_iBC} is the unit normal vector to the triangle A_iBC. The force $F_{bi}(A_iBC)$ is assigned to the vertex not belonging to the common edge. The opposite force divided by two is assigned to the two vertices lying on the common edge. This procedure is done twice, for $i = 1$ and for $i = 2$.

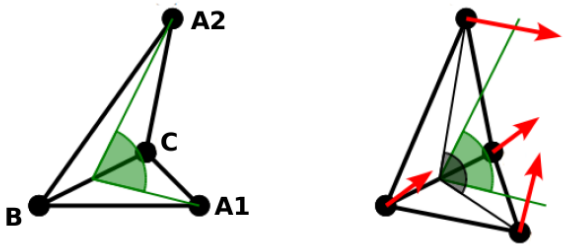

Figure 3: Bending force between two surface triangles.

Unlike the stretching force the bending force is strictly asymmetric. After creating an interaction

```
inter 33 bending_force 0.7 4.0
```

it is important how the bond is created. Particles need to be mentioned in the correct order. Command

```
part 0 bond 33 1 2 3
```

creates a bond related to the angle between the triangles 012 and 123. In Figure 3, the particle 0 corresponds to point A1, particle 1 to C, particle 2 to B and particle 3 to A2. There are two rules that need to be fulfilled:

- there has to be an edge between particles 1 and 2
- orientation of the triangle 012 must be correct, that is the normal vector defined as a vector product 01×02 must point to the inside of the immersed object.

Notice that also concave object can be defined. If θ_0 is larger than π, then the inner angle is concave.

2.2.3 Local area conservation

Syntax

| `inter` *bondid* `area_force_local` S^0_{ABC} k_{al}

Description

This interaction conserves the area of the triangles in the triangulation. This type of interaction is available for closed 3D immersed objects as well as for 2D sheet flowing in the 3D flow.

The deviation of the triangle surface S_{ABC} is computed from the triangle surface in the resting shape $\Delta S_{ABC} = S_{ABC} - S^0_{ABC}$. The area constraint assigns the following shrinking/expanding force to every vertex

$$F_{al}(A) = -k_{al}\frac{\Delta S_{ABC}}{S_{ABC}}w_A$$

where k_{al} is the area constraint coefficient, and w_A is the unit vector pointing from the centroid of triangle ABC to the vertex A. Similarly the analogical forces are assigned to B and C. This interaction is symmetric, therefore after defining the interaction

```
inter 44 area_force_local 0.02 4.0
```

the following commands are equivalent

```
part 0 bond 44 1 2
part 0 bond 44 2 1
part 1 bond 44 0 2
```

2.2.4 Global area conservation

Syntax

| `inter` *bondid* `area_force_global` S^0 k_{ag}

Description

This type of interaction is available solely for closed 3D immersed objects.

The conservation of local area is sometimes too restrictive. We add the global area constraint. Denote by S the current surface of the immersed object, by S_0 the surface in the relaxed state and define $\Delta S = S - S_0$. Then we define the global area conservation force

$$F_{ag}(A) = -k_{ag}\frac{\Delta S}{S}w_A$$

Here, the above mentioned force divided by 3 is added to all three particles.

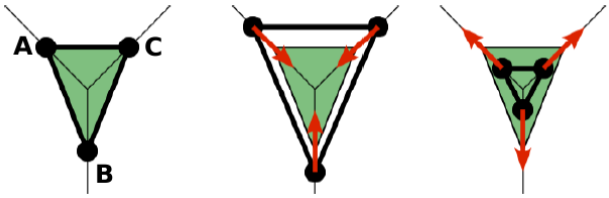

Figure 4: Local area force acting on a triangle of the mesh.

Again, this interaction is symmetric, as is the area_force_local.

2.2.5 Volume conservation

Syntax

| `inter` *bondid* `volume_forcel` V^0 k_v

Description
This type of interaction is available solely for closed 3D immersed objects.

The deviation of the global volume of the cell V is computed from the volume in the resting shape $\Delta V = V - V^0$. For each triangle the following force is computed

$$F_v(ABC) = -k_v \frac{\Delta V}{V^0} S_{ABC}\ n_{ABC}$$

where S_{ABC} is the area of triangle ABC, n_{ABC} is the normal unit vector of plane ABC, and k_v is the volume constraint coefficient. The volume of one immersed object is computed from

$$V = \sum_{ABC} S_{ABC}\ n_{ABC} \cdot h_{ABC}$$

where the sum is computed over all triangles of the mesh and h_{ABC} is the normal vector from the centroid of triangle ABC to any plane which does not cross the cell. The force $F_v(ABC)$ is equally distributed to all three vertices A, B, C.

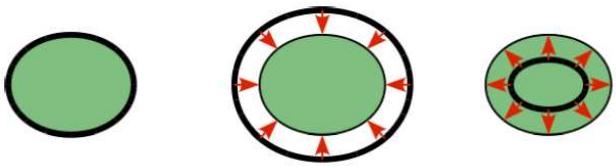

Figure 5: Volume force acting on the whole object.

This interaction is again non-symetric. After the definition of the interaction by

```
inter 22 volume_force 65.3 3.0
```

we need to take care about the order of vertices. By the following command we define the bond

```
part 0 bond 22 1 2
```

Triangle 012 must have correct orientation, that is the normal vector defined by a vector product 01×02 must point inside the immersed object.

3 Tcl example

To demonstrate the usage of our implementation we show a tcl script. The script is to large extent self-explanatory. As the immersed object we use tetrahedron and red blood cell. In the `INPUT` directory, both triangulations are present.

In the beginning we set the filenames of the files used to store the information about the particles, interactions and bonds.

Then we set compulsory ESPResSo constants (`time_step`, `skin`, `box_l`). Next we set the constants ks, kb, kal, kag, kv linked to the dynamical behaviour of the immersed object. Further, geometrical parameters are set and other veriables, such as type ID, mol ID, ...

After this preparatory steps, the script `bondGenerator` is executed. This script generates text files and saves them into `TMP` directory. Subsequently, these text files are included into the tcl script by multiple calls of the tcl command `source`. This creates all the particles, interactions and bonds.

To finish setting up the system we define the cellsystem, the lbfluid, and we prepare the vmd connection.

In the main iteration loop we set the velocity of the fluid on the left side to a constant value. We run the `integrate` command and repeat the main loop.

```
# the filenames must be set

set createPart TMP/createPart;
set fileNodes  INPUT/TETRAmesh-nodes.dat;
set fileTriangles INPUT/TETRAmesh-triangles.dat;
                     # for example with an red blood cell
                     # use CELLmesh-nodes.dat
                     # and CELLmesh-triangles.dat

set bondS TMP/bondsStretching
set bondB TMP/bondsBending
set bondAlocal TMP/bondsAreaLocal
set bondAglobal TMP/bondsAreaGlobal
set bondV TMP/bondsVolume
set bondVA TMP/bondsVolumeAreaGlobal
set partS TMP/partStretching
set partB TMP/partBending
set partAlocal TMP/partAreaLocal
set partAglobal TMP/partAreaGlobal
set partV TMP/partVolume
set partVA TMP/partVolumeAreaGlobal
set vmd "y"




setmd time_step 0.1;    # timestep for the integrator
setmd skin 0.2;
thermostat off;
setmd box_l 100 20 20;  # rectangular channel is defined


set ks 0.05;          # stretching of the cell
set kb 0.01;          # bending
set kal 0.01;         # local area preservation
set kag 0.01;         # global area preservation
set kv 10.;           # volume preservation

set massPart 1.0;     # setting the mass of particles
```

```
set nnode 4;          # number of IB points on the surface
                      # of the immersed object
                      # for example with an red blood cell
                      # set 400
set ntriangle 4;      # number of triangles at the surface
                      # of the immersed object
                      # for example with an red blood cell
                      # set 796

set originX 10;       # the initial coordinates of the center
set originY 10;       # of the immersed object
set originZ 10;

set stretchX 1.0;     # the immersed object will be scaled
set stretchY 1.0;     # by these factors in corresponding directions
set stretchZ 1.0;

set rotateX 0.;       # Rotation by specified angles around
set rotateY 0.;       # X axis, Y axis and Z axis. Angles given
set rotateZ 0.;       # in radians. rotateX=Pi/2 rotates the object
                      # by 90 degrees with the axis of rotation x
                      # such that vector 0,1,0 changes to 0,0,1 and
                      # 0,0,1 changes to 0,-1,0

set typePart 0;       # each immersed object must have
set molPart 0;        # different type and mol ID

set firstBondId 0;    # when adding more immersed objects,
set firstPartId 0;    # these parameters need to be set properly


# the following script generates the source files
# for particles, interactions and bonds

exec ./bondGenerator $fileNodes $fileTriangles $nnode \
$ntriangle $rotateX $rotateY $rotateZ $originX $originY $originZ \
$stretchX $stretchY $stretchZ $createPart$molPart \
$typePart $molPart $firstPartId  $bondS$molPart \
$bondB$molPart $bondAlocal$molPart $bondAglobal$molPart \
$bondV$molPart $bondVA$molPart $partS$molPart $partB$molPart \
$partAlocal$molPart $partAglobal$molPart $partV$molPart $partVA$molPart\
$ks $kb $kal $kag $kv $firstBondId $massPart;

# create particles and bonds from source files

source $createPart$molPart;
source $bondS$molPart;
source $partS$molPart;
source $bondAlocal$molPart;
source $partAlocal$molPart;
```

```
source $bondAglobal$molPart;
source $partAglobal$molPart;
source $bondB$molPart;
source $partB$molPart;
source $bondV$molPart;
source $partV$molPart;

cellsystem domain_decomposition -no_verlet_list;
lbfluid grid 1 dens 1.0 visc 1.5 tau 0.1 friction 0.5;

set upscale 3.;      # for scaling in the visualization

if { $vmd == "y" } {
    prepare_vmd_connection simEspresso 3000 1 $upscale;
                      #visualization
    exec sleep 2
    imd positions $upscale
}


# main iteration loop

set cycle 0
while { $cycle<1000 } {
puts "$cycle";
    if { $vmd == "y"} { imd positions $upscale; };


  # setting the constant velocity
  # of the fluid on the left side of the md_box
  for { set i 0 } { $i < 1} { incr i } {
    for { set j 0 } { $j < 20 } { incr j } {
      for { set k 0 } { $k < 20 } { incr k } {
        lbnode $i $j $k set u 0.5 0.0 0.0;
      }
    }
  }
  integrate 1;
  incr cycle;
}
```

4 Unresolved issues

4.1 Variable viscosity

It would be great to implement the possibility to chose different density and viscosity of the fluid. We did some research in this direction, we also impemented an algorithm that can detect for every lbnode wheter it is located inside

an immersed object or not. The implementation however was done for serial computation only, I suppose. It was also not tested yet.

4.2 Different fluid-structure coupling

It is reasonable to try different approach in fluid-structure coupling. The currently implemented approach using the drag force between the fluid and spherical particles is in fact unphysical if one consideres movement of larger immersed object with particles being only a virtual sites on the membrane of that object. In this case, the immersed object moves only if there is a locally nonzero difference between the fluid velocity and the particle velocity. In other words, if there is a nonzero flow of the fluid through the membrane.

Other approach is in using the no-slip condition. Normally, fluid has zero velocity relative to the boundary. In other words, the local velocity of the fluid near the interface between fluid and structure is the same as the velocity of the particles. The interaction between the fluid and the immersed objects is not done by transfer of the forces but by transfer of the velocities.

It would be good to implement this in espresso and to compare both approaches.

4.3 Orientation of the triangles

It is possible to include a simple check whether all the triangles in the triangulation have correct orientation. This check can be included in the bondGenerator.c file.

In some case, also if the triangle files are not given such that every triangle has good orientation, it is possible to restore good orientation. If we know the point in space inside the immersed object, from which we can see all the boundary points, then we can automatically verify the orientation of the triangles. It is thus possible to implement a simple check for this.

References

[1] I. Cimrák, M. Gusenbauer, and T. Schrefl. Modelling and simulation of processes in microfluidic devices for biomedical applications. *Computers an Mathematics with Applications*. Doi:10.1016/j.camwa.2012.01.062.

[2] M. Dao, C.T. Lim, and S. Suresh. Mechanics of the human red blood cell deformed by optical tweezers. *J. Mech. Phys. Solids*, 51:2259–2280, 2003.

[3] M.M. Dupin, I. Halliday, C.M. Care, and L. Alboul. Modeling the flow of dense suspensions of deformable particles in three dimensions. *Phys Rev E Stat Nonlin Soft Matter Phys.*, 75:066707, 2007.