Manual for Python package particleShear asscociated with the publication An Injectable Meta-biomaterial: From Design and Simulation to In-vivo Shaping and Tissue induction

A. Béduer, F. Bonini, C. Verheyen, M. Genta, M. Martins, J. Brefie-Guth, J. Tratwal, A. Filippova, P. Burch, O. Naveiras, T. Braschler

Table of Content

Code location

The release of python package particleShear used in this CodeOcean capsule is archived at Zenodo: https://doi.org/10.5281/zenodo.4589212. The most recent source version is at Github (https://github.com/tbgitoo/particleShear).

The latest release of the particleShear package can be installed via the usual pip installer commands. Depending on your installation, this can be:

```
pip3 install particleShear
```
or
```
pip install particleShear
```
or
```
python3 -m pip install particleShear
```
or
```
python -m pip install particleShear
```

The minimum required version of Python is 3.5, you may want to check with python --version or python3 --version that indeed the python executable invoked has the correction version.

See also the quick install guide available in this capsule at

/code/Documentation/Simulation particleShear/Quick install.pdf

Manual for the Python package particleShear

Part 1: Mathematical model for simulation

1. Aim of Part I

This part of the documentation for particleShear provides the mathematical foundation.

2. Content

3. Simulation and stress tensor evaluation in granular media

3.1. Introduction to the physical framework of the simulation

Our aim in simulation is to evaluate the behavior of variously crosslinked microgel suspensions. Ignoring the effect of interstitial fluid viscosity – assumed to be small compared to the large elastic forces involved – we are facing a simulation of a granular medium[1,2].

Granular media consist of discrete, interacting particles, and are typically simulated as such[1,2]. The physical framework for simulation of frictionally interacting granular particles is well established[3]. For spherical particles, it involves torque-free forces and frictional forces generating rotational torques[3]. The effects of forces and torques follow Newtonian mechanics, with associated conservation laws for linear and angular momentum[4]. The resulting equation sets are well known[5] and, with minor details regarding various levels of approximation, similar between different simulations documented in the literature[5-7]. For the physical framework, we use mainly the terminology and equations provided by Otsuki et al,[7] as it also relates to microgel suspensions. In order to make the simulation stable at large deformation amplitudes, we implemented two specific changes: adaptation of the frictional torque to reflect particle compression (with the aim of improving conservation of angular momentum) and a non-linear contact law (with the aim of better approaching Hertzian 3D contact forces and also to avoid entanglement of permanently bound spheres at very large amplitude).

Details of the physical framework are given in Section 4. Calculation of the necessary simulation constants from physical material properties is provided in Section 5, followed by a short discussion on particle ensemble equilibration in Section 6.

3.2. Introduction to stress tensor evaluation

A rheometer quantifies shear forces as shear stress and shear deformation as shear strain[8]. Hence in addition to a physical framework enabling the simulation per se, we also need evaluation of the shear stress in response to shear deformation[7]. To do so, we evaluate the shear stress tensor and extract the relavant components for analysis.

Shear stress tensor evaluation in granular media is known to be challenging[1,6]. In standard continuum mechanics, stress tensors are generally symmetric[9]. Averaging the stress tensors over the volume of the granular particles should not change this fact[1]. Yet, there are various formulations for the averaged stress tensor[1,5,7], and while "the global result must be perfectly symmetrical"[1], this is far from guaranteed in practice.

When a non-symmetrical stress tensor is obtained in simulation, this casts serious doubt on the validity of the result as this should not happen on theoretical grounds[1,9]. As non-symmetry has been putatively linked to inertial effects[1,6], our intended large amplitude simulations should be particularly vulnerable to possible deviations. It was therefore our aim to rigorously test all used equations to ensure the demanded symmetric evaluation of the stress tensors[1,6].

All in all, by careful checking, and correction where appropriate, of assumptions and calculations, we do obtain symmetrical stress tensors (to within relative error levels associated with numerical calculation, on the order of 10^{-14} to 10^{-17}). The details are

given in Section 7. This achievement provides a high level of confidence even in the case of large-amplitude simulations.

4. Detailed description of the physical framework

4.1. Simulation layout

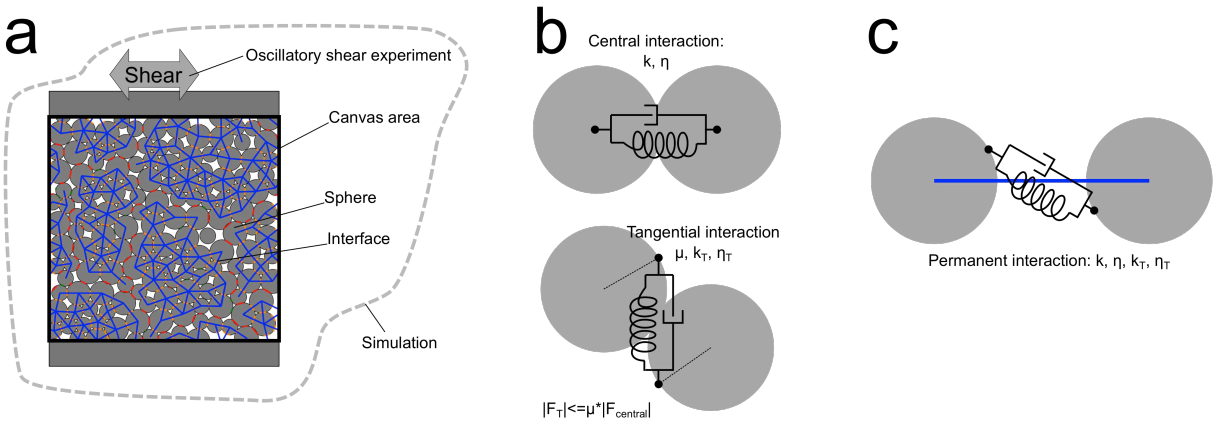

Figure S1-1: Design of the simulation. a) shows a screenshot of a simulation. Spheres (circles in 2D) are arranged on a canvas area. Geometrically touching spheres from transitory interfaces (red for locked, green for slipping); in addition, we implemented the possibility to specify permanent bonds (in blue with orange interface). b) interaction parameters for geometrically touching spheres. Otsuki et al.[7] specify linear viscoelastic interactions characterized by a central spring constant k and viscosity nu, as well as a tangential spring constant k_T and nu_T. Together with the friction coefficient mu, the central force $F_{central}$ sets an upper bound to the tangential force. Indeed, if the viscoelastic tangential force F_T calculated by the magnitude and rate of departure from the original contact point exceeds $mu*F_{central}$ in absolute value, the interface is considered slipping and transmits a maximum force of $mu*F_{central}$ (equations 4-6 of Otsuki et al.[7]). Also, $F_{central}$ is considered 0 if geometric contact is lost. c) In permanent interfaces, the viscoelastic forces calculated in central (with k, nu) and tangential direction (k_T, nu_T) are not bounded. As a consequence, $F_{central}$ can have a large attractive component when the spheres are sufficient elongated, and there is no applicable friction coefficient for $F_{tangential}$.

Fig. S1-1 shows the basic layout and principal parameters of the numerical simulation. To enable direct comparison, we choose to base our simulation as far as possible on the one by Otsuki et al.[7] for spherical microgel suspensions. We do however find a need to review some of the assumptions in [7], as well as to use a different framework for stress tensor evaluation[1], as detailed in the sections below.

We restrict ourselves to the 2D case, implying that the "spheres" are seen as 2D circles, and in reality correspond to infinitely high cylinders with given circular cross-sections. In reference to physical experiments typically performed on spherical microgel suspensions[10,11], we and others[7] use the term "sphere" also in context of 2D simulation. It is however understood that in this case, geometrically, "sphere" means a 2D circle equivalent to an infinitely high 3D cylinder. Through adaption of the force contact law to

better reflect the 3D case[12], we however approach the simulation closer to the one of true 3D spheres.

We implemented the simulation in a custom Python[13,14] package, to be able to extend it to permanently crosslinked particles.

Fig. S1-1a shows a screenshot of a typical simulation including permanent crosslinks. The spheres are in geometric contact and interact through central and tangential (frictional) forces[5,7]. As illustrated in Fig. S1-1b, the central force is characterized by an elastic component, described by a spring constant k, and a viscous component, described by an interaction viscosity η, and similar transversal constants k_T and η_T. The reader is referred to Otsuki et al.[7] for the basic contact and Newtonian kinematic framework, which we implement here with corrective changes (details in section 5).

To add permanent bonds to the simulation, we implement two changes as compared to the geometric contacts[7]. First, the bonds are implemented to be permanent and thus do not rupture regardless of the distance between the spheres. As consequence, large attractive forces can be produced by continued action of the spring constant k at larger separation. Second, there is no upper bound on the tangential force. As the permanent interfaces cannot slip, we integrate the departure from the tangential equilibrium over all time points rather than distinguishing sticking from slipping states.

To avoid crystallization we used a bimodal distribution with a factor of 1.4x between the smaller and the larger radii[7]. We further use shear-periodic Lees-Edwards boundary conditions[7,15].

4.2. Adaptions to large shear amplitude

In order to obtain meaningful simulations at large shear amplitudes (up to 200% deformation), we implemented specific important changes. This is independent of the permanent crosslinking, and concerns two main adjustments:

1. The torque compensation specified by Otsuki et al. [7] (eq. 8 in ref. [7]) provides only approximate conservation of angular momentum. We had to improve the conservation of angular momentum to obtain stable results at larger amplitudes and higher packing density.
2. Otsuki et al. [7] use a linear force law for compressive contact (spring constant k, see Fig. S1-1b). At high compression and shear, this leads to spheres "crossing" each other as the repulsion force does not rise high enough to prevent unphysical complete interpenetration. We added a non-linear term rapidly increasing at very small inter-center distances to prevent this and obtain stable results at high shear. We thus amended the contact law with the following requirements in mind: A) At very low deformation, the law should match the linear law proposed by Otsuki et al. [7]. B) At very high compression, the force should diverge to infinity to prevent complete merging of spheres. C) At intermediate deformations, the contact law should approach the known non-linear rising Hertzian contact law of 3D spherical objects[12].

The solution to these requirements are detailed in the next sections.

4.3. Conservation of angular momentum during frictional contact

In isolated systems without application of external forces, angular momentum is one of the major conserved properties. It is clear that a shear simulation as the one implemented here, but also the one by Otsuki et al.[7], does not correspond to such an isolated system. We do indeed apply and measure external forces. Nevertheless, the internal interactions should not generate angular momentum so that angular momentum change is strictly defined by the externally applied torques[4]. Otherwise, the system generates spurious torque on the shear plates, which will be interpreted as incorrect shear properties and thus G' and G'' values.

Specifically, the frictional interaction between two spheres in the interior of the simulation should not generate net angular momentum. We shall therefore analyze the conservation of angular momentum during a frictional interaction of two spheres.

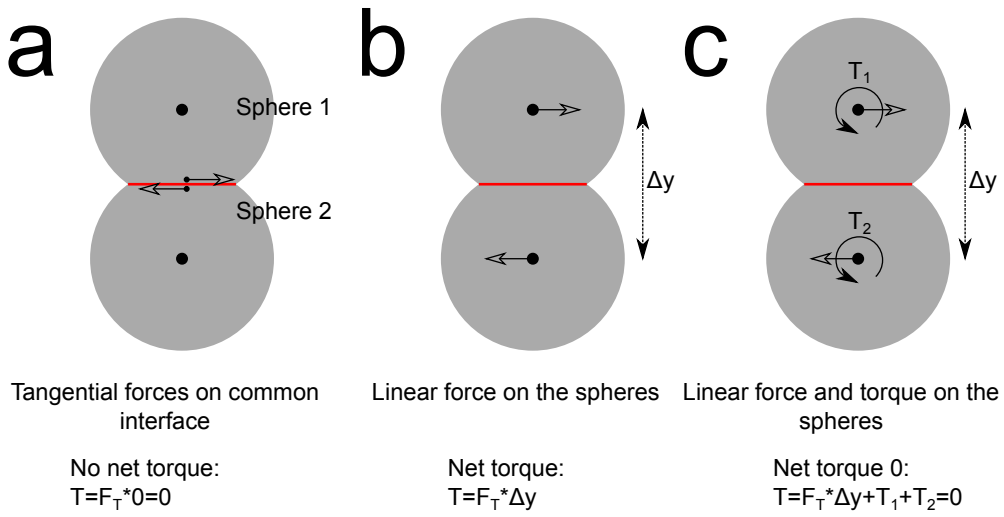

Figure S1-2. Conservation of angular momentum in pairwise frictional interaction. a) The tangential forces of magnitude F_T in the illustrated frictional interaction arise at the common interface. The torque of a force couple is given by force times separation distance, but here the separation distance is zero, so there is no net torque associated with this interaction. b) In the model, however, the tangential forces are reported to the spheres themselves, and thus modeled as acting on the centers. This leads to a net torque of $T=F_T*\Delta y$. c) In order to correctly conserve angular momentum, a net torque of zero is needed. To achieve this, torques T_1 and T_2 are applied on the spheres, and the angular momentum conservation imposes $T_1+T_2=- F_T*\Delta y$.

Fig. S1-2a shows two frictionally interacting spheres (in the actual 2D simulation, circles). There are many different configurations of linear speed and rotation that could give rise to the force couple shown in Fig. S1-2a; an example would be if sphere 2 in Fig. S1-2a were at rest and Sphere 1 spinning clockwise.

The frictional force couple acts tangentially and in opposite directions on sphere 1 and sphere 2 and thus naturally conserves linear momentum. Despite being a force couple, the tangential forces also conserve angular momentum since they arise at the common interface, without spatial separation. However, in the simulation, we need to calculate

the linear accelerations of the spheres. If no special precautions are taken, this is equivalent to the situation shown in Fig. S1-2b, resulting in a net torque of $F_T*\Delta y$ and thus loss of conservation of angular momentum.

Compensatory torques T_1 to sphere 1 and T_2 to sphere 2 are applied in granular media simulations to correct for this[5]. These additional torques are schematically shown in Fig. S1-2c. In the scenario where sphere 1 is rotating clockwise, while in frictional contact with the stationnary sphere 2, the forces and torques shown in Fig. S1-2c have an intuitive meaning : Upon touching sphere 2, the rotation of sphere 1 starts to slow down, while sphere 2 starts spinning in the opposite direction. In addition, sphere 1 starts to gain linear momentum by rolling on top of sphere 2, which is ejected behind.

The expression given by eq. 8 in Otsuki et al.[7] for the compensatory torques T_1 and T_2 is only approximate with regard to conservation of angular momentum. Indeed, the requirement of conservation of angular momentum imposes :

$$T_1+T_2=- F_T*\Delta y \qquad \text{eq. S1-1}$$

Where the torques T_1 and T_2, the tangential force F_T and the relevant inter-center distance Δy are all illustrated in Fig. S1-2. Using the expression given in eq. 8 of Otsuki et al. [7], one instead finds :

$$T_1+T_2=- F_T*(d_1+d_2)/2 \qquad \text{eq. S1-2}$$

where d_1 and d_2 are the diameters of sphere 1 and sphere 2 respectively. Eq. S1-1 and eq. S1-2 are equivalent if and only if $\Delta y=(d_1+d_2)/2$, that is, if the spheres are just touching without being compressed. This is nearly the case in the low packing densities simulated by Otsuki et al.[7], justifying the hard sphere approximation[5] in their case.

However, the error increases with increasing sphere compression and thus increasing packing density, and also transient compression in large-amplitude shear experiments. Hence, we needed a soft[5], rather than hard sphere[5] approximation.

The difficulty of the problem lies in knowing how the total torque of $- F_T*\Delta y$ in eq. S1-1 should be distributed to T_1 and T_2. This in turn depends on the exact position of the interface, since mechanistically, these torques are due to the non-central action of the frictional force with regard to the centers of the spheres. If the spheres have identical Young moduli and comparable diameters, it is reasonable to assume that they are compressed by the same relative extent. In that case, T_1 and T_2 should be distributed according to the relative diameters of the involved spheres. In the general notation with indexed spheres[7] (sphere *i* in contact with sphere *j*, distance between the centers r_{ij}), we therefore have:

$$T_i = - F_{T,ij}*r_{ij}*d_i/(d_i+d_j)=T_{i(Otsuki)}*2* r_{ij}/(d_i+d_j) \qquad \text{eq. S1-3}$$

Eq. S1-3 shows explicitly the difference with eq. 8 in Otsuki[7] : When the spheres are exactly touching, but not compressed, $2* r_{ij}/(d_i+d_j)=1$ and eq. S1-3 converges to eq. 8 in Otsuki[7]. With increasing compression, r_{ij} decreases and smaller torques need to be applied to compensate for the smaller separation between the centers of the two

spheres. In the case of permanent crosslinks, tangential forces are present also when r_{ij}> $(d_i+d_j)/2$, and in this case, larger torques result. At very large shear amplitudes, r_{ij} can be several times larger than $(d_i+d_j)/2$ for the permanent links, making the correction quantitatively important.

4.4. Non-linear central repulsion

A second adaptation to large shear and compression is the adoption of a non-linear repulsion contact law[12,16]. While we find the approach of a linear repulsion law[7] to be satisfactory at low shear amplitudes, problems arise at larger shear amplitudes and also at higher packing densities. The linear contact law foresees a finite maximum repulsion force of

$$|F_{max,linear}| = k*(d_i+d_j)/2 \qquad \text{eq. S1-4}$$

which is reached when two spheres completely merge to have identical center coordinates. A finite repulsion force at complete compression is unphysical, since we do not expect the spheres to be able to merge completely without a major rise in force (and, in reality, destruction of the spheres). Also, contact laws between curved surfaces, including cylindrical surfaces, are fundamentally non-linear at all but very small indentations[17].

A steep rise in force at very large compression is particularly important in the case of crosslinked particles. Indeed, in the presence of permanent bonds, “crossing” of the spheres through each other leads to irreversible entanglement. Empirically, this is not generally observed in suspensions made from distinct, crosslinked hydrogel particles.

Our aim was then to provide a contact law that is linear at small compression to comply as much as possible with the simulation set forth by Otsuki et al.[7] at low shear, but nevertheless avoids sphere merging and entanglement at high shear by an additional non-linear component.

Otsuki et al.[7] use a simple linear Hookean contact law[18]:

$$F_{elastic} = -k * \Delta x \qquad \text{eq. S1-5a}$$

To obtain arbitrarily large forces to prevent merging, we added a divergent term to this linear contact law, as follows:

$$F_{elastic} = -k * \Delta x *(1-2*\Delta x/(d_i+d_j))^{-1} \qquad \text{eq. S1-5b}$$

For small compressions, characterized by $\Delta x<<(d_i+d_j)/2$, the additional term approaches 1, and we recover the linear expression given in eq. S1-5a. For compression nearing completeness, Δx approaches $(d_i+d_j)/2$, and $(1-2*\Delta x/(d_i+d_j))^{-1}$ diverges to positive infinity. In practice, to avoid numerical errors when $\Delta x =(d_i+d_j)/2$, and also to limit artefacts due to very large forces that would require excessively short integration time steps, we impose an upper limit on $F_{elastic}$, typically

$| F_{elastic} |<= |F_{max,linear}|*1000= -k*(d_i+d_j)/2*1000$ eq. S1-6

We use the expression for the elastic force in eq. S1-5b in compression (Δx>=0). For the non-permanent links, no attractive forces are present when the spheres are beyond touching distance, and so $F_{elastic}=0$ for Δx<0. For permanently linked spheres, we use eq. S1-5b in compression (Δx>=0), but the original linear expression (eq. S1-5a) by Otsuki et al. [7] in stretch (Δx<0).

5. Model constants from physical properties

5.1. Linear approximation

The aim of this section is to link simulation constants k, k_T, η, and η_T to the physical properties of the spheres, particularly their Young modulus E.

The 2D simulation of the spheres corresponds to a contact geometry of 3D cylinders with parallel alignment. For this geometry, the linearized contact law for small deformations is[12] (eq. 5.34 in [12]):

$$\frac{F}{L} = \frac{\pi E}{8(1-\nu^2)} \cdot \Delta x \qquad \text{eq. S1-7}$$

where E is the Young modulus of the cylinder material, ν the Poisson ratio, and L the height of the cylinders. In a 2D simulation, the height of the cylinders is not specified and should be irrelevant for the intrinsic assembly properties such as the shear moduli, so for numerical purposes, we assume the extensive properties such as the force F to be given per m of depth. With this convention, the spring constant is (in units N/m per m of depth=Nm^{-2}):

$$k = \left(\frac{F}{L}\right)/\Delta x = \frac{\pi E}{8(1-\nu^2)} \approx \frac{\pi E}{8} \qquad \text{eq. S1-8}$$

where even for the worst and unlikely case of ν=0.5, there would be an error of no more than 25% for the approximation. With eq. S1-8, we can relate the spring constant k to the Young modulus of the constituent spheres. Eq. S1-8 is approximative for a number of reasons (2D geometry, small deformation limit, simplification of the Poisson ratio), but it nevertheless allows a realistic order-of- magnitude estimate of the spring constants to be used in the simulation. For the remaining constants (, we follow essentially the approach by Otsuki et al.[7] : $k_T = k$[7], as they describe the same material. Second, we obtain the viscosities $\eta= \eta_T$ from scaling considerations[7], with the difference that we use a lower value than Otsuki et al.[7] to better reflect the strongly elastic materials under study. We use $\eta= \eta_T=k*\tau*0.1$, rather than $\eta= \eta_T=k*\tau$ as in [7], with the characteristic time constant being related to both the spring constant and average particle mass : $\tau = \sqrt{m/k}$[7] :

$$\eta = \eta_T = 0.1 \cdot \sqrt{mk} \qquad \text{eq. S1-9}$$

5.2. Comparison with Hertzian contact law

In 3D, the contact law would be different. For simplicity, assuming identical radii of the spheres, the Hertzian contact law for a single contact is[12]:

$$F_{Hertz} = \frac{\sqrt{2}\cdot E}{3(1-\upsilon^2)} \cdot R^{1/2} \cdot \Delta x^{3/2} \qquad \text{eq. S1-10}$$

To obtain an estimate of the average contact law in 3D, the particularities of spherical packings in 3D have to be considered. The coordination numbers Z for soft particles in 3D vary with packing density, from a minimum of 6 ensuring minimal stability[19] to a maximum of 12 for spheres of identical size at close-packing[20]. At a packing density of θ =1 in 3D, one calculates a coordination number of about[19] :

$$Z = 2 \cdot D + 7.9 \cdot (\theta - 0.64)^{\frac{1}{2}} \approx 10.7 \qquad \text{eq. S1-11}$$

where D is the dimensionality (3 in 3D), and θ the nominal phase volume indicating packing density[10].

Considering Z/2 indidivual contacts randomly distributed on the surface of a half sphere, we obtain a total normal force :

$$F_{n,tot} = \frac{Z/2}{A_{tot}} \int F_n \mathrm{d}A = \frac{Z/2}{2\pi r^2} \int_{-\pi/2}^{\pi/2} \int_{-\pi/2}^{\pi/2} r^2 \cos\beta d\alpha \, d\beta \cdot \cos\beta \cos\alpha \cdot F_{Hertz} =$$

$$F_{n,tot} = \frac{Z F_{Hertz}}{4\pi} \int_{-\pi/2}^{\pi/2} \cos^2\beta \mathrm{d}\beta \int_{-\pi/2}^{\pi/2} \cos\alpha \, \mathrm{d}\alpha = \frac{Z F_{Hertz}}{4\pi} \cdot \frac{\pi}{2} \cdot 2 =$$

$$F_{n,tot} = \frac{Z}{4} \cdot F_{Hertz} \qquad \text{eq. S1-12}$$

Reported to the inter-layer height of $L = 2R/\sqrt{3}$ a hexagonal close-packing, this gives a scaled force of :

$$\frac{F}{L} = \frac{Z\sqrt{3}}{8R} \cdot F_{Hertz} = \frac{Z\sqrt{2/3}}{8R} \frac{E}{(1-\upsilon^2)} \cdot R^{-1/2} \cdot \Delta x^{3/2} \approx 1.1E \cdot R^{-1/2} \cdot \Delta x^{3/2} \qquad \text{eq. S1-13}$$

where the approximation was obtained by using Z=10.7 (eq. S1-11) and again neglecting the effect of the Poisson coefficien term $(1\text{-}\nu^2)$.

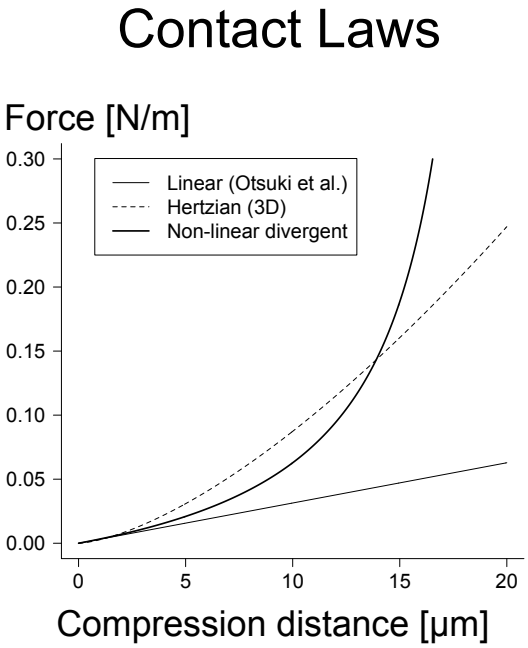

Fig. S1-3 : Comparison of the different contact laws. The linear expression by Otsuki et al.[7] is here evaluated with a spring constant as given by eq. S1-8. The Hertzian, 3D expression[12], is evaluated from eq. S1-13, whereas here we use a non-linear, divergent approach (eq. S1-5b) designed to avoid sphere interpenetration at very strong compression. Calculations for spheres of a radius of 10µm, a Young modulus of E=8kPa ; force reported to depth (linear and non-linear divergent) or pitch (Hertzian) as explained in the text.

Fig. S1-3 finally compares the forces in the three scenarios considered here : the linear expression used by Otsuki et al.[7] (eq. S1-5a, *k* given by eq. S1-8) ; the non-linear, divergent expression used here to avoid sphere interpenetration (eq. S1-5b, *k* again given by eq. S1-8) and the Hertzian contact law as given by eq. S1-13. In all cases, we used E=8kPa and r=10µm. Δx ranged from 0 (spheres in contact, uncompressed) to 2*r=20µm (total compression). The order of magnitude is similar for the three models at relatively small compression, i.e. when Δx is on the scale of a few microns, corresponding to 10-20% compression on an available distance of 2r=20µm. The 3D Hertzian model then departs from linearity, followed by the non-linear divergent law used here. At very high compressions, the non-linear divergent law provides the highest forces, as desired.

5.3. Damping

Given the suspension of the particles in a pore fluid (typically, deionized water or physiological saline), we expect some viscous drag as compare to a free powder. We do not simulate the precise mechanics of this, but do implement a damping coefficient. In shear, we expect there to be a gradient of flow speed in the interstitial fluid, and thus damping relative to this flow speed rather than to absolute v=0. This means that rather than damping the general momentum, the peculiar momentum relative to the expected local flow speed should be damped; this is typically implemented by using Sllod equations of motion[7]. Damping also improves the stability of the simulation, by strongly

reducing inertial phenomena such a sustained oscillation at particular resonance frequencies and propagation of compressive and shear sound waves. We typically adjust the damping coefficient such that during a full oscillatory shear cycle, a decay of about 50% of the speed relative to local shear flow would be observed.

6. Ensemble constitution and pre-equilibration

Pre-equilibration of particle ensembles for oscillatory shear rheology are a known practical challenge due to the dependency of material constants on shear history[8]. Similarly, in numerical simulation, both initial constitution and pre-equilibration can be challenging in light of obtaining repeatable results with random assemblies. We have combined here two approaches commonly employed, namely friction-less assembly[19] followed by a pre-shearing equilibration[21].

We constitute our primary particle assembly by random distribution spheres[19] of the desired size and number on a square 2D simulation canvas of dimensions *L*x*L*. The radii are calculated such that the desired packing fraction is achieved with a 1:1 bimodal distribution with a ratio of 1.4 between the smaller and larger radii[7]:

$$r = L\sqrt{\frac{1}{\theta N\pi(0.5+0.5/1.4^2)}} \qquad \text{eq. S1-14}$$

with L=500µm and N=150 for a typical simulation; the smaller radius would be r/1.4. We typically using packing densities θ between 1 and 2.

From a uniform random distribution, we let the spheres equilibrate under friction-less conditions, as this is considered to yield fully equilibrated assemblies[19]. The equilibration times are held long enough so that there are only minimal remaining oscillations (<10% of the anticipated amplitude during oscillatory shear) during a test baseline period prior to application of the actual shear protocol. We then set the desired friction coefficient, and possible inter-particle crosslinking (Fig. S1-1). After an additional equilibration period in this new configuration, we apply shear for one excitation period, before starting integration for the determination of the storage and loss moduli by demodulation of the force amplitudes.[7]

7. Evaluation of shear stress

In analogy to experimental oscillatory shear experiments, we evaluate the shear stress as the horizontal force (x-component) acting on the horizontal boundary elements (with vertical normal vector). For this, we first evaluate the average stress tensor[1] present in the material and extract the relevant shear force from it.

7.1. Stress tensor

Stress tensors describes the forces acting with in a material, as a function of the orientation of an observation plane element.[9] While stress tensor evaluation for continuous media is well established[9], it is still subject to some debate for granular, discrete materials[1,6]. Mechanical stress is indeed not distributed homogenously in such assemblies, and difficulties in averaging, but also in accounting for various inertial effects lead to approximate stress tensors. While stress tensors should be strictly symmetrical[1], more or less asymmetrical results are obtained if the inertial effects are not appropriately compensated for[1].

7.1.1. Formal requirements for stress tensor evaluation

Whatever the method and formulae employed for stress tensor evaluation, basic physical consideration impose some formal requirements. These formal requirements can be assessed for the analytical equations developed here or used from the literature. In addition, they lend themselves for unit testing to within the limits of numerical precision.

Translational invariance

The first such requirement is translational invariance. Stress tensors describe a state of matter regarding the internal forces present, and this should not depend on the position of an arbitrary observer[1].

Invariance with respect to the resting frame (Translation speed invariance)

The second requirement is at the heart of Newtonian (but not relativistic) mechanics: forces are related to acceleration, not linear speed, on so the linear speed of an arbitrary observer should not change the observed magnitude of the forces, and by extension the stress tensors.

Symmetry

Stress tensor symmetry in the continuous limit arises from a scaling argument ensuring finite rotational acceleration in the limit of vanishingly small mass elements[9]. This argument is of course limited by the existence of discrete molecules, atoms or elementary particles and thus the requirement for stress tensor symmetry is not as fundamental as the invariance requirements above. Nevertheless, given the enormous size scale difference between elementary particles and a typical rheometer, it seems reasonable to impose stress tensor symmetry in stress tensor evaluation.

7.1.2. Fundamental approaches to stress tensor evaluation in granular media

In stress tensor evaluation of granular media, there is a consensus that forces acting internally between particles need to be accounted for. This has given rise to the Love-Weber stress tensor $\langle \sigma_{LW} \rangle$[22,23]:

$$\langle \sigma_{LW} \rangle = \frac{1}{V} \sum \vec{F}_{int} \cdot \vec{l}_{int}^{T}$$ Love-Weber[22,23]

where $\vec{F}_{int}$ is the internal contact arising for a given particle-particle contact and $\vec{l}_{int}$ the separation vector of the centers of gravity of the involved particles and V the simulation volume.[1] Of note, using standard matrix multiplication rules, the product of a column vector such as $\vec{F}_{int}$ with a row vector such as $\vec{l}_{int}^{T}$ (the T denotes the transpose) gives rise to dyadic tensor matrix. The Love-Weber stress tensor matrix quantitatively describes the concept that the stress tensor reflects internal particle interactions; it is translationally invariant and invariant with respect to the choice of the resting frame as desired.

Unfortunately, the Love-Weber tensor is not strictly symmetrical in the presence of tangential forces.[1] This has been attributed to imperfect accounting of inertial effects[1,6], and we are aware of two main approaches of incorporating additional inertial effects in simulations of granular media.

Kinetic gas theory approach

The first approach, referred to as "dynamic stress"[5] is inspired by kinetic gas theory[5]. Momentum is indeed transported by moving particles, and just like for an ideal gas, one can associate this momentum with a pressure or more generally stress tensor[5,24]:

$$\langle \sigma_{\text{dynamic}} \rangle = \frac{1}{V} \sum m_p \, \vec{v} \cdot \vec{v}^{T}$$ Dynamic stress[5,24]

where summation is over the constituent particles, with m_p being the particle mass and $\vec{v}$ the displacement speed of each individual particles.

While such a definition of dynamic stress elegantly allows to address momentum transport[24], to us, it is rather problematic from a fundamental point of view.

By far the most important problem is that this definition of $\langle \sigma_{\text{dynamic}} \rangle$ violates the principle of invariance with regard to the choice of the resting frame. It is indeed trivial to see that for a simplistic system consisting of a single particle, any desired value of $\vec{v}$ can be reached by merely changing the linear speed of the frame of observation. Therefore, every possible dyadic product $\vec{v} \cdot \vec{v}^{T}$ can be obtained, allowing to produce at will every possible value $\langle \sigma_{\text{dynamic}} \rangle$ to within the limits of the symmetric dyadic product $\vec{v} \cdot \vec{v}^{T}$. This is in direct contradiction with the principle of independence from the free choice of the resting frame.

In their simulation, Otsuki et al.[7] use $\langle \sigma_{\text{dynamic}} \rangle$ to correct the Love-Weber tensor for inertial effects (eq. 15 in [7]). They partially address the problem of the resting frame by the introduction of a peculiar momentum: the momenta and therefore speeds are expressed relative to the expected local speed in shear flow[7]. With this relative definition, changing the resting frame does not change $\langle \sigma_{\text{dynamic}} \rangle$ anymore. The solution remains nevertheless partial: how should the expected local speed be defined?

Concretely, in a rheological shear cell, should the resting plane be the top plate, the lower plate, or maybe the middle plane? Less practically minded, what if the rheometer is moving?
Even more fundamentally, while invariance with respect to the choice of resting frame is achieved by the use of peculiar speeds and momenta[7], this necessitates the definition of an arbitrary zero speed and therefore merely shifts the problem from dependence on the resting frame to dependence on an arbitrarily chosen zero-speed. In an isolated system, the constant speed of the center of gravity might constitute a suitable zero-reference, but in our case where particles can leave and enter the simulation area, while external forces are applied, the choice of the zero-speed is indeed a difficult one.

In our setting of unit testing (SI 4), this difficulty is also revealed directly by the fact that $m_p \vec{v} \cdot \vec{v}^T$ is non-zero for a freely moving particle without applied external forces, where on reasonable physical grounds, one would expect $\langle \sigma \rangle = 0$.

A final practical issue is that $\langle \sigma_{\text{dynamic}} \rangle$ is by definition symmetric, and thus naturally unsuitable to correct asymmetry in the Love-Weber stress tensor.

Given our difficulties in unit testing, the inability of $\langle \sigma_{\text{dynamic}} \rangle$ to correct stress tensor asymmetry, and the undesirable dependence on an arbitrary zero-speed, we decided not to make use of the dynamic stress tensor.

Explicit correction for rotational movement and acceleration

A second approach consists in explicit removal of inertial forces linked to internal particle movement. This approach is thoroughly developed for rigid-body particles by Nicot et al.[1], and is found to result in correction terms arising from angular velocity and acceleration.[1]

We tested the solution proposed by Nicot et al.[1], but still found asymmetric stress tensors. In order to evaluate whether there were still fundamental terms lacking, or whether there was some error either in our implementation or the underlying framework, we developed inertial correction terms. In order not to directly replicate possible mistakes by Nicot et al.[1], we kept the derivation technique as independent as possible from the one by Nicot et al. [1] We further limited the calculations to the 2D circular particles at hand in order to avoid the difficulties of generality.

7.1.3. Stress tensor from external force

As a starting point, we use eq. 6 of [1], neglecting gravity:

$$\langle \sigma \rangle = -\frac{1}{V} \sum \vec{F}_{ext} \vec{r}^T + \frac{1}{V} \sum_{particles} \int_{particle} \rho \vec{a} \vec{r}^T \qquad \text{eq. S1-15}$$

where $\vec{F}_{ext}$ designates the forces acting across ensemble boundaries, $\vec{r}$ their point of action (approximated by the center of the particles concerned[1]), V the overall simulation volume (i.e. L^2 in 2D), ρ the density, $\vec{a}$ and the local acceleration at points within the particles.

With the choice of signs given in eq. S1-15, we follow the soil mechanics convention that associates compression rather than traction with positive normal stresses[6].

x^T denotes the transpose of x, and we use the product between a column vector and a row vector following usual matrix multiplication rules to denote the dyadic tensors as before.

The first term in eq. S1-15 is related only to the external forces:

$$\langle \sigma_{external\ force} \rangle = -\frac{1}{V} \sum \vec{F}_{ext} \vec{r}^T \qquad \text{eq. S1-16}$$

The acceleration $\frac{1}{V} \sum_{particles} \int_{particle} \rho \vec{a} \vec{r}^T$ term in eq. S1-15 can be decomposed into a linear acceleration part $\langle \sigma_{linear\ acceleration} \rangle$ reflecting the linear acceleration of the particle centers, and a spin part $\langle \sigma_{spin} \rangle$ reflecting the influence of local rotation on the stress tensor[1]:

$$\frac{1}{V} \sum_{particles} \int_{particle} \rho \vec{a} \vec{r}^T = \langle \sigma_{linear\ acceleration} \rangle + \langle \sigma_{spin} \rangle \text{ with}$$
$$\langle \sigma_{linear\ acceleration} \rangle = \frac{1}{V} \sum_{particles} m \vec{a}_g \vec{r}_g^T \qquad \text{eq. S1-17}$$

where the index $_g$ denotes the center of gravity for each particle.

7.1.4. Internal forces: Stress tensor from internal forces

We agree with Nicot et al.[1] that indeed eq. S1-15 can be rewritten by the use of the Love-Weber stress tensor[22,23] (eq. 28 of .[1]):

$$\langle \sigma \rangle = \langle \sigma_{LW} \rangle + \langle \sigma_{spin} \rangle = \frac{1}{V} \sum \vec{F}_{int} \vec{l}_{int}^T + \langle \sigma_{spin} \rangle \qquad \text{eq. S1-18}$$

where summation is over the internal contacts, counting each contact exactly once. If particles A and B are involved in the contact at hand, $\vec{F}_{int}$would designate the force excerted by particle A on particle B, and the contact vector would relate A to B. These definitions are again chosen to comply with the soil mechanics convention[6]. The term $\frac{1}{V} \sum \vec{F}_{int} \vec{l}_{int}^T$ is the well-known Love-Weber[22,23] expression for stress tensor evaluation:

$$\langle \sigma_{LW} \rangle = \frac{1}{V} \sum \vec{F}_{int} \vec{l}_{int}^T \approx \langle \sigma_{external\ force} \rangle + \langle \sigma_{linear\ acceleration} \rangle \qquad \text{eq. S1-19}$$

For purely central forces, the $\vec{F}_{int}$ and $\vec{l}_{int}$ vectors are collinear for each interaction and thus give rise to strictly symmetric dyadic products. The problem of stress tensor asymmetry can therefore indeed be pinned down not to inertial forces in general, but more specifically to tangential force vectors arising for instance by friction, where collinearity between $\vec{F}_{int}$ and $\vec{l}_{int}$ is lost and $\langle \sigma_{LW} \rangle$ becomes asymmetric[1].

7.1.5. Spin inertia terms

We therefore need to examine in detail the spin inertia terms. The spin acceleration term $\langle\sigma_{spin}\rangle$ is due to tensile stresses generated by the centrifugal forces on the one hand, and unbalanced torque leading to tangential rotational acceleration within the particles[1]:

$$\langle\sigma_{spin}\rangle =< \sigma_{\text{centrifgual}} > +< \sigma_{\text{tangential}} > \qquad \text{eq. S1-20}$$

Centrifugal component

Assuming rigid body motion, the centrifugal forces give rise to a diagonal, isotropic stress tensor contribution[1]. For the 2D geometry (cuts of long cylinders) chosen here, the centrifugal force for an infinitesimal volume element is given by:

$$\mathrm{d}F_{\text{centrifugal}} = \mathrm{d}V\rho \cdot \omega^2 r$$

where ρ is the mass density and ω the angular rotation rate. In steady state, the centrifugal force must be compensated by a corresponding stress tensor gradient. Due to the isotropic, diagonal form of the stress tensor in steady rotational motion[1], this is the equivalent of a shear-free, hydrostatic pressure gradient:

$$\frac{\mathrm{d}F_{\text{centrifugal}}}{\mathrm{d}V} = \rho \cdot \omega^2 r = \frac{\mathrm{d}P_{\text{centrifugal}}}{\mathrm{d}r}$$

and thus, by integration:

$$P_{\text{centrifugal}}(r) = \int_{R_0}^{r} \rho \cdot \omega^2 x \,\mathrm{d}x = \frac{\rho\omega^2}{2}(r^2 - R^2)$$

where the centrifugal pressure contribution was assumed to be zero at the particle surface ($P_{\text{centrifugal}}$=0 for r=R).

Averaging over the cylinder cross section leads to:

$$<P_{\text{centrifugal}} >= \frac{1}{\pi R^2}\int_0^R \frac{\rho\omega^2}{2}(r^2 - R^2)2\pi r dr = -\frac{\rho\omega^2 R^2}{4}$$

The stress tensor contribution is reported to relative to the overall simulation volume, and we obtain:

$$< \sigma_{\text{centrifgual}} >= -\frac{V_p}{V}\frac{\rho\omega^2 R^2}{4}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} = -\frac{\pi}{S^2}\frac{\rho\omega^2 R^4}{4}$$

where S is the width and height of our square simulation area, whereas $V_p = \pi R^2 L$ and $V = S^2 L$ are respectively the particle and total simulation volumes of height L.

The moment of inertia of a solid cylinder around its main axis is given by[25]:

$$I = m\frac{R^2}{2} = \frac{\pi L \rho R^4}{2}$$

The associated spin kinetic energy is given by[26]:

$$K = I \cdot \frac{\omega^2}{2} = \frac{\pi L \rho \omega^2 R^4}{4}$$

so that by analogy with the spherical case[1], we can write concisely:

$$< \sigma_{\text{centrifgual}} >= -\frac{K}{V}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \qquad \text{eq. S1-21}$$

the simulation volume V being again given by $V=S^2L$. One notices that the centrifugal stress tensor is symmetric, and is therefore unable to correct stress tensor asymmetry. We agree on this point with Nicot et al.[1] Eq. S1-21 however differs from the corresponding term in Nicot et al.[1] by a factor of 1.5 (see the coefficient $\frac{2}{3V}$ in front of the $K_s^p \delta_{\text{ij}}$ term in eq. 34 of [1], as compared to the $\frac{1}{V}$ coefficient in eq. S1-21; the sign is different due to the use of the soil mechanics convention[6] here and can be ignored).

For now, we find ourselves with a first difference compared to the development by Nicot et al. [1], albeit still without an explanation for our remaining asymmetric stress tensor terms.

Torque component

Last, but not least, we shall examine the rotational acceleration component of the stress tensor. A net remaining torque T on a given particle leads to rotational acceleration:

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} = \frac{T}{I}$$

where I is again the rotational moment of inertia around the appropriate axis.

This in turn leads to tangential acceleration of the mass elements:

$$\mathrm{d}F_{\text{tangential}} = \mathrm{d}V\rho \cdot \dot{\omega} r = \mathrm{d}V\rho \cdot \frac{T}{I} r$$

Explicit averaging of the fundamental stress tensor expression (dyadic product of force and point of attack vector relative to measurement volume[1]) for this case leads to:

$$< \sigma_{\text{tangential}} >= \frac{1}{V}\int \mathrm{d}\vec{F}_{\text{tangential}}\,\vec{r}^{T} = \frac{1}{V}\int \mathrm{d}V\rho \cdot \dot{\omega}\begin{pmatrix}-r\sin\alpha\\ r\cos\alpha\end{pmatrix}(r\cos\alpha \quad r\sin\alpha)$$

$$< \sigma_{\text{tangential}} >= \frac{\rho \cdot \dot{\omega}}{V}\int_{r=0}^{R}\int_{\alpha=0}^{\alpha=2\pi} 2\pi L r\mathrm{d}r \cdot r^2 \begin{pmatrix}-\sin\alpha\cos\alpha & -\sin^2\alpha\\ \cos^2\alpha & \sin\alpha\cos\alpha\end{pmatrix} d\alpha =$$

$$< \sigma_{\text{tangential}} >= \frac{\rho \cdot \dot{\omega}}{V} \begin{pmatrix} 0 & -1/2 \\ 1/2 & 0 \end{pmatrix} \int_{r=0}^{R} 2\pi L \mathrm{d}r \cdot r^3 =$$

$$< \sigma_{\text{tangential}} >= \frac{\rho \cdot \pi \dot{\omega} L R^4}{4V} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

with the definition of the moment of inertia ($I = \frac{\pi L \rho R^4}{2}$) as before[25], we obtain:

$$< \sigma_{\text{tangential}} >= \frac{T}{2V} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \qquad \text{eq. S1-22}$$

As for the spherical case[1], the net torque is equivalent to an anti-symmetric stress tensor component. The overall stress tensor can now be evaluated (eq. S1-18 to S1-22):

$$\langle \sigma \rangle = \frac{1}{V} \sum \vec{F}_{int} \vec{l}_{int}^{T} + \frac{1}{V} \sum \begin{pmatrix} -K & -T/2 \\ T/2 & -K \end{pmatrix} \qquad \text{eq. S1-23}$$

where T designates the total net torque acting on each particle, while K designates the spin kinetic energy of each particle[1]. By comparison to eq. 34 of [1], we again find a factor of 1.5.

7.2. Symmetry of the stress tensor

Eq. S1-23 differs from the one reported by Nicot et al.(eq. 34 of [1]) by a factor of 1.5 for the second right hand term. This difference has the potential to address the asymmetry of the stress tensor evaluation, since it also affects asymmetric off-diagonal terms.

First, we however have to address a question of interpretation arising from the form of eq. S1-23. The spin kinetic energy *K* is an internal property of each particle, but how should the torques be calculated? Should it be only internal torques, in analogy to the internal forces giving rise to the Love-Weber stress tensor? Or should it be total net torques acting on each particle, including torques transmitted across the system boundary (as one may implicitly assume from eq. S1-23)?

7.2.1. Model system of two frictional particles

Hence, we further check the general development leading to eq. S1-23 in a simple configuration: two particles interacting by friction. This allows to validate eq. S1-23 with respect to the factor 1.5 difference regarding the homologous expression in the literature .(eq. 34 of [1]) and possibly also to answer the question whether strictly internal or total torques should be taken into account.

In our system of two identical particles interacting by frictional forces, let us assume that the first particle is located on the origin, and the second particle at an arbitrary position:

$$\vec{r}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \vec{r}_2 = \begin{pmatrix} r_x \\ r_y \end{pmatrix}$$

Due to the translational invariance that stress tensor must display, this particular choice of the origin does not affect generality. Friction leads to a tangential couple with magnitude F_T perpendicular to the direction $\vec{r}_2$ (counterclockwise rotation, so positive F_T pointing along the direction $\begin{pmatrix} -r_y \\ r_x \end{pmatrix}$.) In the simulation, the tangential forces would be reported to the centers of the spheres (see Fig. S1-2), to that this would give rise to an asymmetric Love-Weber tensor (eq. S1-19):

$$\langle \sigma_{LW} \rangle = \frac{1}{V} \begin{pmatrix} F_T \cdot \left(-\frac{r_y}{|\vec{r}_2|} \right) \\ F_T \cdot \left(\frac{r_x}{|\vec{r}_2|} \right) \end{pmatrix} (r_x \quad r_y) = \frac{F_T}{V} \begin{pmatrix} -\frac{r_x r_y}{|\vec{r}_2|} & -\frac{r_y^2}{|\vec{r}_2|} \\ \frac{r_x^2}{|\vec{r}_2|} & \frac{r_x r_y}{|\vec{r}_2|} \end{pmatrix}$$

The only condition where $\langle \sigma_{LW} \rangle$ is symmetrical is if r_x and r_y are both equal to 0. This however leads to undefined fractions and is in any case physically impossible as it means that the two spheres occupy a single position in space. This confirms the well-known asymmetry of $\langle \sigma_{LW} \rangle$ [1].

Can the asymmetry of $\langle \sigma_{LW} \rangle$ be corrected by the inertial term proposed by eq. S1-22? The answer should be yes[1]. Conservation of angular moment implies that we have a total compensatory torque $T = -F_T |\vec{r}_2|$; the exact distribution of the total torque onto the two interacting particles is not important in this context. Disregarding $\langle \sigma_{\text{centrifugal}} \rangle$, which does not contribute to correction of stress tensor asymmetry, the overall stress tensor reads:

$$\langle \sigma \rangle = \langle \sigma_{LW} \rangle + \langle \sigma_{\text{tangential}} \rangle = \frac{F_T}{V} \begin{pmatrix} -\frac{r_x r_y}{|\vec{r}_2|} & -\frac{r_y^2}{|\vec{r}_2|} \\ \frac{r_x^2}{|\vec{r}_2|} & \frac{r_x r_y}{|\vec{r}_2|} \end{pmatrix} - \frac{F_T |\vec{r}_2|}{2V} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} =$$

$$\langle \sigma \rangle = \langle \sigma_{LW} \rangle + \langle \sigma_{\text{tangential}} \rangle = \frac{F_T |\vec{r}_2|}{V} \begin{pmatrix} -\frac{r_x r_y}{|\vec{r}_2|^2} & -\frac{r_y^2}{|\vec{r}_2|^2} + \frac{1}{2} \\ \frac{r_x^2}{|\vec{r}_2|^2} - \frac{1}{2} & \frac{r_x r_y}{|\vec{r}_2|^2} \end{pmatrix}$$

Since $\frac{r_y^2}{|\vec{r}_2|^2} + \frac{r_x^2}{|\vec{r}_2|^2} = 1$, we can further write:

$$\langle \sigma \rangle = \frac{F_T |\vec{r}_2|}{V} \begin{pmatrix} -\frac{r_x r_y}{|\vec{r}_2|^2} & -\left(1 - \frac{r_x^2}{|\vec{r}_2|^2}\right) + \frac{1}{2} \\ \frac{r_x^2}{|\vec{r}_2|^2} - \frac{1}{2} & \frac{r_x r_y}{|\vec{r}_2|^2} \end{pmatrix} = \frac{F_T |\vec{r}_2|}{V} \begin{pmatrix} -\frac{r_x r_y}{|\vec{r}_2|^2} & \frac{r_x^2}{|\vec{r}_2|^2} - \frac{1}{2} \\ \frac{r_x^2}{|\vec{r}_2|^2} - \frac{1}{2} & \frac{r_x r_y}{|\vec{r}_2|^2} \end{pmatrix}$$

indicating successful recovery of a symmetrical stress tensor as it ought to be. This development is not dependent on the exact particle geometry, suggesting that eq. S1-22 should be valid regardless of exact particle geometry.

For the simulation, many particle interactions take place. However, since the sum of symmetrical tensors remains symmetric, we can conclude that for as long as we include only the internally generated moments and forces into the calculation of the overall

stress tensor as defined by eq. S1-23, the result should be symmetrical. The Love-Weber expression takes explicit care not to include externally applied forces; the development here implies that the same should be done regarding the torques, only torques exchanged by the particles, but not torques delivered externally should be taken into account:

$$\langle\sigma\rangle = \frac{1}{V}\sum \vec{F}_{int}\,\vec{l}_{int}^{T} + \frac{1}{V}\sum \begin{pmatrix} -K & -T_{int}/2 \\ T_{int}/2 & -K \end{pmatrix} \qquad \text{eq. S1-24}$$

Taking this precaution, together with distribution of internal torques that strictly conserve angular momentum (eq. S1-3) should allow to retrieve a symmetric stress tensor via eq. S1-24.

7.2.2. Comparison to Nicot et al.[1]

The analysis of the two-particle system given above suggests that eq. S1-24, and not its reported variant in the literature (eq. 34 of [1]) should be correct. The question is then why there should be such a difference. Our development suggest that eq. S1-24 is independent of particle geometry, but precaution still indicates that we should examine the proper spherical 3D case at the heart of the developments by Nicot et al.[1]

So, we re-evaluated the entire calculation by Nicot et al. [1] to identify a probable cause for the difference in the final result. While the general developments in [1] seem correct to within the limits of their various underlying conditions, we eventually found a probable integration mistake. Indeed, Nicot et al. [1] define a particle inertia matrix by (text between eq. 25 and eq. 26 in [1]):

$$\chi_{ij}^{p} = \int_{V_p} \rho r_i r_j \mathrm{d}V$$

For reasons of symmetry, χ_{ij}^{p} is diagonal with identical elements for a spherical particle[1]. Using the notation in [1] (indexation by i and j for the 3 cartesian coordinate elements x, y, z in 3D; δ being the 3x3 identity matrix; and r_p the particle radius), the integration leading to eq. 29 should be:

$$\chi_{ij}^{p} = \delta_{ij}\cdot\rho\int_{V_p} x^2 \mathrm{d}x\mathrm{d}y\mathrm{d}z = \delta_{ij}\cdot\rho\int_{z=-r_p}^{z=r_p}\int_{y=-\sqrt{r_p^2-z^2}}^{y=\sqrt{r_p^2-z^2}}\int_{x=-\sqrt{r_p^2-y^2-z^2}}^{x=\sqrt{r_p^2-y^2-z^2}} x^2 \mathrm{d}x\mathrm{d}y\mathrm{d}z$$

$$= \frac{2\cdot\rho}{3}\delta_{ij}\int_{z=-r_p}^{z=r_p}\int_{y=-\sqrt{r_p^2-z^2}}^{y=\sqrt{r_p^2-z^2}} \left(r_p^2-y^2-z^2\right)^{3/2}\mathrm{d}y\mathrm{d}z$$

with:

$$a = \sqrt{r_p^2 - z^2}$$

we have:

$$= \frac{2\cdot\rho}{3}\delta_{ij}\int_{z=-r_p}^{z=r_p}\int_{y=-a}^{y=a} (a^2-y^2)^{3/2}\mathrm{d}y\mathrm{d}z$$

$$= \frac{2 \cdot \rho}{3} \delta_{ij} \cdot \frac{3}{8} \int_{z=-r_p}^{z=r_p} a^4 \cdot \left(\sin^{-1}(1) - \sin^{-1}(-1)\right) \mathrm{d}z =$$

$$= \frac{\pi \cdot \rho}{4} \delta_{ij} \int_{z=-r_p}^{z=r_p} \left(r_p^2 - z^2\right)^2 \mathrm{d}z =$$

$$= \frac{\pi \cdot \rho}{4} \delta_{ij} \left(2r_p^5 - \frac{4r_p^5}{3} + \frac{2r_p^5}{5}\right) = \frac{4\pi \cdot \rho \cdot r_p^5}{15} \delta_{ij}$$

Since the particle mass for a sphere is given by:

$$m_p = \frac{4\pi \cdot \rho \cdot r_p^3}{3}$$

we finally have:

$$\chi_{ij}^p = \frac{m_p \cdot r_p^2}{5} \delta_{ij}$$

Compared to eq. 29 in Nicot et al.[1], there is indeed a factor of 1.5 of difference, since Nicot et al.[1] indicate $\chi_{ij,\mathrm{Nicot}}^p = \frac{2m_p \cdot r_p^2}{15} \delta_{ij}$.

We are thus lead to think that in the report by Nicot et al. [1], an integration error happened when evaluating the inertia matrix χ_{ij}^p for spherical particles; this introduced the discrepancy between eq. S1-23 and its homolog reported by Nicot et al.(eq. 34 of [1]). While this does not affect the overall development and conclusions in Nicot et al.[1], it causes an error for the important case of spherical microgel suspensions.

In our simulations, once we implemented eq. S1-24 instead of eq. 34 of [1], the asymmetry errors dramatically dropped to relative levels on the order of 10^{-14} to 10^{-17}, compatible with residual imprecision due to representation of numbers by a finite amount of digital memory.

7.3. Shear stress

Once the stress tensor calculated, the relevant shear stress is evaluated easily. It is the σ_{xy} component of the stress tensor. In analogy to rheometry, we measure the external applied force, so that the relevant shear tension is:

$$\tau_{xy}(t) = -\sigma_{xy}(t) \qquad \text{eq. S1-25}$$

As outlined in [7], the storage G' and loss moduli G'' can be obtained by demodulation of the time-dependent shear stress $\tau(t)$.

We evaluate $\tau_{xy}(t)$ using both the $\langle\sigma\rangle$ and $\langle\sigma_{external\ force}\rangle$ expressions (eq. S1-24, eq. S1-16). $\langle\sigma_{external\ force}\rangle$ represents external measurement possibilities, while $\langle\sigma\rangle$ is fully compensated for internal inertial effects and thus expected to be perfectly symmetric as

it ought to be for a stress tensor[1]. We used the two expressions to qualitatively assess whether the simulation was run at sufficiently low oscillatory frequency for the desired amplitudes. In general, we aimed at a difference between the associated $G = \sqrt{G'^2 + G''^2}$ values of no more than 30%; at low and moderate deformations, we more typically achieve differences below 10%.

8. Bibliography

1 Nicot, F., Hadda, N., Guessasma, M., Fortin, J. & Millet, O. On the definition of the stress tensor in granular media. *Int J Solids Struct* **50**, 2508-2517, doi:10.1016/j.ijsolstr.2013.04.001 (2013).

2 Ries, A., Wolf, D. E. & Unger, T. Shear zones in granular media: three-dimensional contact dynamics simulation. *Phys Rev E Stat Nonlin Soft Matter Phys* **76**, 051301, doi:10.1103/PhysRevE.76.051301 (2007).

3 Brendel, L., Unger, T. & Wolf, D. E. in *The Physics of Granular Media* (eds H. Hinrichsen & D. E. Wolf) (Wiley-VCH, 2005).

4 Strauch, D. *Classical Mechanics - An Introduction*. (Springer, 2009).

5 Luding, S. in *The Physics of Granular Media* (eds H. Hinrichsen & D. E. Wolf) (Wiley-VCH, 2005).

6 Fortin, J., Millet, O. & de Saxce, G. Construction of an averaged stress tensor for a granular medium. *Eur J Mech a-Solid* **22**, 567-582, doi:10.1016/S0997-7538(03)00054-8 (2003).

7 Otsuki, M. & Hayakawa, H. Discontinuous change of shear modulus for frictional jammed granular materials. *Phys Rev E* **95**, 062902, doi:10.1103/PhysRevE.95.062902 (2017).

8 Schramm, G. *A Practical Approach to Rheology and Rheometry*. (Gebrueder HAAKE GmbH, 1994).

9 Lai, M., Krempl, E. & Ruben, D. *Introduction to continuum mechanics*. (Elsevier, 2010).

10 Evans, I. D. & Lips, A. Concentration-Dependence of the Linear Elastic Behavior of Model Microgel Dispersions. *J Chem Soc Faraday T* **86**, 3413-3417, doi:DOI 10.1039/ft9908603413 (1990).

11 Bhattacharjee, T. *et al.* Polyelectrolyte scaling laws for microgel yielding near jamming. *Soft Matter* **14**, 1559-1570, doi:10.1039/c7sm01518f (2018).

12 Popov, L. V. *Contact Mechanics and Friction*. (2017).

13 *Python Software Foundation. Python Language Reference, version 3.7.*, <https://www.python.org> (

14 Briggs, J. R. *Python for Kids: A playful introduction to programming*. (No Starch Press, Inc., 2013).

15 Lees, A. W. & Edwards, S. F. The computer study of transport processes under extreme conditions. *Journal of Physics C: Solid State Physics* **5**, 1921-1928 (1972).

16 Chang, C. S. in *Mechanics of Granular Materials: An introduction* (A. A. Balkema, 1999).

17 Pereira, C. M., Ramalho, A. L. & Ambrosio, J. A. A critical overview of internal and external cylinder contact force models. *Nonlinear Dynam* **63**, 681-697, doi:10.1007/s11071-010-9830-3 (2011).

18 Hooke, R. *De Potentia Restitutiva, or of Spring. Explaining the Power of Springing Bodies* (Printed for John Martyn, 1678).
19 van Hecke, M. Jamming of soft particles: geometry, mechanics, scaling and isostaticity. *J Phys Condens Matter* **22**, 033101, doi:10.1088/0953-8984/22/3/033101 (2010).
20 Liu, J. X. & Davies, T. J. Coordination number-density relationships for random packing of spherical powders. *Powder Metall* **40**, 48-50, doi:DOI 10.1179/pom.1997.40.1.48 (1997).
21 Dagois-Bohy, S., Tighe, B. P., Simon, J., Henkes, S. & van Hecke, M. Soft-Sphere Packings at Finite Pressure but Unstable to Shear. *Phys Rev Lett* **109**, doi:ARTN 095703
10.1103/PhysRevLett.109.095703 (2012).
22 Love, A. E. H. *A Treatise on the Mathematical Theory of Elasticity*. (Cambridge University Press, 1927).
23 Weber, J. Recherches concernant les contraintes intergranulaires dans les milieux pulvérulents. *Bull. Liaison P. et Ch.* **20**, 1-20 (1966).
24 Luding, S., Lätzel, M., Volk, W., Diebels, S. & Herrmann, H. J. From discrete element simulations to a continuum model. *Computer Methods in Applied Mechanics and Engineering* **191**, 21-28, doi:https://doi.org/10.1016/S0045-7825(01)00242-0 (2001).
25 den Hartog, J. P. *Mechanics*. (Dover Publications, Inc., 1948).
26 Holzner, S. *Physics for Dummies*. (Wiley Publishing, 2006).

Part 2: Usage instructions for the Python simulation

1. Aim of Part 2

Part 2 provides high-level usage instructions for the custom Python package particleShear.

2. Content

3. Software installation

3.1. Overview

We implemented the numerical simulation in Python[1,2]. We organized the code as installable package Python "particleShear". This package contains a set of Python classes allowing to model various particles and particle assemblies. The package also provides preconfigured simulations with shear experiments. This section describes installation of the software package.

3.2. Installation

3.2.1. Python

The package particleShear was tested with **Python 3.5.0 and Python 3.7.0** and should run with these versions or above. It uses the graphical interface **Tkinter**, and care should be taken to install a distribution of Python that includes Tkinter.

3.2.2. Pre-requirements

The package particleShear uses the libraries **math**, **random**, **io** and **pathlib,** which are all part of the standard distribution of Python. The unit tests also require **unittest**, which is also part Python's standard distribution.

The package particleShear needs the PIL module for image handling. This module is available by installing the **Pillow** package. The particleShear package states this dependency in its setup file, and therefore, installation of the particleShear package should normally trigger the installation of Pillow. If not, it can be installed separately (see below).

For file-saving as jpg images, the Ghostscript application (executable independent of Python) may also need to be installed and configured on the search PATH of the operating system.

The Pillow package is freely available for download, and is typically installed by an automatic installer such as pip. An example command on MacOSX to install Pillow would be:

```
python –m pip install Pillow
```

Alternatively, if using an integrated development environment such as Pycharm, packages such as Pillow will need to be installed specifically in these environments.

3.2.3. Package installation

The package particleShear can be installed using Python's default installer "pip". With the advent of Python3, the file name of the actual executables has become a bit variable, such that the installation commands also vary a bit depending on the installation. Typically, they can be:

```
pip3 install particleShear
```

or

```
pip install particleShear
```

or

```
python3 -m pip install particleShear
```

or

```
python -m pip install particleShear
```

In our experience, the `python -m pip` or `python3 -m pip` commands are safer because especially in systems that have seen multiple python updates, the pip or pip3 executables install the packages to places that cannot always be found by the python or python3 executables…

The install commands above look for the most recent suitable version on the python package index server pypi (https://pypi.org/search/?q=particleShear). This means that by using the above commands, you will install the most recent version of particleShear.

For reproducibility purposes, it may be interesting to install the particleShear package version used for this CodeOcean capsule. It is available on Zenodo (https://doi.org/10.5281/zenodo.4589212). To install this reference version, download the zip ("particleShear-v1.0.2.zip") from Zenodo, and unzip it. An example command on MacOSX to install the package particleShear would be:

```
python -m pip install /path/to/unzippedfilecontents/ tbgitoo-particleShear-0bb216a
```

with `/path/to/unzippedfilecontents` path part to be replaced by its actual value, which depends on where you downloaded the file and the operating system.

Again, other standard python package installation mechanisms are also suitable, as for any other Python package.

4. Package usage

4.1. Basic usage

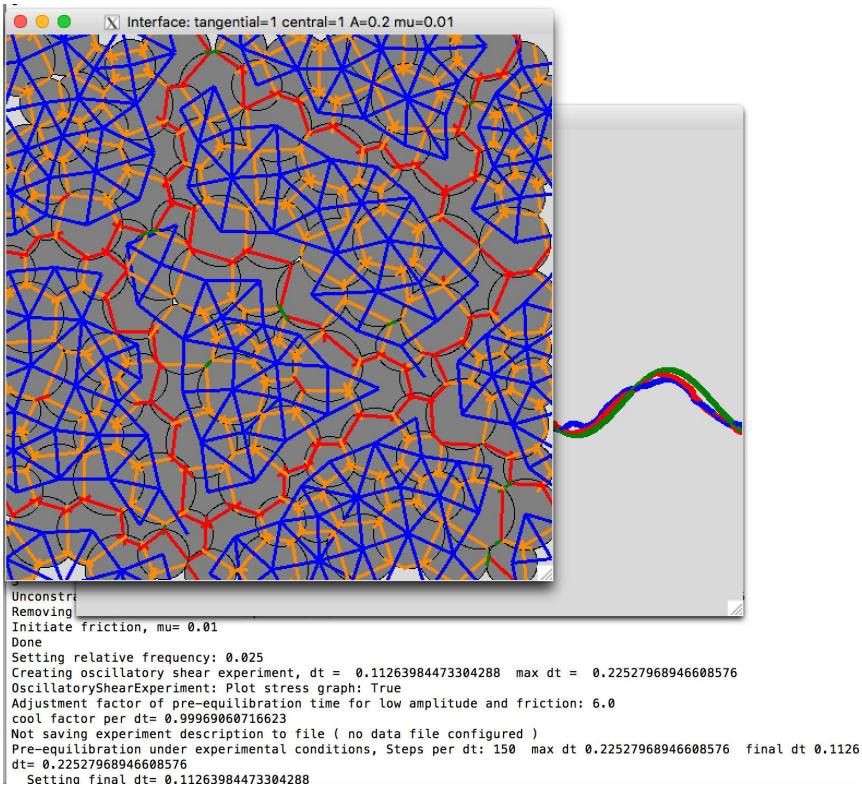

Fig. S2-1: Screenshot from a running simulation. The simulation was launched from a python interpreter running in a MacosX terminal; the text output is seen in the background. Two graphical windows are shown by default: a first window displaying the particles and interfaces being simulated (top window), and a second window displaying strain excitation and stress measurement during rheology.

The package can either be used as a high-level interface to the simulation, or as an API to program specific derived simulations, additions or corrections.

The most basic usage consists in explicitly invoking high-level functions of the package in a Python shell. For this, a python interpreter is launched, for instance via the command:

```
python
```

or

```
python3
```

depending on your installation. In the Python shell line environment, the particleShear package needs to be imported via:

```
from particleShear import *
```

Having imported the package, a simulation with default parameters can be launched via:

```
doParticleShearSimulation()
```

This launches a simulation with graphical display (Fig. S2-1), so the particle ensemble should be shown rapidly on a graphical popup window. Once pre-equilibration of the particle ensemble is completed, a second graphical display should pop up, displaying the rheology measurement. Green is excitation amplitude, blue is shear stress (eq. S1-25 from) as measured by external force (eq. S1-16) and red is shear stress (eq. S1-25) as measured by overall stress tensor tensor (eq. S1-24). The shear measurements consists in a baseline period to validate ensemble equilibration without deformation, followed by 3 cycles of sinusoidal excitation, followed by a second baseline period to follow re-equilibration. At the end of the simulation, G' and G'' values are determined by demodulation[3] using the shear stresses from both tensors and displayed in the Python command line environment. By default, the simulation and rheological evaluation are displayed graphically, but no data is saved. To save data, or images, as well as to change the parameters of the simulation, specific options need to be passed to the function `doParticleShearSimulation` (see Simulation options below).

Alternatively, rather than running the commands directly from the command line, python scripts (text files with the extension .py) are more typically used. Such a .py file needs to contain at least the import statement to make the package functionality available, and a function call to make the simulation run.

4.2. Simulation options

In high-level usage, the options of the simulation are passed to `doParticleShearSimulation` in the form of named parameters. For instance, to run the simulation with general default options, but a particle Young modulus of 20kPa instead of the default 8kPa, one would invoke `doParticleShearSimulation` via the Python commands:

```
from particleShear import *
doParticleShearSimulation(Young_modulus_spheres=20000)
```

The full list of parameters, along with their description and default value is given in Table S2-1. Here, we investigated the effects of only a subset of these parameters, this is documented in the "Varied here?" column in Table S2-1.

Parameter	**Default**	**Varied here?**	**Description**
root_folder	""	Yes (to save data locally)	Folder location for storing output data. Needs to be provided only if saveOutputImages or saveData are set to True
theAmplitude	0.2	Yes (various values)	Strain amplitude for the oscillatory excitation. Provided in absolute value, not percent, so for instance 0.5 = 50%
theMu	0.01	Yes (various values)	Friction coefficient μ for the geometrical contacts between the spheres
do_permanent_links	True	Yes (compariso	Indicates whether there should be permanent links between

		n between crosslinked and non-crosslinked)	neighboring particles. Providing False runs a simulation with spherical particles only as in Otsuki et al.[3]
cut_lines	5	Yes (0 for bulk scaffold)	Main parameter for particle generation by placing cuts through the permanently crosslinked network. Use depends on the value of doCutByTriangulation: If doCutByTriangulation is True, this indicates the number of triangulation points used to generate the particles; if doCutByTriangulation = False, indicates the number of straight cutting lines used for particle generation
Young_modulus_spheres	8000	Yes (for comparison)	Young modulus of the constituent spheres, in Pa
N	150	No	The number of spheres to be placed in the simulation area
packing_fraction	1.5	Yes (between 1 and 2, for comparison)	The relative density of spheres, used in the calculation of their radius via eq. 14
density	1000	No	Density of the spheres, in kgm^{-3} (approximately the density of water)
bimodal_factor	1.4	No	Ratio between the large spheres and the small spheres, we used the factor of 1.4 proposed by Otsuki et al.[3]
relative_viscosity	0.1	No	Viscosity constant relative to the one defined by Ostuki et al., which is $\sqrt{mk}$ [3]. In other words, allows to set an arbitrary value for the coefficient in eq. 9
relative_transversal_link_strength	1	No	This is the ratio between the central spring constant k and the transversal spring constant k_T. Otsuki et al.[3] used $k=k_T$, and so did we by using the default value of 1.
relative_frequency	0.025	No	Frequency for the oscillatory shear excitation, expressed relative to the characteristic

			frequency $f = \frac{1}{\tau} = \sqrt{k/m}$
avoid_horizontal_angle_degree	15	No, but has no effect here	Applicable only if doCutByTriangulation = False, allowing to avoid cuts too near to horizontal, which would create preferential shear planes
interface_reenforcement_central	1	No	Allows to adjust the relative k values of geometric vs. permanent interfaces
interface_reenforcement_tangential	1	No	Allows to adjust the relative k_T values of geometric vs. permanent interfaces
central_repulsion_coefficient	1	No	Allows to adjust the strength of non-linear central repulsion. 1 corresponds to the force law given by eq. 5b, 0 to the linear law given by eq. 5a and used in the simulation by Otsuki et al.[3]
keep_viscosity_coefficients_constant	False	No, but has no effect here	If interface_reenforcement_central or interface_reenforcement_tangential are different from 1, should the η and η_T values be changed along with the k and k_T values ?
avoid_height_spanning_particles	True	No	Only has an effect when do_permanent_links is True and cut_lines >= 1: When crosslinking the assembly into individual crosslinked particles, should we avoid particles spanning the entire height?
cut_top_bottom	False	No	Should we artificially place horizontal cuts near the top and bottom shear plates (to mimick wall slip)?
saveOutputImages	False	Yes (to make illustrations)	Should snapshots of the simulation canvas be taken and stored on disk?
imageFileType	"jpg"	Yes	If saveOutputImages is True, should we save bitmap (jpeg, encoded by "jpg") or vector (postscript, encoded by "ps") images?
remove_link_fraction	0	Yes (to emulate weaker internal	After creation of crosslinked particles, should we remove a fraction of the crosslinks (removal of "non-essential"

		crosslinkin g by porosity)	crosslinks only to keep identity of particles)? If yes, a value > 0 provides the target fraction that should be randomly removed
edge_fuzziness	0	No	Possiblity to make the particle edges less straight
doCutByTriangulation	True	No	Particles are produced by first fully crosslinking the neighboring spheres, followed by “cutting” into individual particles. Two methods are available: cutting by straight lines, or by triangulation between randomly assigned particle centers (this option is chosen by doCutByTriangulation=True)
doDrawing	True	Yes (to run on cluster where graphical display causes the scripts to crash)	Possibility to switch on or off graphical display of the spheres and interfaces during simulation
saveData	False	Yes (to store output data)	Save simulation description and output data in text file
saveStressTensorData	True	Yes (to store stress tensor output or not)	Only has an effect if saveData=True. Indicates whether detailed stress tensor data should be saved in addition to summary output data
plotStress	True	Yes (to run on cluster where graphical display causes the scripts to crash)	Should strain excitation amplitude and shear force response be displayed graphically?
relative_y_scale_force	1e-4	No	To adjust the y-scaling of the shear force to match the graphical display
pre_periods	2	No	How long (relative to one excitation period = 1/f) should baseline be recorded before starting to apply shear?
periods	3	No	How many shear periods should

			be recorded?
post_periods	1	No	How long (relative to one excitation period = 1/f) should baseline be recorded after the shear excitation experiment?
cool_factor	0.5	No	Dissipation of speed relative to interstitial fluid: Decay factor per excitation period of linear speed in excess to anticipated local flow rate (see section Damping)

Table S2-1. Simulation parameters. These parameters are the named parameters of the function `doParticleShearSimulation`. The simulation area is currently fixed to a default of 500x500 micrometers = 500x500 in the high-level function `doParticleShearSimulation`, although it can be changed in the actual particle ensembly classes (see Part 3).

4.3. Custom contact laws

4.3.1. Built-in contact laws

Adjustment of the contact force law (see Part 1, Fig. S1-3) between the spheres is possible via the argument `central_repulsion_coefficient` passed to `doParticleShearSimulation`. This allows to adjust the strength of the additional central repulsion relative to the simple linear law (see Table S2-1). The contact force law concerns only the elastic, not the viscous part of the sphere-sphere interactions.

4.3.2. Arbitrary contact laws

Beyond the built-in default contact law described above, it is also possible to provide custom contact laws via a function callback mechanism.

For this, it is necessary to set necessary to set the static class field `CircleBasicElasticity.call_back_elastic_force_law`. This field should be set to a function providing the force between pairs of spheres during the simulations. Indeed, this function will be called for each relevant pair of spheres with four arguments, in order:

1) The actual distance between the centers of two spheres (`d`)
2) The equilibrium distance between the two spheres (`d0`)
3) The spring constant k for contact link between the two spheres (`k`)
4) The central repulsion coefficient for contact link between the two particles (`central_repulsion_coefficient`)

The callback function is expected to return the numerical value of the force between the particles, positive values indicating attraction, negative values indicating repulsion.

By default, `CircleBasicElasticity.call_back_elastic_force_law` is set to the function `elastic_force_law`. This function is defined in particleShearBase/CircleBasicElasticity.py (accessible at https://github.com/tbgitoo/particleShear) and can serve as an example to implement new custom contact law functions.

4.3.3. Predefined non-standard contact laws

For the purpose of simulation of highly compressible, sponge-like particles with a plateau force law (see /code/Documentation/CodeOceanOnlyResults/CodeOceanOnlyResults.pdf, section 7), we implemented a convenience function that can be used instead of the default. This function is `elastic_force_law_plateau`, it is defined in particleShearBase/CircleBasicElasticity.py (see https://github.com/tbgitoo/particleShear).

To use this alternate force law, set the callback function prior to running the simulation:

```
CircleBasicElasticity.call_back_elastic_force_law=elastic_force_law_plateau
```

As the callback function is limited to four arguments at usage, we also provide a configuration class for fine-tuning the plateau law. This class is `PlateauConfiguration` , it is also defined in particleShearBase/CircleBasicElasticity.py. This class has three static members (it is not meant to be instantiated as an object), namely `relative_plateau_begin`, `relative_plateau_end` and `relative_plateau_slope`. Collectively, these three members set the plateau behaviour to be employed: the first two parameters set the beginning and end of slope (0=full compression, 1=particles just in contact, generally with `0<relative_plateau_begin<=relative_plateau_end<1`. The parameter `relative_plateau_slope` describes the decrease of the slope on the plateau relative to a simple linear law, 0 indicating a completely horizontal plateau, 1 being no change (that is, with `relative_plateau_slope=1`, `elastic_force_law_plateau` becomes equivalent to the default function `elastic_force_law`. Note that `elastic_force_law_plateau` incorporates enhanced central repulsion, just like `elastic_force_law` does.

4.4. Display during simulation

The simulation can be configured to produce different kinds of output depending on the options passed to `doParticleShearSimulation` (see Table S2-1).

By default, the simulation displays the particle ensemble during simulation and plots the strain and stress (options `doDrawing=True` and `plotStress=True`). If necessary, this graphical display can be switched off by passing `False` for the two options.

4.5. Image saving

By default, the simulation does not write image files to disk. Image saving can however be enabled by passing `saveOutputImages=True` to `doParticleShearSimulation`. For

this to work, the `root_folder` argument must be set to some existing folder where the images can be stored on disk. The simulation will then create a sub-folder for the images in the folder designated by `root_folder`, and save a series of snapshots taken at 200 regularly spaced time-points during the simulation. Snapshots are only taken during the actual rheological shear protocol: baseline as determined by the `pre_periods` option, oscillatory excitation as determined by the `periods` option, and post-excitation recovery as determined by the `post_periods` option, according to Table S2-1. No snapshots are taken during ensemble constitution and pre-equilibration before the baseline period. Since the images saved are actual snapshots of the graphical display, graphical display must be enabled for image saving (option `doDrawing=True`).

4.6. Saving output as text

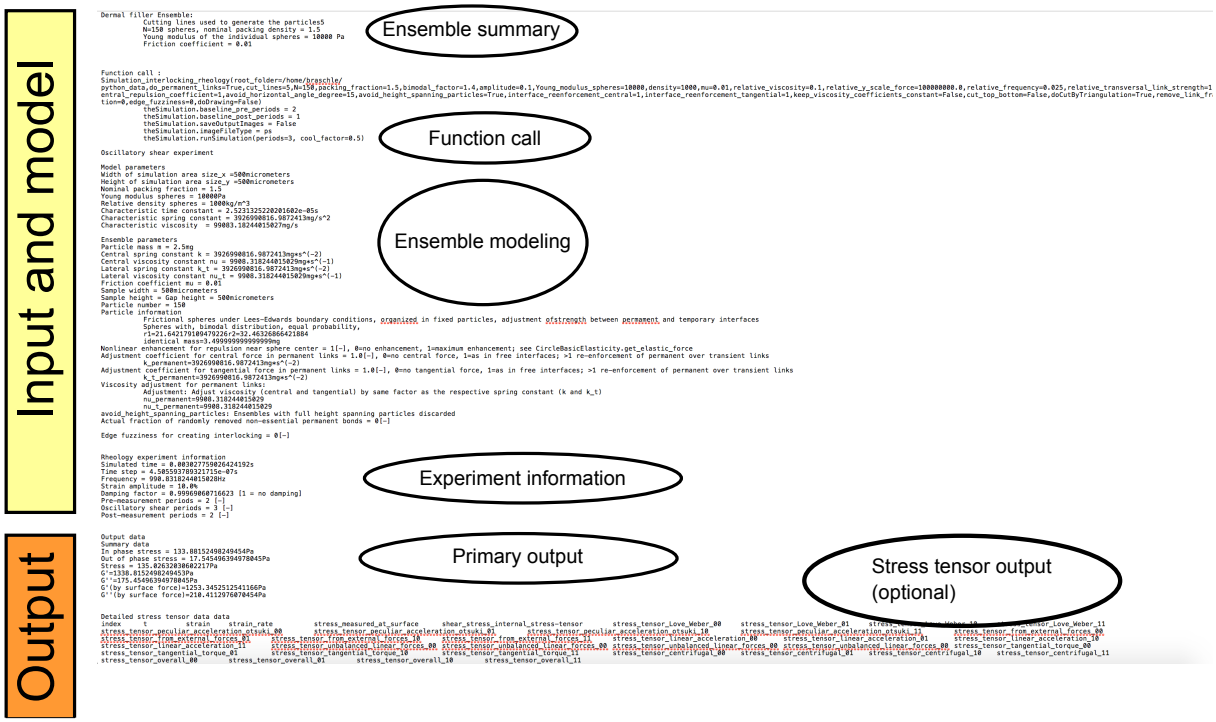

Fig. S2-2. General structure of text output files generated by the simulation. Only the first few lines of the stress tensor output are shown.

Besides images, the simulation can also be configured to store the main parameter and output information in an ASCII text file (Fig. S2-2). To do so, the `saveData` option should be passed as `saveData=True` to `doParticleShearSimulation`. As for image saving, the `root_path` argument should be set to a folder where the text file can be stored.

If `saveData=True` and `root_path` points to a folder where the simulation can store files, it will create a text file and store both the simulation description and output data. The further behavior depends on the value of `saveStressTensorData`. If `saveStressTensorData=False`, the simulation will only save the simulation parameters and the primary output values, producing a relatively short file, whereas if `saveStressTensorData=True`, various stress tensor values will be saved at all the simulation time-point, leading generally to file sizes in the lower Megabyte range (Optional stress tensor output, Fig. S2-2). The input values given to `doParticleShearSimulation`

(Table S2-1) are recapitulated in the ensemble summary, function call and ensemble modeling section. The ensemble modeling section also contains model-derived simulation parameters such as the spring and viscosity constants used to model the particle interactions (see Fig. S1-1 and Part 1 in general).

The primary output values are described in more detail in Table S2-2.

Designation in text file	Description	Units
In phase stress	Shear stress τ_{xy} (eq. S1-25), evaluated from symmetrical overall stress tensor (eq. S1-24); demodulated by excitation amplitude (in-phase part)	Pa (indicated in text file)
Out of phase stress	Shear stress τ_{xy} (eq. S1-25), evaluated from symmetrical overall stress tensor (eq. S1-24); demodulated by excitation amplitude (out-of-phase part)	
Stress	Total stress from the in- and out-of-phase componentas: $\sqrt{(\text{in phase})^2 + (\text{out of phase})^2}$	
G’	G’ as in-phase value by demodulation from τ_{xy} (eq. S1-25), evaluated from symmetrical overall stress tensor (eq. S1-24); see eq. 15 of Otsuki et al.[3] for the demodulation	
G’’	G’’ as out-of-phase value by demodulation from τ_{xy} (eq. S1-25), evaluated from symmetrical overall stress tensor (eq. S1-24); see eq. A1 of Otsuki et al.[3] for the demodulation	
G’(by surface force)	G’ as in-phase value by demodulation from τ_{xy} (eq. S1-25), evaluated from the external force stress tensor (eq. S1-16); see eq. 15 of Otsuki et al.[3] for the demodulation	
G’’(by surface force)	G’’ as out-of-phase value by demodulation from τ_{xy} (eq. S1-25), evaluated from the external force stress tensor (eq. S1-16); see eq. A1 of Otsuki et al.[3] for the demodulation	

Table S2-2. Primary simulation output values in the output text files. If `saveData=True` *is passed to* `doParticleShearSimulation`*, a text output file will be saved to disk in the folder designated by the* `root_path` *argument. The global structure of the text file is shown in Fig. S2-2. This tables contains the primary output fields (see Fig. S2-2 for the localization of the primary output section in the text file).*

If `saveStressTensorData=True` is passed, both the simulation overview and the detailed stress tensor values as a function of time are saved.

Table S2-3 contains a detailed description of the stress tensor fields stored in the text file if `saveStressTensorData=True`.

Designation in text file	Description	Units

index	Index starting at 0 at the outset of the oscillatory shear experiment (that is, after ensemble pre-equilibration, for which no values are stored)	-
t	Time from the outset of the oscillatory shear experiment	s
strain	Applied strain. No strain is applied during baseline acquisition (see argument pre_periods in Table S2-1), followed by a oscillatory shear period (see argument periods in Table S2-1), followed again by a period with out applied shear movement (see argument post_periods in Table S2-1). Strain is given as absolute values, (so, not in percent).	-
strain_rate	Momentaneous rate of change of the strain (d(strain)/dt)	s^{-1}
stress_measured_at_surface	Instantaneous shear stress value τ_{xy} (eq. S1-25), evaluated from the external force stress tensor (eq. S1-16)	Pa
shear_stress_internal_stress-tensor	Instantaneous shear stress value τ_{xy} (eq. S1-25), evaluated from symmetrical overall stress tensor (eq. S1-24)	Pa
stress_tensor_Love_Weber_00 to stress_tensor_Love_Weber_11	4 columns for the Love-Weber[4,5] stress tensor (eq. S1-19)	Pa
stress_tensor_peculiar_acceleration_otsuki_00 to stress_tensor_peculiar_acceleration_otsuki_11	4 columns for the inertial compensation stress tensor proposed by Otsuki et al. (last right-hand term in eq. 15 of ref. [3])	Pa
stress_tensor_from_external_forces_00 to stress_tensor_from_external_forces_11	4 columns for the stress tensor as evaluated from external forces (eq. S1-16)	Pa
stress_tensor_linear_acceleration_00 to stress_tensor_linear_acceleration_11	4 columns for the linear acceleration stress tensor (eq. S1-17)	Pa
stress_tensor_unbalanced_linear_forces_00 to stress_tensor_unbalanced_linear_forces_11	4 columns for the unbalanced linear force stress tensor; we do not consider gravity, so this should be equal to the linear acceleration stress tensor (eq. S1-17)	Pa
stress_tensor_tangential_torque_00 to stress_tensor_tangential_torque_11	4 columns for the unbalanced torque stress tensor, taking into account internal and external torques (last right-hand term for eq. S1-23, for K=0 for all particles)	Pa
stress_tensor_internal_tangential_torque_00 to stress_tensor_internal_tangential_torque	4 columns for the internal unbalanced torque stress tensor, taking into account only internal torques (last right-hand term for eq. S1-24, for K=0 for all particles)	

_00		
stress_tensor_centrifugal_00 to stress_tensor_centrifugal_11	4 columns for the centrifugal acceleration stress tensor (last right-hand term for eq. S1-24, for T_{int}=0 for all particles)	Pa
stress_tensor_overall_00 to stress_tensor_overall_11	4 columns for the overall symmetric internal stress tensor (eq. S1-24). The second and third column should be identical.	Pa

Table S2-3. Stress tensor output values in the output text files This tables contains the primary stress tensor output fields (see Fig. S2-2 for the localization of the stress tensor output section in the text file). The stress tensor output fields are only saved if `saveStressTensorData=True` *is passed to* `doParticleShearSimulation`.

5. Replication instructions

For the purpose of verification, but also potential future developments by others, we give the full replication instructions for our simulations here (Table S2-4 below). At present (2021) this requires the use of a cluster (total ca. 30'000 CPU hours). These simulations were run at a historical development stage with manual install of both Python and R evaluation libraries. We highly recommend the use of the present, automated distribution but provide this information for the purpose of reproducibility. The necessary historical files can be found at https://doi.org/10.5281/zenodo.2653804.

If you are looking to run a few simulations locally for demo purposes or to start your own developments, please refer to the Quick Install Guide on the CodeOcean capsule (at /code/Documentation/Simulation particleShear/Quick install.pdf), the use of cluster is only necessary to replicate the large number of simulations run for the manuscript "An injectable meta-biomaterial". If you're interested in looking at the visual output, standard scenarios are provided as Supplementaries to "An injectable meta-biomaterial" but also in the results section showing the results of reproducible runs of this capsule (see /results/Simulation demo/), since each reproducible run actually runs the a few simulations.

Ideally, running the complete set of simulation takes little human effort, but there are a number of requirements that will most certainly call for some time investment when transferring our simulations from the Baobab cluster (University of Geneva) to different cluster or even to a different account on the Baobab cluster. First, the Baobab cluster at the University of Geneva runs via the slurm workload manager[6], if another job distribution utility is used or if this is done manually, then full reconfiguration is required. The mode of installation of Python and Python modules can be another source of cumbersome problems if not offered by default by the cluster. Finally local file paths need to be adapted in numerous files (this might be itself best be done with scripts on the full scale).

So while very little human effort is required to run simulations once everything is configured, the configuration of the cluster environment itself, and adaptation of paths in various files may take a few days.

Step	Computation / human effort	Details	Paths, files, comments
1) Copy files to cluster environment	10 minutes	Transfer of the Python module particleShear and the custom python scripts to run the simulations	The steps to transfer the required files from the CodeOcean capsule to the cluster are described in /data/Raw/Simulation/cluster_setup/01_upload_files.txt in this capsule Access to a linux cluster via ssh and file-upload via scp is required. Other access modes are possible but the commands obviously will need to be adapted.
2) Python simulation	10 min launch per round, ca. 30'000 CPU-hours in total	Running of particleShear simulation defined by a set of scripts	The steps to setup up the simulation environment and launch simulations are described in /data/Raw/Simulation/cluster_setup/02_setup_and_run.txt The simulation setup requires access to a linux cluster via ssh. It is mandatory that Python is installed or can be installed during simulation setup, and that custom Python modules as well as publicly accessible modules can be installed. It will most certainly be necessary to adapt the corresponding commands in /data/Raw/Simulation/cluster_setup/02_setup_and_run.txt for this. Running the simulations is at present done using the sbatch and srun utilities, which are part of the slurm workload manager system[6] used. The exact configuration of the cluster job environment (partitions, time limits, …) will also need to be adapted (see and adapt /data/Raw/Simulation/cluster_setup/slurm_job.sh)
3) Recovery of output files	Via wireless, ca. 24h	Download ASCII output files generated by the simulations	We use rsync commands for the download (provided here, see /data/Raw/Simulation/cluster_setup/03_download_results.txt) We ran the simulations block-wise (avoiding storage in the same primary output folders, see /data/Raw/Simulation/cluster_setup/02_setup_and_run.txt) and recovered the

			different blocks in different folders for manageability, but this is not strictly necessary for replication
4) Installation of custom R libraries	1h on laptop	Installation of custom R libraries needed for analysis of the output files downloaded in step 3	We use 4 historical R libraries for the analysis : plot.counts, probabilityUtils, textureAnalyzerGels, particleShearSimulation. Historical versions, at https://doi.org/10.5281/zenodo.2653804, "10 R libraries.zip"
5) Splitting of output files	Overnight on laptop	Splitting of ASCII files into small header ASCII and stress tensor data in large R-Data files	For the sake of storage space, we did not keep the full text output files from the simulation (about 500GB), but split the simulations into a small text header file and a compressed rda file each, allowing to store the full data in 120GB. A complete workflow from primary simulation output to graphing is provided in this capsule at /code/Simulation demo/demo_evaluation, and it is probably easiest to adapt this workflow to new simulations. For documentation purposes, the splitting scripts (with local paths) are however available. The ones for the main simulation at our 2019 Zenodo repository: https://doi.org/10.5281/zenodo.2653804 ,"11 R Raw data reading.zip", scripts in folders "Simulation/01_read_overview_do_not_run", then "Simulation/02_split_text_and_rda_do_not_run" For, these scripts need to be adapted regarding local file paths. Aside from the main simulations, for Fig. 5e, we ran a second set of simulations with a plateau contact law to emulate higher porosity. These simulations were done in 2020, and the splitting scripts are stored for reference in the capsule (at /data/Raw/Simulation/nonlinear_force_law/splitting_scripts, 01_R_script_read_overview.R and 02_R_script_split_text_and_R.R). As before, these scripts can only be run directly on the full text output of

			simulations, and are provided for reference.
6) Primary data reading	Covered by reproducible run on CodeOcean. A few hours. Same if run locally on PC.	Gathering overview files from the simulation output (split), and correction for "jumps" generated by spheres leaving and entering the simulation area in the Python simulations	Done during reproducible CodeOcean runs (R-scripts at /code/Data analysis/Data import/Simulation, for plateau law in /code/Data analysis/Data import/Simulation/nonlinear_force_law)
7) Resampling for boot-strapping analysis	Covered by reproducible run on CodeOcean. A few minutes	Resample for bootstrapping analysis of overall features of the stress-strain curves (Fig. 2f, Fig. 2g in the main text, online methods)	Specifically, R-Script /code/Data analysis/Data import/Simulation/05_R_script_bootstrap.R, applies only to main simulation.
8) Figure plotting and statistical analysis	Covered by reproducible run on CodeOcean. At most a few minutes per subfigure	Replicate simulation figures and statistical analysis for the main text and supplementaries	Scripts under /code/Data analysis

Table S2-4. Replication instructions

We provide replication instructions for various levels of replication, from full re-running of the simulations to replotting of the main results. The CodeOcean part (steps 6-8) is run automatically at each CodeOcean capsule "Reproducible Run", ensuring reproducibility by desing. It can of course also be replicated and adapated locally after download of the contents of the CodeOcean capsule. However, unless directly downloaded to the system root in MacOSX or Linux ("/"), all the paths need to be adapted.

5.1. Replication of the Python simulation

The most complete, but most demanding approach regarding computing ressources is to re-run the historical Python simulations to generate a new dataset. This is described in steps 1 to 3 of Table S2-4 above.

5.2. Replication of raw data reading

Replication of the full simulation is demanding in terms of computing power (ca. 30'000 CPU hours), and also generates a very large data volume (about 500GB). For this reason, the data was compressed prior to upload to the CodeOcean capsule (to about 120GB). This compression is achieved by transforming the large primary text output files to corresponding small header files and R-Data files (Steps 4-5 in Table S2-4).

5.3. Replication of data analysis and Figure plotting

Data analysis is carried out reproducibly within this CodeOcean capsule (Steps 6-8 in Table S2-4). This can be replicated locally after download of the CodeOcean capsule contents and adaptation of the file paths if desired.

6. Bibliography

1 *Python Software Foundation. Python Language Reference, version 3.7.*, <https://www.python.org> (

2 Briggs, J. R. *Python for Kids: A playful introduction to programming*. (No Starch Press, Inc., 2013).

3 Otsuki, M. & Hayakawa, H. Discontinuous change of shear modulus for frictional jammed granular materials. *Phys Rev E* **95**, 062902, doi:10.1103/PhysRevE.95.062902 (2017).

4 Love, A. E. H. *A Treatise on the Mathematical Theory of Elasticity*. (Cambridge University Press, 1927).

5 Weber, J. Recherches concernant les contraintes intergranulaires dans les milieux pulvérulents. *Bull. Liaison P. et Ch.* **20**, 1-20 (1966).

6 SchedMD. *slurm workload manager*, <https://slurm.schedmd.com/documentation.html> (2020).

Part 3: Implementation of the numerical simulation

1. Aim of Part 3

This part provides documentation to the Python implementation of the numerical simulation in particleShear.

2. Content

3. Overview

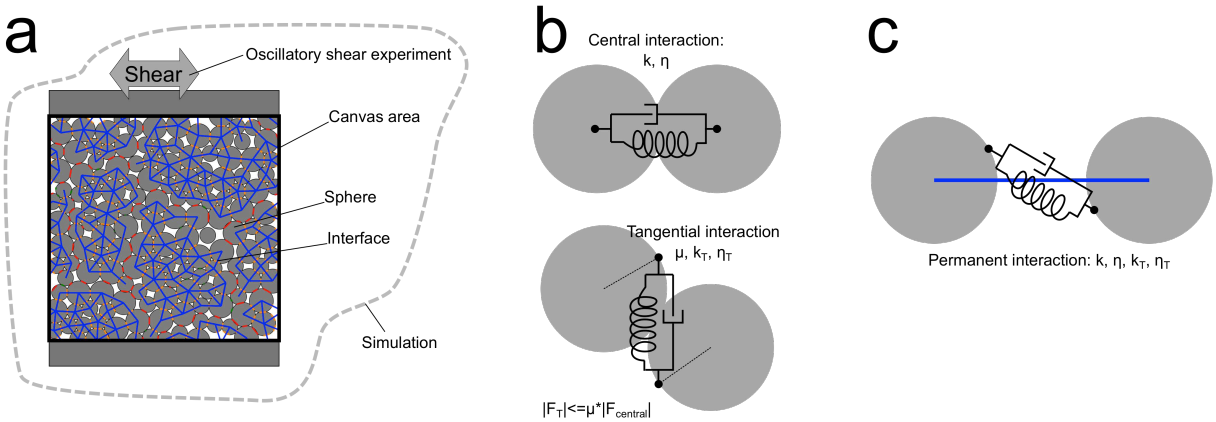

Figure S3-1: Design of the simulation. a) shows a screenshot of a simulation with "spheres" (circles) arranged on a canvas, and with permanent (blue) and non-permanent (red or green according to slipping) interaces. b) interaction parameters for geometrically touching spheres. See SI-1. c) Interaction in permanent interfaces. See SI-1.

Fig. S3-1, replicated here for convenience from Part 1 shows the basic layout and principal parameters of the numerical simulation. We implemented this simulation in a custom Python package, to be able to extend it to permanently crosslinked particles. The simulation combines the physical framework from Otsuki et al. [1], corrected for conservation of angular momentum, with the stress tensor evaluation framework by Nicot et al.[2], with additional corrections (Part 1).

Knowledge of the detailed implementation is not necessary for basic usage of the package, as described in Part 2. It however should facilitate both checking and extension of the package by using it as an application programming interface.

4. Package structure

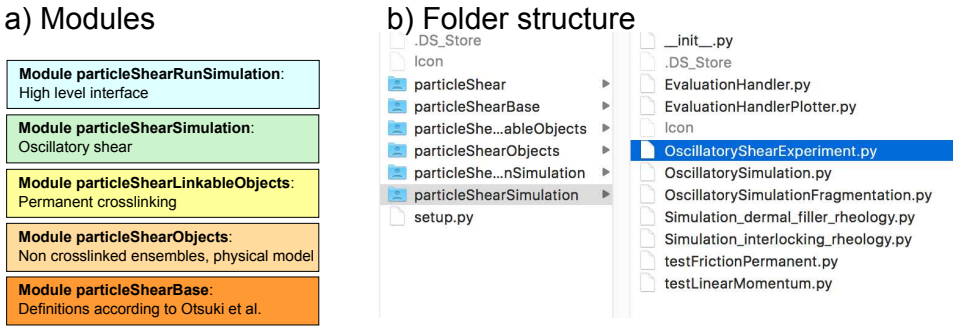

Figure S3-2: Structuring of the particleShear package. a) Hierarchy of the modules contained in the particle shear package. Otsuki et al. designates reference [1]. b) Associated folder structure. The folders corresponding to the five modules listed in Fig. S3-2a contain python source code files defining the classes of each module. Here, this is highlighted for the example of the "OscillatoryShearExperiment" class in the module "particleShearSimulation". In addition to the actual source code files, there are also some elements required for installable Python packages. First, each module folder contains an __init__.py file necessary specifying the publicly accessible elements. The folder

"particleShear" contains only the __init__.py file. Second, there is a setup.py file at the root of the package, also required for installable Python packages.

The python scripts in the particleShear package are structured into 5 modules, as shown in Figure S3-2a. Each module provides a set of classes and sometimes functions, which can be imported by corresponding import statements. For instance, the high-level functions are located in the particleShearRunSimulation module, so it is also possible to launch the simulation with default settings by:

```
from particleShearRunSimulation import *
doParticleShearSimulation()
```

At the folder level (Fig. S3-2b), each of the modules corresponds to its own subfolder in the package directory, and has its own source code files (.py) declaring the functions and classes associated with the specific module. In each subfolder, there is also an __init__.py file declaring the functions and classes available through the `import` statements. The subfolder « particleShear » contains an __init__.py file, but no other .py files; its goal is to enable the global import statement :

```
from particleShear import *
```

importing the public classes from all five modules.

The module hierarchy shown in Fig. S3-2 is reflected in the dependencies : a module depends on the modules below it, but not the ones above. For instance, the classes in particleShearLinkableObjects inherit directly or indirectly from classes in particleShearObjects and particleShearBase. On the other hand, they do not depend on elements in particleShearSimulation or particleShearRunSimulation.

5. Class hierarchy

We next describe the class hierarchy underpinning the Python implementation of the particle simulations provided here. Each class is physically located in a Python file (extension .py). The file name is identical to the class name, and the folder where it is stored within the package is given by the module name (Fig. S3-2b).

5.1. Spheres

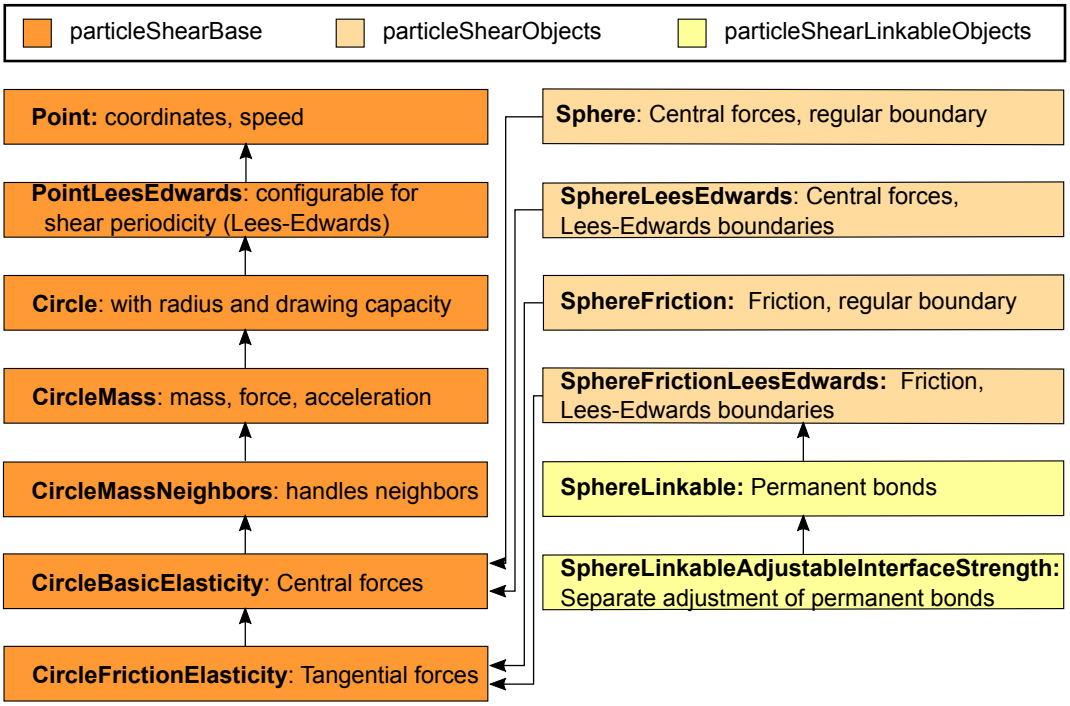

Fig. S3-3. Class hierarchy for the particle-objects. Arrows indicate class heritage, the colors code for the modules as defined in Fig. S3-2 and indicated in the legend.

The particleShear package contains a number of classes modeling the interacting particle objects ("spheres"; to be strict, in a 2D simulation, these correspond to infinite cylinders, shown as circles in horizontal cross section). The class hierarchy of these particle classes is shown in Fig. S3-3. The particle classes describe the properties of the interacting elementary spherical particles, the specificity and complexity increasing through the class heritage diagram. They are distributed among the modules particleShearBase, particleShearObjects, and particleShearLinkableObjects, as shown in Fig. S3-3.

The particle classes contained in particleShearBase and particleShearObjects implement the simulation by Otsuki et al.[1], with improved conservation of angular momentum (eq. S1-3 in Part 1) and the possibility to use a non-linear enhancement of the repulsion at large compression (eq. S1-5b). The particle classes in particleShearLinkableObjects allow the addition of permanent bonds between neighboring particles.

The classes named "Circle…" inherit from `PointLeesEdwards`. These classes are therefore all capable of handling the shear-periodic Lees-Edwards boundary conditions. They are however also able to handle ordinary, non-periodic boundary conditions, the behavior depending on the setting of the instance variables. On the contrary, the particles named "Sphere…" in modules particleShearObjects and particleShearLinkableObjects provide fixed behavior with regards to boundary conditions: `Sphere` and `SphereFriction` should be used with regular, non-periodic boundary conditions, whereas `SphereLeesEdwards`, `SphereFrictionLeesEdwards`, `SphereLinkable` and `SphereLinkableAdjustableInterfaceStrength` should be used with Lees-

Edwards boundary conditions. Regardless of their name, "Circle…" and "Sphere…" classes both describe 2D objects.

The default simulations launched by `doParticleShearSimulation` exclusively use the class `SphereLinkableAdjustableInterfaceStrength`. For generality, this class implements the possibility to have different spring and viscosity constants for permanent and transient interactions. In agreement with Otsuki et al.[1], this possibility was not used here (with the exception of some unit tests, see SI 4).

5.2. Assemblies

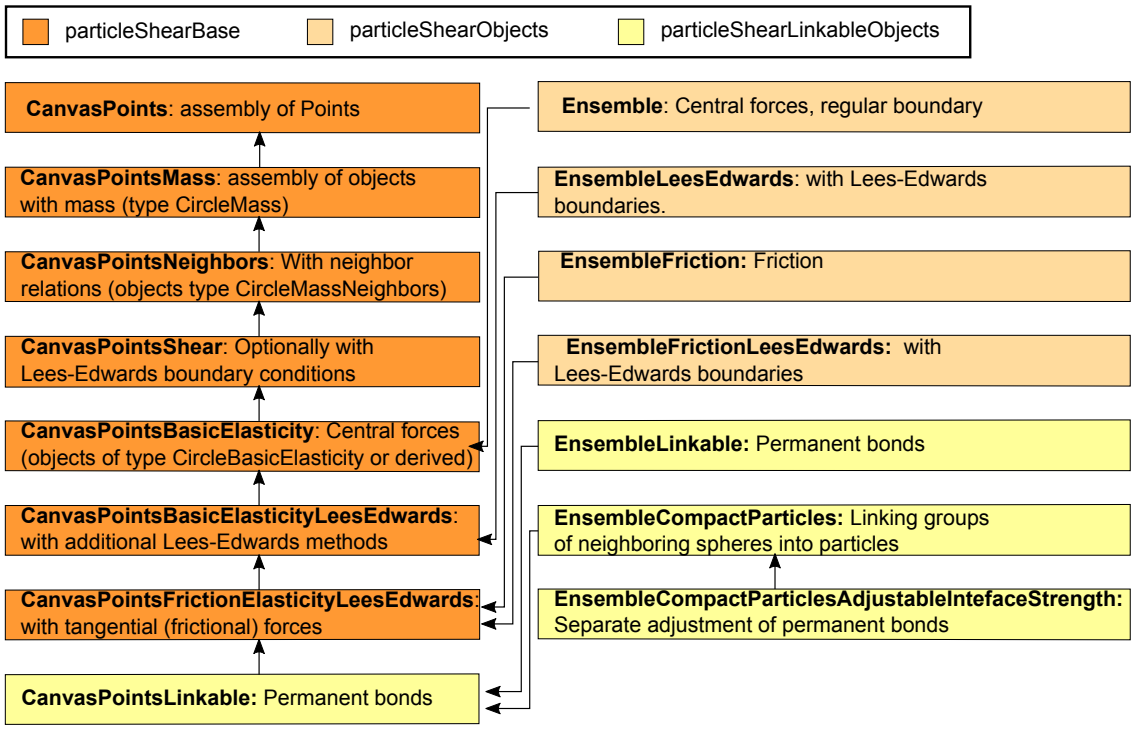

Fig. S3-4. Class hierarchy for the assembly objects. Arrows indicate class heritage, the colors code for the modules as defined in Fig. S3-2 and indicated in the legend.

The "Spheres" are assembled into particle assemblies. To handle the various particle types defined in the section above, we implement a similar hierarchy of assembly classes (Fig. S3-4). The common feature of all these assembly classes is that they hold a list of sphere objects, which are instances of one of the classes shown in Fig. S3-3. The "CanvasPoints…" classes are containers and do not by default contain any actual instances of particle classes. This role is reserved to the "Ensemble…" series, defined in modules particleShearObjects and particleShearLinkableObjects.

The simulations encoded in `doParticleShearSimulation` exclusively use the class `EnsembleCompactParticlesAdjustableInterfaceStrength`. However, by default, the spring and viscosity constants for permanent and transient interface are identical.

5.3. Neighbor relations

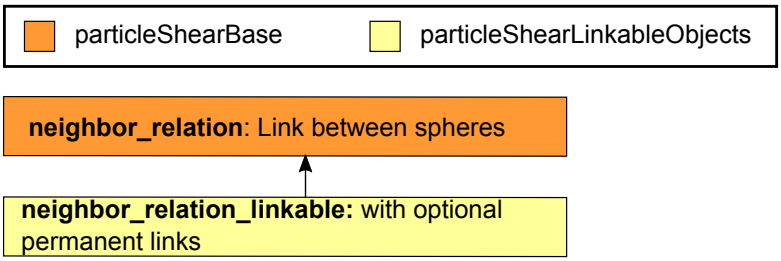

Fig. S3-5. Class hierarchy for the neighbor relation objects. Arrows indicate class heritage, the colors code for the modules as defined in Fig. S3-2 and indicated in the legend.

Within the assemblies, the particles have defined neighbors, be it by transient geometric contact or permanent bonds. The neighbor relations are known to the spheres, encoded by list of instances of the neighbor relation classes (Fig. S3-5).

5.3.1. Particle model

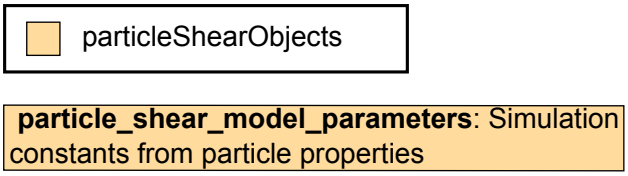

Fig. S3-6. Class hierarchy for the particle model classes. In the current version of the package, only one particle model class is implemented: particle_shear_model_parameters in module particleShearObjects.

To facilitate estimation of the local model properties of the particles (spring constants, mass, radius) from more physical properties (Young modulus, density, number of spheres, size of the simulation area), we defined a helper class `particle_shear_model_parameters` in pacakage particleShearObjects, as shown in Fig. S3-6.

5.4. Stress tensor evaluation

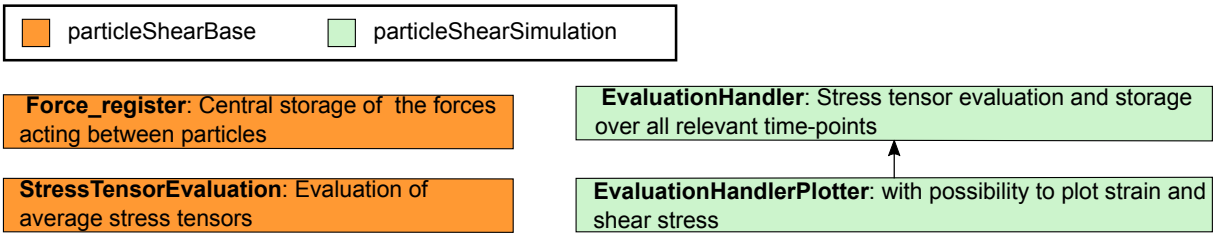

Fig. S3-7. Classes for stress tensor evaluation. Arrows indicate class heritage, the colors code for the modules as defined in Fig. S3-2 and indicated in the legend.

The classes participating in the evaluation of the stress tensors are shown in Fig. S3-7. The average stress tensor characterizes mechanical state of the particle ensemble under simulation. We implement two main stress tensor evaluation methods: From the external applied forces (eq. S1-16), and the overall stress tensor from the internal force and rotational interactions (eq. S1-24). Stress tensor evaluation at a given time-point is handled by specific a specific class: `StressTensorEvaluation`, which evaluates both equations. To perform stress tensor evaluation, a `StressTensorEvaluation` object evaluates the forces stored during the simulation in a force register object (`Force_register`).

The class `EvaluationHandler` and its derivatives make use of a `StressTensorEvaluation` object to evaluate the stress tensors and other aggregated properties. The `EvaluationHandler` objects store the results of the evaluation over all the measurement time points of the simulation. The `EvaluationHandlerPlotter` class additionally permits to plot the shear stress during the simulation, using eq. S1-25 to evaluate the relevant shear stress from the stored tensors.

5.5. Experiments

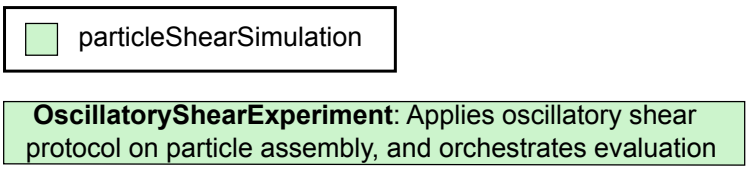

Fig. S3-8. Characterization experiments. Currently, OscillatoryShearExperiment is the only class implementing a mechanical characterization experiment.

Experiments are classes that allow to perform a pre-defined experimental protocol on a given particle assembly. Current, only one type of experiment is defined: rheological characterization by oscillatory shear, in class `OscillatoryShearExperiment`.

At instanciation, an `OscillatoryShearExperiment` object is provided with a particle assembly on which to perform the oscillatory shear experiment. After optional pre-equilibration, the `OscillatoryShearExperiment` object will apply a pre-defined shear protocol on the particle assembly, evaluating and registering the stress tensors at each time step by making use of an internal `StressTensorEvaluation` object. At the end of the experiment, elastic storage and viscous loss moduli are evaluated by demodulation of the shear stress (eq. S1-25, from stress tensors by eq. S1-16 and eq. S1-24) with the sinusoidal oscillatory strain excitation function (equations 14 and A1 of Otsuki et al.[1]). Optionally, the simulation parameters, the moduli, as well as the detailed shear stress and stress tensor values can be stored in an ASCII text file.

5.6. Simulations

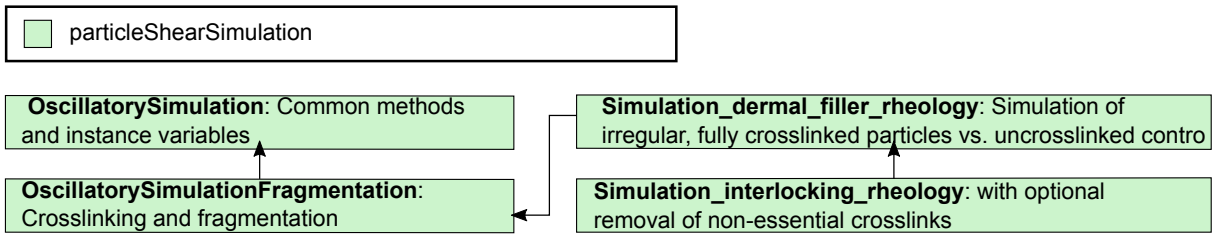

Fig. S3-9. Classes for particle shear simulations. Arrows indicate class heritage, the colors code for the modules as defined in Fig. S3-2 and indicated in the legend.

The simulation classes encode assembly of particle ensembles, along with their defined analysis by means of a virtual oscillatory shear experiment (as defined by the class `OscillatoryShearExperiment`).

The two base classes `OscillatorySimulation` and `OscillatorySimulationFragmentation` provide methods and instance variable definitions of general use, but are not intended to be used directly. Rather, the two derived classes `Simulation_dermal_filler_rheology` and `Simulation_interlocking_rheology` should be used as they define suitable particle ensembles.

The default simulations launched by the high-level function `doParticleShearSimulation` exclusively uses the class `Simulation_interlocking_rheology`.

5.7. Graphical output configuration

The class `Graphical_output_configuration` in module particleShearBase is used by the spheres and neighboring relation to adjust their graphical display.

6. Procedural elements

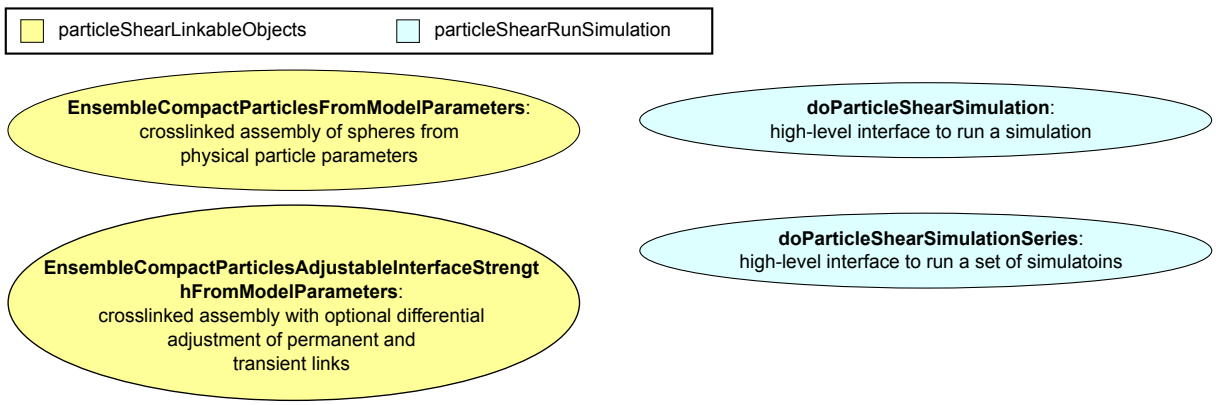

Fig. S3-10. Main procedural elements. The functions `EnsembleCompactParticlesFromModelParameters` *and* `EnsembleCompactParticlesAdjustableInterfaceFromModelParameters` *streamline the instanciation of the corresponding* `EnsembleCompactParticles` *and* `EnsembleCompactParticlesAdjustableInterface` *objects (see Fig. S3-4) using a model (class* `particle_shear_model_parameters`*, see Fig. S3-6). The* `doParticleShearSimulation` *function is the main high-level entry point;* `doParticleShearSimulationSeries` *allows to run a predefined series of simulation with different parameters.*

The main procedural elements are outlined in Fig. S3-10. The `EnsembleCompactParticlesFromModelParameters` and `EnsembleCompactParticlesAdjustableInterfaceFromModelParameters` functions are helper functions to facilitate generation of `EnsembleCompactParticles` respectively `EnsembleCompactParticlesAdjustableInterface` objects from macroscopic physical properties. The `doParticleShearSimulation` provides the high-level entry point to be used for launching parametrized default simulations (Table S1-1), wheres `doParticleShearSimulationSeries` launches a series of simulations from arrays of parameters.

7. Object relations

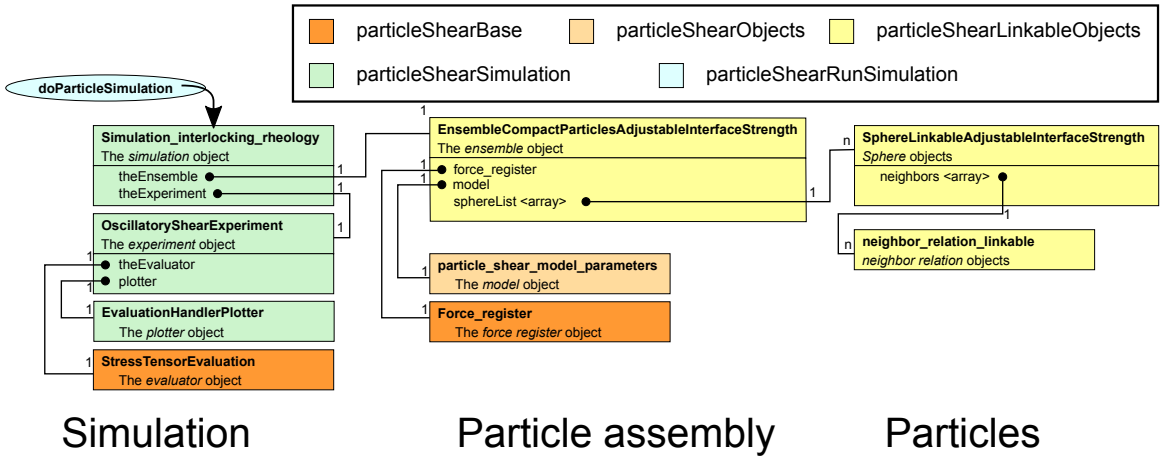

Fig. S3-11. Relation between the main objects. The primary entry point in basic usage is invocation of the high-level function `doParticleSimulation` *by the user or a script. The central object created is the simulation object (of class* `Simulation_interlocking_rheology`*). This object orchestrates the creation of all the other objects: The various objects related to simulation and evaluation ("Simulation" in the figure), the particle assembly ("Particle assembly" in the figure) and the particles and neighbor relations ("Particles" in the figure).*

Fig. S3-11 shows the main objects created during a simulation as triggered by a call to *doParticleSimulation*.

Various classes handle aspects of the simulation *per se*: a `Simulation_interlocking_rheology` object is created as the master object and directly or indirectly handles creation and interaction of all the other objects. An `OscillatoryShearExperiment` object handles the defined application of shear, as well as stress tensor evaluation via a `StressTensorEvaluation` object. The stress tensor matrices generated at different time-points throughout the simulation are stored, and optionally displayed graphically, by an `EvaluationHandlerPlotter` object.

The simulation is run on a particle assembly object of class `EnsembleCompactParticlesAdjustableInterfaceStrength`; for the creation of this object, a model is instantiated from the class `particle_shear_model_parameters`, used to estimate the simulation constants from physical ensemble parameters. During the simulation, the forces are stored in a force register, instantiated from `Force_register`.

The particle assembly object holds the constituent spheres in its attribute `sphereList`. At each instant of the simulation, each sphere knows its neighbors, by listing attributes of the relation to them in a `neighbors` array, which contains objects of the type `neighbor_relation_linkable`. While the number of spheres and permanent neighbor relations is constant during a given simulation, the non-permanent neighbor relations change both in number and identity as the spheres move.

With respect to the class hierarchy (Fig. S3-3 to S3-9), only a restricted set of the classes is used directly in the simulation (Fig. S3-11). The classes used are the classes at the end of the class heritage chains, as they contain the most specialized functionality.

Finally, some objects are referenced from multiple other objects. For example, the force register object is shown to be associated with the ensemble object in Fig. S3-11. It is however necessary for each sphere to signal the forces acting on it to the force register, so the force register object is also known to each sphere. Such secondary references are a convenience of programming and are not shown in Fig. S3-11 as it would make the diagram unnecessarily difficult to read.

8. Callback functions

To set custom contact force-distance laws between the interacting particles, the compressive, elastic part can be set arbitrarily with a callback function. For this, the static class variable

```
CircleBasicElasticity.call_back_elastic_force_law
```

has to be set appropriately. See section 4.3 of part 2 of this manual for further details.

9. Simulation process flow

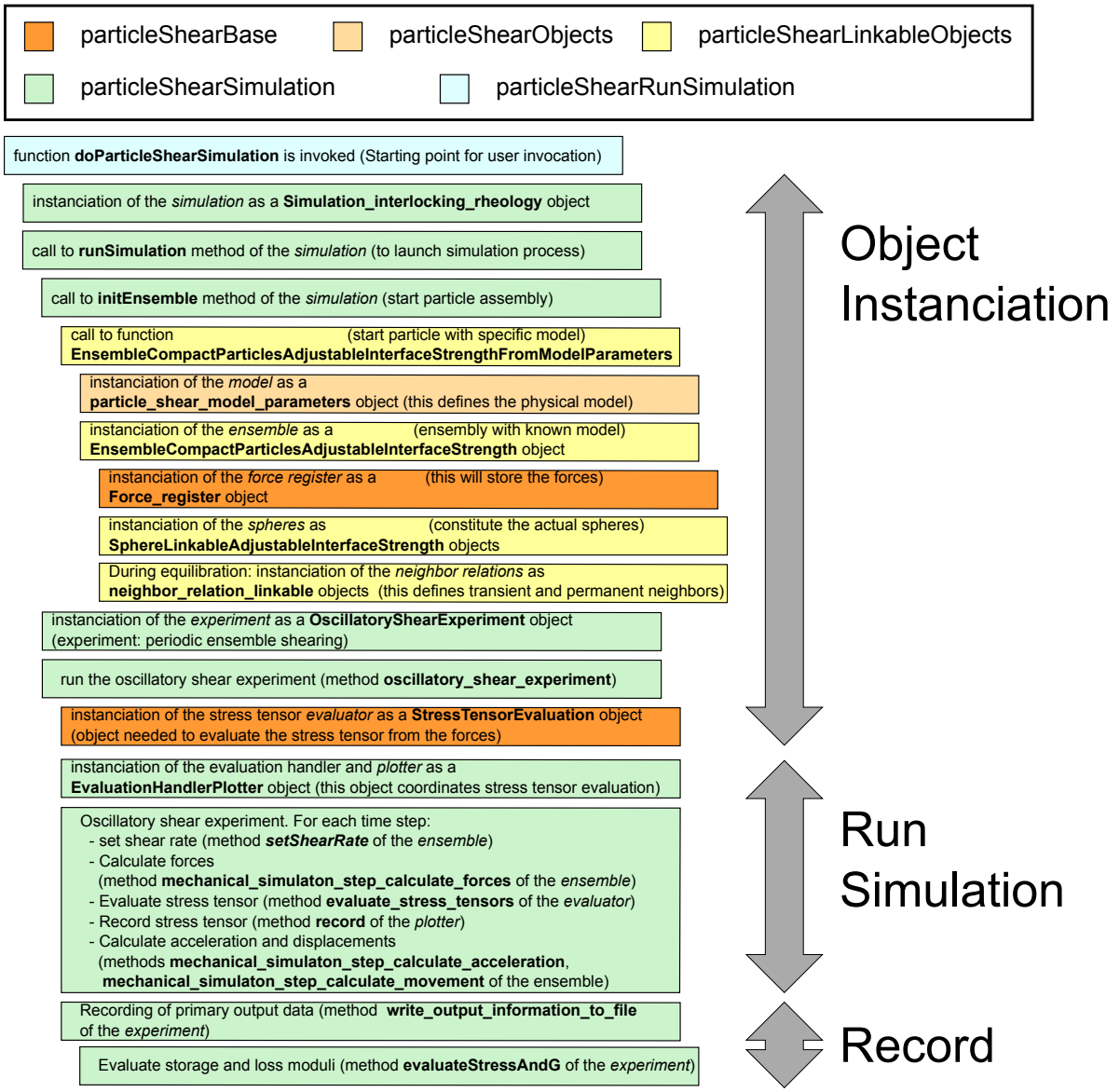

Fig. S3-12. Pseudo code for a particle shear simulation. Indentation indicates that the calls occur within blocks. The diagram is simplified as there are both many more levels and many more function calls; it outlines the most important steps, objects and methods.

Fig. S3-12 shows a simplified process flow (pseudocode) of a simulation as invoked by `doParticleShearSimulation`. The figure shows the most important function calls, object instanciations, and method calls.

Fig. S3-12 illustrates that the process flow can be grossly divided into three phases: first, the main objects shown in Fig. S3-11 are progressively initiated; once all the objects available, the actual oscillatory shear experiment is run. The summary and detailed stress tensor data can then optionally be recorded in a ASCII text file.

During the simulation, care needs to be taken about the evaluation order. The various tensors should be calculated with particle coordinates and forces valid for the same time

point. Thus, the order of execution should be as noted in Fig. S3-12 (phase "Run Simulation"): calculation of forces, evaluation of stress tensors, and only then updating of the particle positions.

10. Implementation of specific mathematical elements

To fully specify a software, it is not enough to indicate the algorithm in the form of mathematical equations. There are details of the implementation not evident from the concise notation, or implementation details influencing numerical precision, that are clear only from the source code itself. For this reason, Table S3-1 indicates the location of the implementation of some critical mathematical elements in the code.

What	Eq.	Module	Class	Method
Torque distribution	Eq. S1-3	particleShearBase	CircleFrictionElasticity	distribute_tangential_couple
Central force	Eq. S1-5a, S1-5b	particleShearBase	CircleBasicElasticity	get_elastic_force
Spring constant k	Eq. S1-8	particleShearObjects	particle_shear_parameters	__init__ (constructor)
Time constant τ	Eq. 1 in [1]	particleShearObjects	particle_shear_parameters	__init__ (constructor)
Transverse Spring constant k_T	Here, equal to k[1]	particleShearLinkableObjects	-	EnsembleCompactParticlesAdjustableInterfaceStrengthFromModelParameters in EnsembleCompactParticlesAdjustableInterfaceStrength.py (possibility to configure differently from k)
Viscosity constant η	Eq. S1-9	particleShearLinkableObjects	-	EnsembleCompactParticlesAdjustableInterfaceStrengthFromModelParameters in EnsembleCompactParticlesAdjustableInterfaceStrength.py ($\eta = k^* \tau$, k and τ from above)
Transverse Spring constant η_T	Here, equal to η	particleShearLinkableObjects	-	EnsembleCompactParticlesAdjustableInterfaceStrengthFromModelParameters in EnsembleCompactParticlesAdjustableInterfaceStrength.py (possibility to configure differently from η)
Damping	-	particleShearBase	CircleFrictionElasticity	cool (application of the damping for a time step)
		particleShearSimulaton	OscillatorySimulation	runSimulation (estimation of damping coefficient per dt from damping per period)
Radius r	Eq. S1-14	particleShearObjects	-	r_estimate in Ensemble.py
Mass m (per L)	$m = \pi\rho r^2$ ref. [1]	particleShearObjects	particle_shear_parameters	__init__ (average mass)
		particleShearObjects	-	adjust_m in Ensemble.py (scale to actual radius)
σ (external force)	Eq. S1-16	particleShearBase	StressTensorEvaluation	evaluate_force_stress_tensors

σ_{LW}	Eq. S1-19	particleShearBase	StressTensorEvaluation	evaluate_force_stress_tensors
σ	Eq. S1-24	particleShearBase	StressTensorEvaluation	evaluate_stress_tensors
τ_{xy}	Eq. S1-25	particleShearBase	StressTensorEvaluation	evaluate_externally_applied_shear_stress (stress tensor external force)
		particleShearBase	StressTensorEvaluation	evaluate_quasistatic_shear_stress (stress tensor Love-Weber[3])
G'	Eq. 14 in [1]	particleShearSimulaton	OscillatoryShearExperiment	demodulation
G''	Eq. A1 in [1]	particleShearSimulaton	OscillatoryShearExperiment	demodulation
Lees-Edwards boundaries[1]	-	particleShearBase	PointLeesEdwards	d (modified distance) n (modified relative vector) relative_speed (modified relative speed) boundary_conditions (shear periodicity)
Sllod stabilization[1]	-	particleShearBase	PointLeesEdwards	cool (damping of speed relative to anticipated local speed due to shear)
		particleShearBase	CanvasPointsBasicElasticityLeesEdwards	adjust_sphere_speed_to_shear_rate_change (adjust speed to anticipate change when changing shear rate)

Table S3-1. Implementation of specific mathematical elements. The table indicates where some central mathematical elements of the simulation are implemented in the code. For class methods, the module, class and method name jointly identify the location in the code. The classes are indeed defined in Python (.py) files named after the class name, and method names are unique within a given class. For procedural elements, there is no class (- sign in the table), but the relevant Python file is indicated along the method name to facilitate localization. Equations denoted with Eq. S1-XX are equations from Part 1.

11. Full documentation of the Application Programming Interface API

We generated a full API documentation by using pdoc, using the following commands (terminal, MacOSX):

cd <folder where the documentation should go>

pdoc --html --overwrite particleShear

The resulting HTML document is also available for Download (same folder as this document). pdoc and documentation in Python in general is particular because it uses introspection: the object's knowledge about itself. As a result, inherited methods are displayed as if they were implemented directly in the class at hand.

12. Bibliography

1 Otsuki, M. & Hayakawa, H. Discontinuous change of shear modulus for frictional jammed granular materials. *Phys Rev E* **95**, 062902, doi:10.1103/PhysRevE.95.062902 (2017).

2 Nicot, F., Hadda, N., Guessasma, M., Fortin, J. & Millet, O. On the definition of the stress tensor in granular media. *Int J Solids Struct* **50**, 2508-2517, doi:10.1016/j.ijsolstr.2013.04.001 (2013).

3 Love, A. E. H. *A Treatise on the Mathematical Theory of Elasticity*. (Cambridge University Press, 1927).

Part 4: Test cases for the numerical simulation

1. Aim of Part 4

Part 4 provides the test descriptions and results for key test scenarios for the particle ensemble simulation (used for Fig. 2 in the main text).

2. Content

3. Introduction

3.1. Unit tests

In simple cases, it is possible to estimate the stress tensors analytically. This allows testing of the simulation and the algorithms for evaluation of the stress tensors. Implementing the test scenario as unit tests also allows automated testing procedures, ensuring accurateness of the software throughout development.

Beyond these practical considerations, the availability of simple unit tests with *a priori* defined expected results also helped improving aspects of the simulation and stress tensor evaluation beyond published state of the art[1,2].

3.2. Improvements achieved with unit testing

3.2.1. Conservation of angular momentum

The unit tests indeed revealed residual non-conservation of angular momentum in the framework given by Otsuki et al.[2] and prompted the development of a more accurate expression for the distribution of the frictional torque (eq. S1-5b in Part 1). The error is small in the configuration described by Otsuki et al.[2], but can become limiting in the case of very large amplitude simulations, at the heart of our material development.

3.2.2. Inertial stress tensor correction

We initially based our simulations on the published algorithm by Otsuki et al.[2] Unit testing however revealed that the inertial correction term used (eq. 15 in ref. [2]), not only fails to correct the asymmetry of the Love-Weber stress tensor[1,3,4], but introduces artifacts in simple cases where it should be identically zero. The inertial correction term used by Otsuki et al. [2] historically stems dynamic pressure of ideal gases in containers at rest or nearly so,[5] where conceivably low mass flow speed and therefore approximate isotropy of the force network orientation will average out asymmetric contributions[6]. In a simulated linear shear cell with highly organized movement of particles this is not appropriate. A more general development is presented by Nicot et al.[1], serving as a starting point to develop inertial correction terms for the simulation at hand.

In the unit tests, the rotational inertia terms for spherical particles indicated by Nicot et al. (i.e. eq. 33 in ref. [1]) performed well in simple test cases. However, in the presence of friction, it also failed to provide symmetric overall stress tensors. In search for possible errors, we carried out a simplified mathematical development (eq. S1-20 to eq. S1-23, and the analytical verification leading to eq. S1-24 in Part 1). We found an expression for the rotational acceleration tensor σ_{spin} very similar, but not identical to eq. 33 by Nicot et al. [1]. We indeed found the same general terms and form, but a difference of a factor of 3/2 for the leading coefficient. With the corrected coefficient (eq. S1-24), we finally obtained symmetric stress tensors in the unit tests, to within numerical precision of the simulation (relative errors below 10^{-15}). Going back to the development by Nicot et al.[1], we identified a minor integration error in their calculation of the inertia matrix (eq. 29 in [1]; see Part 1) as the likely cause. Fixing this issue, we finally obtained overall stress tensors symmetric to within numerical precision. Hence, we can now state that not only, "the global result must be perfectly symmetrical"[1]; it actually is. Performing the unit

tests enabled challenging the otherwise overly complex frameworks with simple, intuitive cases with at least qualitatively known expected behaviours.

4. Test variables

We test the set of variables defined in Table S4-1.

Symbol	Evaluation measure
σ_{LW}	Love-Weber stress tensor `StressTensorEvaluation.stress_tensor_LW` eq. S1-19
σ	Overall internal stress tensor `StressTensorEvaluation.overall_stress_tensor` eq. S1-24
$\sigma_{\text{linear acceleration}}$	Stress tensor due to unbalanced linear forces `StressTensorEvaluation.stress_tensor_unbalanced_forces` Eq. S1-17
$\sigma_{\text{external force}}$	Stress tensor estimated from external forces `StressTensorEvaluation.stress_tensor_with_external_forces` (eq. S1-16)

σ_{Otsuki}	Inertial term in Otsuki et al.[2]
	`StressTensorEvaluation.`
	`stress_tensor_linear_acceleration_otsuki`
	(second right-hand term of eq. 15 in [2])

$\int \mathrm{d}m \cdot \vec{r} \times \vec{v}$	Conservation of angular momentum `EvaluationHandler.angular_momentum_around_origin`
$\int \mathrm{d}m \cdot \vec{v}$	Conservation of linear momentum `EvaluationHandler.total_mass*EvaluationHandler.v_mean`

Table S4-1. Test variables for unit tests. The table indicates the mathematical symbol in the first column. A short description as well as the implementation variable are given in the second column. The implementation is indicated as `Classname.Field_name`; the classes of the implementation are listed in Part 3.

5. Implementation and usage

The test cases are implemented in a separate sub-package, particleShearTest, that is not accessible directly via the general import. Should direct use be required, it can be imported via:

```
from particleShearTest import *
```

Generally, the main use of the unit test is to run all of them sequentially. After installation of the package particleShear and its dependencies (see Part 2), the tests can for instance be run via the following command:

```
python -m unittest particleShearTest
```

This should automatically run all the tests in the particleShearTest folder. Alternatively, each script in the particleShearTest can also be run manually.

Currently, 53 out of 55 unit tests pass, 2 fail. The reason for the two failures is that for reasons of comparison, the simulations optionally implement the inertial correction for the stress tensor given by Otsuki et al. [2] (eq. 15 in [2]). At least in our interpretation, this correction gives erroneous non-zero stress tensors in some cases where there should be no forces at all (see Table S4-2 and Table S4-4 below).

6. Test systems without boundary interaction

A first series of test cases consists of isolated systems without boundary interaction or boundary particles. Under these conditions, angular and linear momentum needs to be conserved, and given the absence of external force, $\sigma_{\text{external force}}$ should remain zero..

6.1. Single Sphere

A first elementary ensemble consists single sphere, moving freely without any external applied force. We consider two different cases: a single sphere in purely linear movement, and a single sphere spinning but not moving linearly.

6.1.1. Linearly moving single sphere

The test case of a single, linearly moving sphere is depicted in Fig. S4-1.

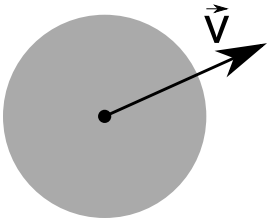

Fig. S4-1. Test case of a single, freely moving sphere. No force is applied, and the sphere proceeds by linear, constant movement along its speed vector $\vec{v}$. For this test case, all stress tensor components should be identically 0, and linear and angular momentum conserved.

For a freely moving sphere (Fig. S4-1), all the stress tensor expressions need to be identically zero for all elements. In order to avoid particular cases with movement along the axes, we set both the x- and y- speed non-zero, and different from each other.

We evaluate the various stress tensor components after 20 steps of movement of the sphere and assess whether they remain zero. After a further 20 steps of simulation, we also evaluate whether the overall angular and linear momentum have changed. The results are given in Table S4-1. All tests pass, with the exception of the inertial correction term proposed by Otsuki et al. [2], which gives a finite, non-zero stress tensor value in a situation where all the terms should be identically equal to zero. It is not clear here whether this is an implementation or interpretation error or whether there is a problem with the actual term proposed by Otsuki et al. [2]. Given the satisfactory

performance in the unit test, we therefore rather rely on the inertial tensor expressions adapted from Nicot et al.[1] (eq. S1-24).

Of note, for angular momentum conservation, we only evaluate a short time period before shear periodicity (Lees-Edwards periodic boundary conditions) causes sudden apparent displacements of the sphere incompatible with a naïve implementation of angular momentum conservation.

Scenario	Test	Expected	Actual	Pass
Translating single sphere	σ_{LW}	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	σ	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	$\sigma_{\text{linear acceleration}}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	$\sigma_{\text{external force}}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	σ_{Otsuki}	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Non-zero, with positive and negative diagonal elements, symmetric	No
	$\int \mathrm{d}m \cdot \vec{r} \times \vec{v}$	constant	Constant (to within relative error of 10^{-15})	Yes
	$\int \mathrm{d}m \cdot \vec{v}$	constant	Constant (to within relative error of 10^{-15})	Yes
Script: particleShearTest/test_singleSphereLinear.py				

Table S4-2. Unit testing results for the scenario of a single, force-free sphere where all stress tensors should evaluate to zero, and where angular and linear momentum should be conserved. The tests are defined in Table S4-1 and here symbolized by their mathematical symbol given in Table S4-1.

6.1.2. Freely spinning single sphere

Fig. S4-2. Test case of a single, freely rotating sphere. No force is applied, and the sphere proceeds by indefinite rotation at an angular rotation rate ω. For this test case, the external force and Love-Weber stress tensor (eq. S1-16 and eq. S1-19) should be zero, but the total stress tensor (eq. S1-24) should reflect the tensile state due to the presence of the centrifugal forces (eq. S1-21).

In a freely spinning sphere (Fig. S4-2), there is an internal stress due to the centrifugal force associated with the rotation.

However, for as long as the rotating sphere does not touch the boundaries, its rotation energy will not be transmitted to the outside world, and so the stress tensor defined by the external forces (eq. S1-16) should not be affected and all its components should

remain identically zero, as for the linearly moving sphere. Likewise, the Love-Weber stress tensor (eq. S1-19) is known to reflect only interactions between particles, but not internal acceleration[1], so it should also remain identically at 0. Only the total stress tensor according to eq. S1-24 should yield non-zero diagonal elements, according to eq. S1-21.

As shown in Table S4-3, the simulation passes all the unit tests for this scenario.

Scenario	Test	Expected	Actual	Pass
Rotating single sphere	σ_{LW}	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	σ	$-\frac{K}{V}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$ eq. S1-21	Relative error <10^{-15}	Yes
	$\sigma_{\text{linear accelera}}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	$\sigma_{\text{external force}}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	σ_{Otsuki}	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	$\int \mathrm{d}m \cdot \vec{r} \times \vec{v}$	constant	Constant (to within relative error of 10^{-15})	Yes
	$\int \mathrm{d}m \cdot \vec{v}$	constant	Constant (to within relative error of 10^{-15})	Yes
Script: particleShearTest/test_singleSphereRotation.py				

Table S4-3. Unit testing results for the scenario of a single, force-free rotating sphere where all stress tensors except for the centrifugal-force related part (eq. S1-21) should be zero.

6.2. Interaction between two spheres

Next, we test the interaction of two spheres and the resulting stress tensors.

6.2.1. Central forces: Non-frictional collision between two spheres

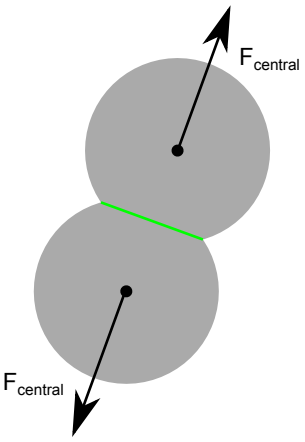

Fig. S4-3. Test case of a pair of spheres with purely central interaction. For this test case, there are no external forces, so the stress tensor from external forces should be zero (eq. S1-16), whereas the Love-Weber stress tensor should be symmetric (eq. S1-24) with positive diagonal terms. The overall internal stress tensor should be equal to the Love-Weber stress tensor since there is no rotation due to the absence of tangential forces.

We first test the central interaction in a test case where two spheres collide by linear movement. For this test case (Fig. S4-3), we can anticipate some characteristics of the various stress tensor expressions. For as long as the spheres do not touch the boundaries, the external force tensor $\sigma_{\text{external force}}$ should be identically zero as there is no transmission of force to the outside world. As there is no friction in this test scenario, the Love-Weber stress tensor (eq. S1-19) should be symmetrical; since the interaction is compressive, its diagonal terms should be positive (Soil mechanics sign convention[7]). Without tangential forces, there should be no rotation, so σ_{spin} should be zero. One then concludes that $\sigma = \sigma_{LW} = \sigma_{\text{external force}} + \sigma_{\text{linear acceleration}} = \sigma_{\text{linear acceleration}}$ and so the linear acceleration stress tensor can be tested against the numerical value of the Love-Weber stress tensor in this test case. Linear and angular moment should still be conserved.

Without spinning motion, the Love-Weber stress tensor is complete and there should be no need for additional inertial correction[1]. This implies that also the inertial correction proposed by Otsuki et al.[2] should be zero in this test case.

Scenario	Test	Expected	Actual	Pass
Collision without friction	σ_{LW}	Strictly symmetrical Positive diagonal elements	Strictly symmetrical Positive diagonal elements	Yes
	σ	Identical to Love-Weber	Relative error <10^{-15}	Yes
	$\sigma_{\text{linear acceleration}}$	Identical to Love-Weber	Relative error <10^{-15}	Yes
	$\sigma_{\text{external force}}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	σ_{Otsuki}	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Non-zero, with positive and negative diagonal elements, symmetric	No
	$\int \mathrm{d}m \cdot \vec{r} \times \vec{v}$	constant	Constant (to within relative error of 10^{-15})	Yes
	$\int \mathrm{d}m \cdot \vec{v}$	constant	Constant (to within relative error of 10^{-13})	Yes
Script: particleShearTest/test_twoSpheresCentral.py				

Table S4-4. Unit testing results for the scenario of a collision with purely central forces between two identical spheres.

The unit tests shown in Table S4-4 for the central interaction between two colliding spheres are all successful, with the exception of the test for the inertial term proposed by Otsuki et al.[2]. This confirms the results from the test case with a linearly moving sphere (Table S4-2).

6.2.2. Tangential force by frictional contact: Frictional pair of spheres without repulsion

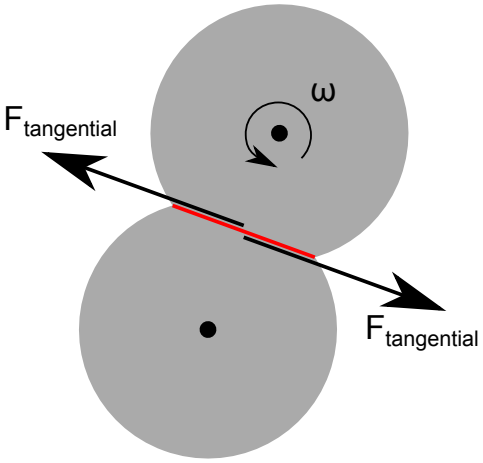

Fig. S4-4. Test case of a pair of spheres with purely tangential interaction. For this test case, there are no external forces, so the stress tensor from external forces should be zero (eq. S1-16). The Love-Weber stress tensor is generally asymmetric, whereas the overall internal stress tensor σ should by strictly symmetrical[1].

Scenario	Test	Expected	Actual	Pass
Purely frictional interaction	σ_{LW}	Asymmetric, value $\frac{1}{V}\vec{F}_T\overrightarrow{\Delta r}^T$	Relative error <10^{-15}	Yes
	σ	Symmetrical	Relative error <10^{-15}	Yes
	$\sigma_{\text{linear acceleration}}$	Identical to Love-Weber	Relative error <10^{-15}	Yes
	$\sigma_{\text{external force}}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	σ_{Otsuki}	?	Non-zero, with positive and negative diagonal elements, symmetric	N/A
	$\int \mathrm{d}m \cdot \vec{r} \times \vec{v}$	constant	Constant (to within relative error of 10^{-14})	Yes
	$\int \mathrm{d}m \cdot \vec{v}$	constant	Constant (to within relative error of 10^{-15})	Yes
Script: particleShearTest/test_twoSpheresTangential.py				

Table S4-5. Unit testing results for the scenario of a collision with purely tangential forces between two identical spheres.

For the test case of a interaction between two identical spheres with purely tangential interaction (Fig. S4-4), we can again anticipate some characteristics of the stress tensor expression. Given the single force couple present, σ_{LW} is given by $\sigma_{LW} = \frac{1}{V}\vec{F}_T\overrightarrow{\Delta r}^T$; generally, this is neither symmetric nor antisymmetric. After correction for the spin terms (eq. S1-24), the overall stress tensor σ should be symmetric; as before, since there is no external force, $\sigma_{\text{external force}} = 0$, and as consequence, $\sigma_{\text{linear acceleration}}$=$\sigma_{LW}$. Table S4-5 shows the results of the unit tests, they pass.

Given the failures of the Otsuki inertial compensation in term in the single sphere unit tests (Table S4-2, Table S4-4), we find it difficult to anticipate a value for this expression, so we report it, but do not test it (entry N/A in the Pass column in Table S4-5).

6.2.3. Collision of two spheres with general interaction

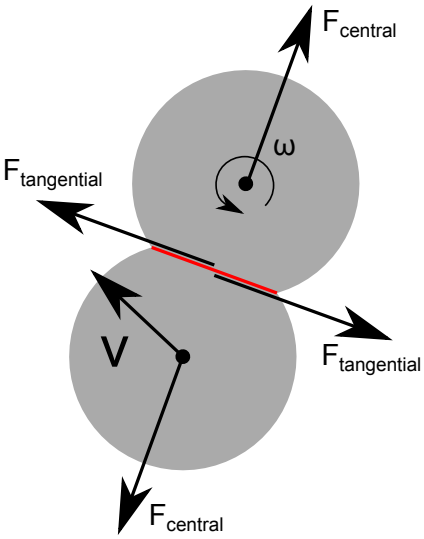

Fig. S4-5. Test case of a pair of spheres with general interaction. For this test case, there are no external forces, so the stress tensor from external forces should be zero (eq. S1-16). The Love-Weber stress tensor is generally asymmetric, but should be dominated by the compressive state simulated (positive diagonal elements under the soil mechanics sign convention[7]). The overall internal stress tensor σ should by strictly symmetrical[1].

Scenario	Test	Expected	Actual	Pass
Collision with general interaction	σ_{LW}	In the configuration chosen, compressive state, so diagonal values should be positive	Asymmetric, diagonal values positive	Yes
	σ	Symmetrical Diagonal values positive	Relative error <10^{-15}	Yes
	$\sigma_{\text{linear acceleration}}$	Identical to Love-Weber	Relative error <10^{-15}	Yes
	$\sigma_{\text{external force}}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	σ_{Otsuki}	?	Non-zero, symmetric	N/A
	$\int \mathrm{d}m \cdot \vec{r} \times \vec{v}$	constant	Constant (to within relative error of 10^{-15})	Yes
	$\int \mathrm{d}m \cdot \vec{v}$	constant	Constant (to within relative error of 10^{-14})	Yes
Script: particleShearTest/test_twoSpheresGeneral.py				

Table S4-6. Unit testing results for the scenario of a collision of two identical spheres with general interaction forces.

Fig. S4-5 shows a collision between two spheres with general interaction terms. There are four types of forces (see Fig. S1-1 in Part 1): viscous and elastic central forces, as well as viscous and elastic tangential forces forces. The interaction set is therefore complete as far as the numerical simulations presented here are concerned. We do not foresee any *a priori* values, but can still test the form of the tensors. σ_{LW} is generally asymmetric, but the overall internal stress tensor σ should be symmetric. Since $\sigma_{\text{external force}} = 0$ for this isolated system, $\sigma_{\text{linear acceleration}} = \sigma_{LW}$ should hold. In addition, conservation of angular and linear momentum are expected. Further, we chose an interaction configuration with significant compression, oriented obliquely compared to

the coordinate system, implying that the diagonal terms of σ_{LW}, σ and $\sigma_{\text{linear acceleration}}$ should all be positive with the soil mechanics sign convention[7].

The results comply with the expectations set forth (Table S4-6). Again, we are at odds for predicting a value for the Otsuki inertial term, so we do not test it (entry N/A in the Pass column in Table S4-6).

7. Boundary interactions

For bounded systems, there are generally trans-border forces, and so conservation of angular and linear momentum for the internal system are no more guaranteed. We therefore drop these tests from the test panel for the unit tests on bounded systems. Further, since in our implementation the inertial term proposed by Otsuki et al.[2] fails some of the key unit tests (Table S4-4, Table S4-2), we do not consider it further here.

We consider two elementary boundary interactions: one with a fixed sphere (as would happen by interaction with particles immobilized to an external rheometer device), and through the shear-periodic Lees-Edwards boundary conditions.

7.1. Single sphere colliding with fixed sphere

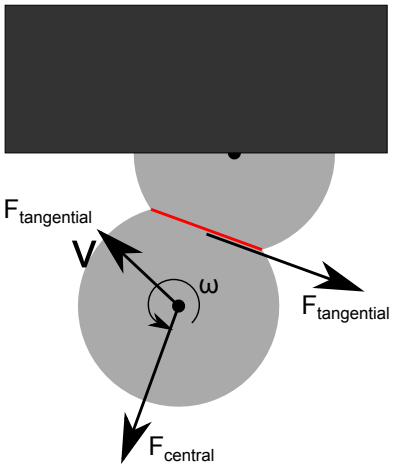

Fig. S4-6. Test case of a mobile sphere interacting with a fixed sphere on the boundary. For this test case, there are no strictly internal forces or torques, since only the mobile sphere is considered as being part of the system under study, whereas the fixed sphere is part of the outside world. As a result, the Love-Weber stress tensor should be zero (eq. S1-16). Likewise, there are no internal moments, so according to eq. S1-24, the overall internal stress tensor σ should be diagonal with negative contributions due to the centrifugal term. The black area denotes an outside-world device holding the immobilized sphere in place, such as the plate of a rheometer.

First, we consider a mobile sphere colliding with a single sphere fixed to the boundary. The fixed sphere is considered as belonging to the outside world, whereas the mobile sphere is the system under study. This configuation is reminiscent of single kinetic gas particle system.

Regarding the stress tensor, different approaches can be taken in such a system: the kinetic theory of gases ascribes a pressure to the kinetic force exerted by the gas particles colliding with the walls and thus a non-zero stress tensor value for the gas. In the stress tensor theory of granular material on the other hand, the average stress tensor is obtained by subtracting particle acceleration from the external force[1]. In this view purely kinetic pressure arising from particle collisions with the wall would be considered an inertial, rather than proper elastic phenomenon; only internal collisions, which are absent in single mobile particle system, would give rise to an internal stress tensor different from zero. We follow this view for the test case at hand, and thus expect an overall internal stress tensor of zero.

A similar question arises for the stress tensors arising from particle rotation. Kinetic gaz theory does not consider stresses internal to the gaz particles (arising for instance from molecular rotation), and thus would ignore the contribution of particle rotation to the stress tensor. Like before, we follow here the approach in granular material mechanics[1] and do consider the tensile contribution of particle rotation via $\sigma_{\text{centrifgual}}$ (eq. S1-21).

Formally, as there are no internal force couples, σ_{LW} should be identically zero. Further, as all the external force transmitted to the single internal sphere results in acceleration of the sphere, we should have $\sigma_{\text{linear acceleration}} = -\sigma_{\text{external force}}$. There will also be an exchange of torque, but as no strictly internal torque arises, σ should be given by $\sigma_{\text{centrifgual}}$ (eq. 24 for T_{int}=0). Hence, the overall stress tensor σ should be diagonal with negative, identical diagonal elements.

As shown in Table S4-7, the simulation passes the relevant unit tests.

Scenario	Test	Expected	Actual	Pass
Collision with fixed boundary sphere	σ_{LW}	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$	Yes
	σ	Diagonal with negative diagonal entries (due to σ_{spin})	Diagonal with negative, identical entries	Yes
	$\sigma_{\text{linear acceleration}}$	No a priori value		N/A
	$\sigma_{\text{external force}}$	$-\sigma_{\text{linear acceleration}}$	relative error < 10^{-15}	Yes
Script: particleShearTest/test_oneSphereFreeOneSphereBoundary.py				

Table S4-7. Unit testing results for the scenario of a collision of a single mobile sphere with a fixed boundary sphere.

7.2. Pair of spheres interacting through Lees-Edwards periodicity

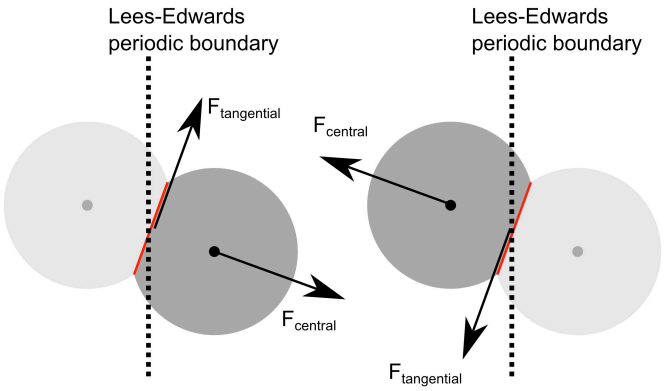

Fig. S4-7. Test case of a pair of mobile spheres interacting with virtual copies arising through Lees-Edwards boundary periodicity. For this test case, there are no strictly internal forces or torques, since we consider interactions through the Lees-Edwards boundaries to be approximations to interactions with a large, continuous medium. The forces arising through the periodic interaction are therefore considered as external forces. As a result, the Love-Weber stress tensor should be zero (eq. S1-16). Likewise, there are no internal moments, so according to eq. S1-24, the overall internal stress tensor σ should be diagonal with negative contributions due to the centrifugal term.

By virtue of the periodicity of the Lees-Edwards boundary conditions, spheres near a given boundary can interact with sphere located near the opposite boundary of the simulation region (Fig. S4-7). This emulates an infinite assembly by repetition. We consider the interactions across such periodic Lees-Edwards boundaries as external interaction, as they represent an approximation to interactions to neighboring spheres outside the actual simulation region.

In this view, a pair of spheres interacting through a Lees-Edwards periodic boundary represents the interactions the two spheres have with the outside world. For this reason, we count all the forces in this test case as being externally applied forces; the same holds for the torques. As a consequence, there no strictly internal forces present in the system and the σ_{LW} should be identically zero. The only non-zero term for the overall stress tensor σ arises from the spinning motion of the spheres, and more specifically the centrifugal for term $\sigma_{\text{centrifugal}}$, imparting negative, identical diagonal terms and zero-value off-diagonal terms. As before, $\sigma_{LW} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ implies $\sigma_{\text{external force}} = -\sigma_{\text{linear acceleration}}$. Table S4-8 shows the unit testing results based on these anticipated characteristics of the stress tensors. The simulation passes the unit tests.

Scenario	Test	Expected	Actual	Pass
Interaction across Lees-Edwards boundary	σ_{LW}	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	Yes
	σ	Diagonal with negative diagonal entries (due to $\sigma_{\text{centrifugal}}$)	Diagonal with negative, identical entries	Yes
	$\sigma_{\text{linear acceleration}}$	No a priori value		N/A
	$\sigma_{\text{external force}}$	$-\sigma_{\text{linear acceleration}}$	relative error < 10^{-15}	Yes
Script: particleShearTest/test_twoSpheresLeesEdwards.py				

Table S4-8. Unit testing results for the scenario of general interaction of two spheres across the Lees-Edwards boundaries.

8. Assemblies

8.1. Assembly of spheres

8.1.1. Equilibrated spherical particle assembly without shear

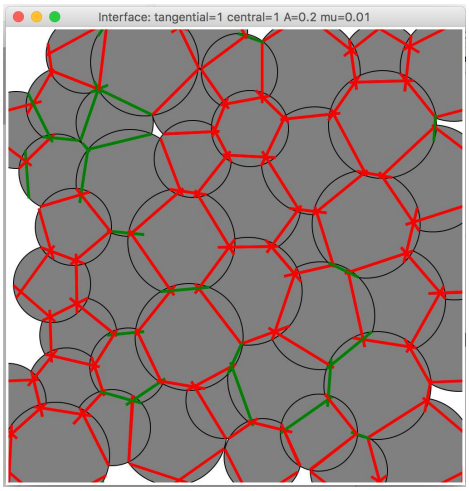

Fig. S4-8. Test case of a small particle ensemble (N=45) with frictional and central interaction, after minimal pre-equilibration. The ensemble is under compression (packing fraction 1.5), with Lees-Edwards boundary conditions but without externally applied shear.

In this test scenario, we constitute a small assembly for spherical particles (N=45), under default simulation conditions (see Table Table S2-1 in Part 2). After a short pre-equilibration period, we assess the stress tensor values as listed in Table S4-9. We keep the particle number low and pre-equilibration time short to restrict the use of computation ressources; in larger ensembles with longer equilibration time as used in the actual simulations, the accuracy would be higher, especially regarding the reduction of $\sigma_{\text{linear acceleration}}$ by pre-equilibration.

Scenario	Test	Expected	Actual	Pass
Pre-equilibrated spherical microgel assembly	σ_{LW}	Positive diagonal elements; in standard simulations conditions, diagonal elements > 0.5 (0.5kPa) off-diagonal elements smaller in absolute value: $\lvert\sigma_{LW(0,1)}\rvert < 0.1\sigma_{LW(0,0)}$ and $\lvert\sigma_{LW(1,0)}\rvert < 0.1\sigma_{LW(0,0)}$	Complies	Yes
	σ	Symmetrical: $\lvert\sigma_{(0,1)} - \sigma_{(1,0)}\rvert <$	Complies	Yes

		$10^{-15} \cdot (\lvert\sigma_{(0,0)}\rvert + \lvert\sigma_{(1,1)}\rvert)$ Positive diagonal elements; in standard simulations conditions, diagonal elements > 0.5 (0.5kPa)		
	$\sigma_{\text{linear acceleration}}$	Values of all four entries less than 10% of $\lvert\sigma_{(0,0)}\rvert$	Complies	Yes
	$\sigma_{\text{external force}}$	For diagonal elements: less than 10% difference from corresponding σ values	Complies	Yes
Script: particleShearTest/test_simulationFreeSpheres.py				

Table S4-9. Unit testing results for the scenario of an assembly of spherical microgel particles.

8.1.2. Equilibrated spherical particle assembly during application of shear

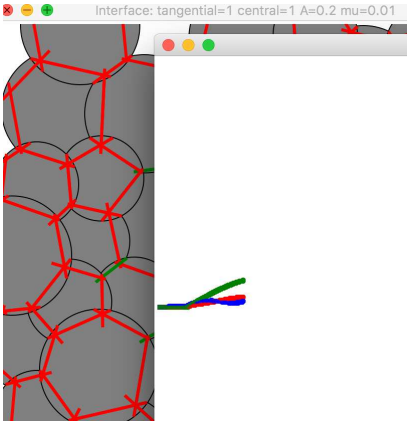

Fig. S4-9. Test case of a small particle ensemble (N=45) with frictional and central interaction, after minimal pre-equilibration and during application of shear. The ensemble is under compression (packing fraction 1.5), with Lees-Edwards boundary conditions.

This scenario is identical to the previous one, except for that we initiate application of a 20% relative deformation oscillatory shear cycle (Fig. S4-9). We interrupt the shear cycle after shortly after the maximum shear deformation. As the spherical suspensions cannot sustain such shear loads without yielding, the results should not fundamentally differ from the previous scenario. The symmetry constraints should be maintained. As shown in Table S4-10, the simulation passes these unit tests.

Scenario	Test	Expected	Actual	Pass
Pre-equilibrated spherical microgel assembly	σ_{LW}	Positive diagonal elements; in standard simulations conditions, diagonal elements > 0.5 (0.5kPa)	Complies	Yes
	σ	Symmetrical: $\lvert\sigma_{(0,1)} - \sigma_{(1,0)}\rvert < 10^{-15} \cdot (\lvert\sigma_{(0,0)}\rvert + \lvert\sigma_{(1,1)}\rvert)$ Positive diagonal elements; in standard simulations	Complies	Yes

		conditions, diagonal elements > 0.5 (0.5kPa)		
	$\sigma_{\text{linear acceleration}}$	Values of all four entries less than 10% of $\lvert\sigma_{(0,0)}\rvert$	Complies	Yes
	$\sigma_{\text{external force}}$	For diagonal elements: less than 10% difference from corresponding σ values	Complies	Yes
Script: particleShearTest/test_simulationFreeSpheresShear.py				

Table S4-10. Unit testing results for the scenario of an assembly of spherical microgel particles under a 20% oscillatory shear load.

8.1.3. Equilibrated crosslinked particle assembly during application of shear

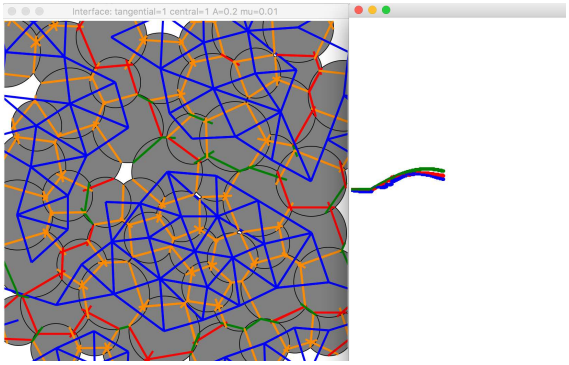

Fig. S4-10. Test case of a small crosslinked particle ensemble (N=60) with frictional and central interaction, after minimal pre-equilibration and during application of shear. The ensemble is under compression (packing fraction 1.5), with Lees-Edwards boundary conditions.

In this scenario, we crosslink the spheres into distinct particles. We again initiate application of a 20% relative deformation oscillatory shear cycle, with interruption shortly after the maximum shear deformation. Due to the presence of the irregular particles, the suspension is capable of sustaining some elastic load, and we anticipate that the off diagonal elements rise. However, the symmetry constraints should be maintained. As shown in Table S4-11, the simulation passes these unit tests.

Scenario	Test	Expected	Actual	Pass
Pre-equilibrated spherical microgel assembly	σ_{LW}	Positive diagonal elements; in standard simulations conditions, diagonal elements > 0.5 (0.5kPa)	Complies	Yes
	σ	Symmetrical: $\lvert\sigma_{(0,1)} - \sigma_{(1,0)}\rvert < 10^{-15} \cdot (\lvert\sigma_{(0,0)}\rvert + \lvert\sigma_{(1,1)}\rvert)$ Positive diagonal elements; in standard simulations conditions, diagonal elements > 500 (i.e. 0.5kPa)	Complies	Yes
	$\sigma_{\text{linear acceleration}}$	Values of all four entries less than 10% of $\lvert\sigma_{(0,0)}\rvert$	Complies	Yes

	$\sigma_{\text{external force}}$	For diagonal elements: less than 10% difference from corresponding σ values	Complies	Yes
Script: particleShearTest/test_simulationCrosslinkedShear.py				

Table S4-11. Unit testing results for the scenario of an assembly of spherical microgel particles under a 20% oscillatory shear load.

9. Simulations under actual conditions

The actual simulations are more time consuming, so we do not include them into the battery of unit tests. However, the scenario of crosslinked particles described above is complete, with the exception of a smaller sphere number and shortened equilibration protocols. Given the compliance of the software with the unit test requirements under these conditions, we are confident that the stress tensor values are evaluated correctly also for larger particle numbers and longer equilibration times, as these parameters are not per se linked to the simulation and evaluation algorithms.

9.1. Stress tensor output in text files

It is possible to store the stress tensor values at all time-points of the simulation in ASCII text files, allowing for a posteriori verification of the various stress tensor expressions among them, and for the symmetry of the overall stress tensor.

The overall stress tensor components are stored under the headings "stress_tensor_overall_00" to "stress_tensor_overall_11" (see Part 2, Table S2-3). The symmetry of the overall stress tensor implies that the entries in the columns "stress_tensor_overall_01" and "stress_tensor_overall_10" should be identical for all time points.

9.2. Relations among stored stress tensor values

Beyond the symmetry of the overall stress tensor, the following relations can be used to check the integrity of the evaluated stress tensors *a posteriori*:

Rewriting eq. S1-24 using the column headers in the output text file (see Part 2, Table S2-3):

```
stress_tensor_overall = stress_tensor_Love_Weber +
stress_tensor_centrifugal +
stress_tensor_internal_tangential_torque
```

This stress tensor relation holds for every component separately, so for instance for the 11 component, it translates to:

```
stress_tensor_overall_11 = stress_tensor_Love_Weber_11 +
stress_tensor_centrifugal_11 +
stress_tensor_internal_tangential_torque_11
```

Analogous relations hold for the 00, the 01 and the 10 component.

Rewriting eq. S1-19, again with the column headers in the output text files, we also have:

```
stress_tensor_Love_Weber = stress_tensor_from_external_forces
+ stress_tensor_linear_acceleration
```

where, as for eq. S1-19, we neglect gravity. The relation again holds for each of the four components separately.

Testing symmetry of the overall stress tensor and these relations on the numerical output allows some a posteriori assessment of the consistency of the values obtained. One has to bear in mind that due to numerical inaccuracy inherent in digital representation of numbers, there will be some inaccuracy, and neither symmetry nor the relations given by eq. S1-19 and eq. S1-24 will be exact. The error should however be close to numerical representation accuracy, so that relative errors will be very low (on the order of 10^{-13} or better with current Python implementations).

10. Bibliography

1 Nicot, F., Hadda, N., Guessasma, M., Fortin, J. & Millet, O. On the definition of the stress tensor in granular media. *Int J Solids Struct* **50**, 2508-2517, doi:10.1016/j.ijsolstr.2013.04.001 (2013).

2 Otsuki, M. & Hayakawa, H. Discontinuous change of shear modulus for frictional jammed granular materials. *Phys Rev E* **95**, 062902, doi:10.1103/PhysRevE.95.062902 (2017).

3 Love, A. E. H. *A Treatise on the Mathematical Theory of Elasticity*. (Cambridge University Press, 1927).

4 Weber, J. Recherches concernant les contraintes intergranulaires dans les milieux pulvérulents. *Bull. Liaison P. et Ch.* **20**, 1-20 (1966).

5 Luding, S. in *The Physics of Granular Media* (eds H. Hinrichsen & D. E. Wolf) (Wiley-VCH, 2005).

6 Luding, S., Lätzel, M., Volk, W., Diebels, S. & Herrmann, H. J. From discrete element simulations to a continuum model. *Computer Methods in Applied Mechanics and Engineering* **191**, 21-28, doi:https://doi.org/10.1016/S0045-7825(01)00242-0 (2001).

7 Fortin, J., Millet, O. & de Saxce, G. Construction of an averaged stress tensor for a granular medium. *Eur J Mech a-Solid* **22**, 567-582, doi:10.1016/S0997-7538(03)00054-8 (2003).