Elasticity of objects

A method to determine the elasticity of soft objects

We have two goals:

1. Compute the *local* elastic properties of objects
2. Compute the elastic moduli of objects that cannot be istantaneously well-approximated by ellipsoids

In both these cases we will first have to define a reference configuration.

The reference configuration

We start with a trajectory of m points. That is, with the evolution of the positions of m points. A definition of the reference configuration may be the collection of average positions of the point, computed over the whole trajectory.

If we are dealing with soft objects such as microgels or star polymers, there is the problem that these objects can freely diffuse and rotate. As a result, taking simple averages will not work. We thus resort to algorithms designed to align two configurations by minimising the sum of the relative distances between the same pair of points. This kind of issues are called point set registration in computer vision.

- If the points in the two configurations retain their identity (in other words, if there is *correspondence* between the two sets of points), then the problem is known as Wahba's problem and it can be straighforwardly solved by using singular value decomposition. Crosslinkers in a microgels do not change their identity over time and hence will require this type of algorithms.
- If there is no correspondence between the two sets of points (*i.e.* if the identity of the points changes from configuration to configuration), then there is no analytical solution and iterative methods have been devised. Currently I'm using the iterative closest point method to approximate a solution. Arms in a star polymer can change identity over time since they can freely diffuse around the central bead and hence will require this kind of algorithms.

Once we choose a method with which two configurations can be aligned, the reference configuration is computed by taking the first configuration as fixed, aligning every other configuration to it and then taking the averages over the point positions.

Evaluate the local elasticity

The basic idea is adapted from Ref. [1]. The algorithm can be applied to all the particles or to a subset. In any case we call N the number of particles we use.

We consider istantaneous configurations (ICs) composed by N points, each point m being identified by its position $\vec{r}^m = \left(r_x^m, r_y^m, r_z^m\right)$. The same point in the reference configuration (RC) has position $\vec{R}^m = \left(R_x^m, R_y^m, R_z^m\right)$.

Given the two points m and n, we define their relative distance in the RC and IC as $\vec{\Delta R}^{mn}$ and $\vec{\Delta r}^{mn}$, respectively. Note that prior to computing these values one has to aling the IC over the RC (with the methods discussed above). These two relative positions are then mapped onto each other by a unique linear transformation:

$$\vec{\Delta r}^{mn} = \hat{F}^m \cdot \vec{\Delta R}^{mn}$$

where $\hat{F}^m$ is the local deformation tensor, where*local* here means relative to point m. It is clear that for each point m we have N_m equations, where N_m is the number of "neighbouring" points of point m. In general the number, identity (and weight, as we will see) of the neighbours can be chosen at will. If we were considering a continuum then we would have N_m equations that would yield the same tensor $\hat{F}^m$. However, here we are dealing with a discrete system, and the system of equations will not be satisfied by a single tensor. We thus look for the single mapping that gives the same results in the least-squares sense. In other words, we want to obtain the deformation tensor $\hat{\bar{F}}^m$ that minimises the difference between the observed distances and the distances we would have if there was a single linear mapping. The latter quantity is just the sum of the squared errors weighted by the "importance" we want to assign to each neighbour n and hence it is defined as

$$\phi^m = \sum_{n=1}^{N_n} (\vec{\Delta r}^{mn} - \hat{\bar{F}}^m \cdot \vec{\Delta R}^{mn}) \cdot (\vec{\Delta r}^{mn} - \hat{\bar{F}}^m \cdot \vec{\Delta R}^{mn}) w(R^{mn}).$$

We added w_n because, in general, errors made to map distances between m and points n should be weighted less if they are farther apart. Note that we explicitly write that w depends only on the relative distance between n and m. This is in contrast with Ref. [1], where they simulate crystalline structures and therefore use the order of the neighbouring shell rather than the inter-point distance to weight the errors.

ϕ^m can be minimised in the least-squares sense by taking its derivative with respect to the components of $\hat{\bar{F}}^m$ and setting it to zero:

$$\frac{\partial \phi^m}{\partial \bar{F}_{ij}} = \sum_{n=1}^{N_n} \left(-2\Delta r_i^{mn} \Delta R_j^{mn} + 2\Delta R_j^{mn} \bar{F}_{ik}^m \Delta r_k^{mn} \right) w(R^{mn}) = 0$$

where in the last equality we use the Einstein's convention of summing up over the repeated index k. Solving the equation for the components of $\hat{\bar{F}}^m$ yields

$$\bar{F}_{ik}^m \sum_{n=1}^{N_n} \Delta R_k^{mn} \Delta R_j^{mn} w(R^{mn}) = \sum_{n=1}^{N_n} \Delta r_i^{mn} \Delta R_j^{mn}$$

which can be written in matrix form as

$$\hat{\bar{F}}^m \cdot \sum_{n=1}^{N_n} \vec{\Delta R}^{mn} \otimes \vec{\Delta R}^{mn} w(R^{mn}) = \sum_{n=1}^{N_n} \vec{\Delta r}^{mn} \otimes \vec{\Delta R}^{mn}$$

where $\otimes$ is the outer product (*i.e.* $(\vec{R} \otimes \vec{r})_{ij} = R_i r_j$). Defining

$$\hat{A}^m = \sum_{n=1}^{N_n} \vec{\Delta r}^{\,mn} \otimes \vec{\Delta R}^{\,mn}$$

$$\hat{D}^m = \sum_{n=1}^{N_n} \vec{\Delta R}^{\,mn} \otimes \vec{\Delta R}^{\,mn} w(R^{mn})$$

we can write

$$\hat{\bar{F}}^m \cdot \hat{D}^m = \hat{A}^m$$

and therefore (provided $\hat{D}^m$ is not singular)

$$\hat{\bar{F}}^m = \hat{A}^m \cdot (\hat{D}^m)^{-1}.$$

We have thus obtained a deformation tensor that is *local*, since it is pertaining to one specific particle. From $\hat{\bar{F}}^m$ we can compute the Green-Lagrange strain tensor $\hat{C}^m$ and, from it, istantaneous values for $J^m = \sqrt{\det(\hat{C}^m)}$ and $I^m = \mathrm{Tr}(\hat{C}^m)/(J^m)^{2/3}$.

Evaluate the global elasticity

In order to compute the elastic moduli we need to find an expression for the *overall* deformation tensor. In order to do so we do a similar (albet simpler) construction than before. This is *not* taken by Ref. [1] but it's been derived by me, which means that it is more likely that there are errors. Be aware!

The idea here is that the deformation of the object is *homogeneous*, so that the local deformation gradient is equal to the global one. **This is the crucial approximation that we make**, and every time we are considering whether we want to apply this method or not we need to evaluate how realistic this approximation is.

First of all we define a point with respect to which we evaluate the deformations. For star polymers this is the central bead, while for microgels this might be either a central monomer or the centre of mass of the structure. Let $\vec{O}$ be the position of this point. In this case instead of the inter-point distances we use the distances between each point and the centre. For point m this distance in the RC is $\vec{\Delta R}^{\,m} = \vec{R}^m - \vec{O}$ and $\vec{\Delta r}^{\,m} = \vec{r}^m - \vec{O}$ in the IC. However, we now want to estimate the *global* deformation tensor $\hat{F}$ rather than the local one. In the continuum this quantity would satisfy the following relation for each m:

$$\vec{\Delta r}^{\,m} = \hat{F} \cdot \vec{\Delta R}^{\,m}.$$

Since our system is discrete we know that no single transformation will satisfy all these equations. We therefore look for the deformation tensor that $\hat{\bar{F}}$ that minimises the error we are committing in the least-squares sense, which in this case is

$$\phi = \sum_{m=1}^{N} \left(\Delta r_i^m - \bar{F}_{ij} \cdot \Delta R_j^m\right) \cdot \left(\Delta r_i^m - \bar{F}_{ik} \cdot \Delta R_k^m\right)$$

where once again we use Einstein's convention (here with indices i, j and k). If we take the derivative with respect to $\bar{\bar{F}}$ components and set it to zero we find

$$\frac{\partial \phi}{\partial \bar{F}_{ij}} = 2\bar{F}_{ij} \sum_{m=1}^{N} \left((\Delta R_j^m)^2 - \Delta r_i^m \Delta R_j^m \right) =$$

from which we find

$$\bar{F}_{ij} = \frac{\sum_{m=1}^{N} \Delta r_i^m \Delta R_j^m}{\sum_{m=1}^{N} (\Delta R_j^m)^2}$$

Note that in the last two equations the i and j indices are on both sides of the equal sign, which means that we are not summing up over them (*i.e.* Einstein's convention does not apply).

As in the previous section, the tensor $\bar{\bar{F}}$ can be used to compute $\hat{C}$ and then I and J which, in this case, refer to the entire configuration rather than to the single points.

How to code it up

If, like me, you want to write a code that computes $\bar{\bar{F}}$, you may be interested in writing the previous relation in a matricial form so as to make it easier to evaluate it (for instance with `numpy`). This can be done by defining $\vec{S}^m \equiv \left((\Delta R_x^m)^2, (\Delta R_y^m)^2, (\Delta R_z^m)^2 \right)$ and then the two matrices

$$\hat{A} = \sum_{m=1}^{N} \vec{\Delta r}^{\,m} \otimes \vec{\Delta R}^{\,m}$$

$$\hat{D} = \sum_{m=1}^{N} \vec{I} \otimes \vec{S}^m$$

where $\vec{I} = (1, 1, 1)$, so that

$$\bar{F}_{ij} = \frac{A_{ij}}{D_{ij}}$$

which means that $\bar{\bar{F}}$ is the component-wise ratio between $\hat{A}$ and $\hat{D}$.

References

[1] P. M. Gullett, M. F. Horstemeyer, and H. Fang, *A deformation gradient tensor and strain tensors for atomistic simulations*, *Modelling Simul. Mater. Sci. Eng.* **16** 015001 (2007)