Pizza3 - WORKSHOP - PostTreamement - Part 2bis

INRAE\Han Chen, Olivier Vitrac - rev. 2023-09-10

Table of Contents

1. Synopsis

This extension of the Part2 discusses advanced shear-stress details beyond those presented in Part2. It uses in particular the last features introduced recently: local vrial stress tensor calculated along with Landshoff and Hertz forces and Cauchy stresses reconstructed from projected forces on a Cartesian grid.

The main purpose of this document is to encourage a common understanding how the Landshoff forces can be equilibrated with Hertz contact forces and lead to a steady state. The physical origin and the mathematical formalism of theses forces are very different in nature and emerge from large simulations due to the transfer of momentum mediated between all these forces.

The main important equation: how Hertz contacts can develop a force which give sense of friction at wall and equates the shear stress in the flow regardless of the rigidity of Hertz contacts? There is a possible direct validation by noting that the The viscous tensor component is associated to the flow (x) can be defined independently from forces:

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y}\right)$$

In MD-like simulations, a similar stress can be derived from the virial stress tensor as:

$$\sigma_{\alpha\beta} = \frac{1}{V}\sum_{i<j} r_{ij,\alpha} f_{ij,\beta}$$

where V is the volume, $r_{ij,\alpha}$ is the α-component of the distance vector between particles i, j , and $f_{ij,\beta}$ is the β-component of the force between particles i and j.

For a validation based on `example2` results, the most relevant components are those relating to shear and normal forces, specifically σ_{xy} and σ_{yy}.

- σ_{xy} corresponds to the shear effects and should be closely related to the SPH-based shear stress τ_{xy} in the fluid. This component would be a primary point of comparison.
- σ_{yy} captures the effects of the Hertzian contacts along the y-direction (i.e., the direction opposite to which the wall is moving). In the Hertz contact model, the "macroscopic" normal force is acting along the y-direction, which contributes to this component.

Therefore, based on the equality of mechanical states between the fluid and the wall, we should expect:

$$\sigma_{xy}^{\text{SPH}} = \sigma_{xy}^{\text{Hertz}}(\text{shear})$$
$$\sigma_{yy}^{\text{SPH}} = \sigma_{yy}^{\text{Hertz}}(\text{normal})$$

which reads (first index: direction, second index: vcomponent)

$$\sigma_{21}^{\text{SPH}} = \sigma_{21}^{\text{Hertz}}$$
$$\sigma_{22}^{\text{SPH}} = \sigma_{22}^{\text{Hertz}}$$

2. Theory and numerical implementation

2.1 Landshoff forces and their virial approximation: `forceLandshoff`

2.1.1. Function Description

The function `forceLandshoff` computes Landshoff forces, also known as viscous pressures, between fluid atoms in Smoothed-Particle Hydrodynamics (SPH) simulations. The function can also calculate the virial stress tensor and associated normal vectors. It is designed to be computationally efficient by leveraging vectorization.

2.1.2. Inputs

- `X`: $n_X \times 3$ matrix containing the coordinates of the atoms.
- `vX`: $n_X \times 3$ matrix containing the velocity components of the atoms.
- `V`: $n_X \times 1$ cell array representing the Verlet list for each atom, detailing its neighboring atoms.
- `config`: A structure containing various parameters like the gradient kernel function, smoothing length h, speed of sound c_0, coefficient q_1, density ρ, and mass of the bead m.

2.1.3. Outputs

Depending on the number of requested outputs, the function can return:

- `F`: $n_X \times 3$ vector of Landshoff forces for each atom.
- `W`: $n_X \times 9$ matrix of virial stress tensors.
- `n`: $n_X \times 3$ matrix of normal vectors.

2.1.4. Equation

The Landshoff force F_{ij} between two fluid particles i and j is calculated as follows:

$$F_{ij} = \alpha \frac{m}{\rho} \left(\frac{v_{ij} \cdot r_{ij}}{r_{ij}^2 + \epsilon h^2} \right) \nabla W(r_{ij})$$

where

- $\alpha = q_1 \times c_0 \times h$

- m is the mass of the bead

- ρ is the fluid density

- v_{ij} is the relative velocity between particles i and j

- r_{ij} is the position vector between particles i and j

- $\nabla W(r_{ij})$ is the gradient of the smoothing kernel (1D gradient)

- ϵh^2 is a small number to prevent division by zero

- h is the smoothing length

2.1.5. Numerical Implementation

The primary loop in the function iterates over each atom (primary atom) and uses its corresponding Verlet list to identify the neighbors. The core of the algorithm uses this list to calculate the **Landshoff force** using the following variables and equations:

1. *Relative Position* $\vec{r}_{ij}$: Vector from neighbor atom j to the primary atom i.
2. *Relative Velocity* $\vec{v}_{ij}$: Velocity of atom i relative to j.
3. *Projected Velocity* $r_{v_{ij}}$: The dot product of $\vec{r}_{ij}$ and $\vec{v}_{ij}$.

The Landshoff force $\vec{F}_{ij}$ is calculated using:

$$\vec{F}_{ij} = \text{prefactor} \times \frac{r_{v_{ij}}}{r_{ij}^2 + \epsilon h^2} \times \text{gradkernel}(r_{ij}) \times \hat{r}_{ij}$$

```
j = V{i}; % neighbors of i
rij = X(i,:) - X(j,:);    % position vector j->i
rij2 = dot(rij,rij,2);    % dot(rij,rij,2)
rij_d = sqrt(rij2);       % norm
rij_n = rij ./ rij_d;     % normalized vector
vij = vX(i,:) - vX(j,:);  % relative velocity of i respectively to j
rvij = dot( rij, vij, 2); % projected velocity
ok = rvij<0;
if any(ok)
    muij = rvij(ok) ./ ( rij2(ok) + epsh );
    Fij = prefactor * muij .* config.gradkernel(rij_d(ok)) .* rij_n(ok,:);
    Fbalance = sum(Fij,1);
    F(i) = norm(Fbalance);
    n(i,:) = Fbalance/F(i);
end
```

Here, `gradkernel` is the gradient of the kernel function, $\hat{r}_{ij}$ is the unit vector along $\vec{r}_{ij}$, and `prefactor` is a constant calculated from given parameters.

2.1.6 Local Virial Stress Tensor approximation

The virial stress tensor $\mathbf{W}$ is also calculated when `askvirialstress` is set to true from the number of outputs.

1. *Single output (*`askvirialstress=false`*)*: $\vec{F}$, a vector containing the magnitudes of Landshoff forces for each atom.
2. *Multiple outputs (*`askvirialstress=true`*)*: $\vec{F}$, $\mathbf{W}$ (Virial stress tensor), $\vec{n}$ (normal vectors).

For each particle, the stress tensor is an outer product of the relative position vector and the Landshoff force vector:

$$\mathbf{StressTensor} = -\frac{1}{\mathrm{vol}}\left(\vec{r}_{ij}^{\,T} \times \vec{F}_{ij}\right)$$

Where vol is the volume of the summation spherical neighborhood, calculated using the smoothing length $h$and density ρ.

The stress tensor for each particle is then reshaped into a 1D vector and stored in $\mathbf{W}$.

```
% for virial stress calculation
volmin = config.m/config.rho;
vol = 4/3 * pi * config.h^3;
if askvirialstress
    W = zeros(nX,dX*dX,classX);
    if vol<volmin
        error('the value of h (%0.4g) leads to a volume smaller than atoms
(%0.4g)',config.h,volmin)
    end
else
    W = [];
end
stresstensor = zeros(dX,dX,classX);
% [...]
if askvirialstress
     stresstensor(:) = 0;
     for ineigh = 1:length(find(ok))
          % -rij' * Fij is the outerproduct (source: doi:10.1016/j.ijsolstr.2008.03.016)
          stresstensor = stresstensor - ( rij(ineigh,:)' * Fij(ineigh,:) )/vol
     end
     W(i,:) = stresstensor(:)';
end
```

Thus, the Verlet list is used to efficiently calculate forces only between neighboring particles, and the virial stress tensor is calculated as an outer product of the force and relative position vectors, normalized by the particle volume.

2.1.7. Comparison with Monaghan's Standard Formulation

In the standard formulation by Monaghan, viscous forces often have a more empirical form, sometimes based on artificial viscosity terms. The function `forceLandshoff` is somewhat more generalized, and its formulation can be tailored through various input parameters, including the smoothing length, the speed of sound, and other coefficients.

It is also capable of computing the virial stress tensor, allowing for a more in-depth understanding of the fluid's state, which may not always be straightforward in Monaghan's original formulation.

The mathematical notation is consistent with the typical conventions used in multiscale modeling and SPH simulations, making it well-suited for advanced applications, including those requiring a detailed understanding of fluid mechanics and mass transfer.

2.2 Hertz Contact: `ForceHertz`

The function `forceHertz` is designed to calculate Hertzian contact forces between particles based on their positions and velocities. It also outputs the virial stress tensor and normal vectors under specific conditions. This is particularly useful for setting up boundary conditions between fluid-solid and solid-wall domains in multi-scale modeling scenarios.

2.2.1. Hertz Contact Force Calculation

For each pair of interacting atoms i and j, a vector $\vec{r}_{ij}$ is calculated representing the relative position of j with respect to i. The Hertz contact force $\vec{F}_{ij}$ is computed using:

$$\delta = r_{\text{cut}} - ||\vec{r}_{ij}||$$

$$r_{\text{geom}} = \frac{R_1 R_2}{r_{\text{cut}}}$$

$$K = \frac{E}{3(1-2\nu)}$$

$$\vec{F}_{ij} = 1.066666667 \times K \times \delta \times \sqrt{\delta \times r_{\text{geom}}} \times \hat{r}_{ij}$$

```
%SMD source code of LAMMPS: lammps-2022-10-04/src/MACHDYN/pair_smd_hertz.cpp ln. 169
```

Where:

- r_{cut} is the cut-off distance.
- R_1, R_2 are the radii of particles i and j.
- E is the effective elastic modulus calculated using the Bertholet formula.
- ν is the Poisson's ratio, assumed to be 0.25 in the code.
- K is the effective bulk modulus.

```
% parameters
E = sqrt(config(1).E*config(2).E); % Bertholet formula
rcut = config(1).R + config(2).R;
% [ ... ]
rij = X(i,:) - X(j,:);    % position vector j->i
rij2 = dot(rij,rij,2);    % dot(rij,rij,2)
rij_d = sqrt(rij2);       % norm
rij_n = rij ./ rij_d;     % normalized vector
iscontact = rij_d < rcut;
stresstensor(:) = 0
if any(iscontact) % if they are contact
   %Fij = E * sqrt((rcut-rij_d(iscontact))*config(1).R*config(2).R/rcut) .* rij_n(iscontact,:);
   % formula as set in the SMD source code of LAMMPS: lammps-2022-10-04/src/MACHDYN/
pair_smd_hertz.cpp ln. 169
   delta = rcut-rij_d(iscontact);
   r_geom = config(1).R*config(2).R/rcut;
   bulkmodulus = E/(3*(1-2*0.25));
   Fij = 1.066666667 * bulkmodulus * delta .* sqrt(delta * r_geom).* rij_n(iscontact,:); %
units N
   Fbalance = sum(Fij,1);
   F(i) = norm(Fbalance);
   n(i,:) = Fbalance/F(i);
end
% Comment OV: The factor 1.066666667 appears to be an empirical correction,
```

2.2.2. Virial Stress Tensor

When the variable `askvirialstress` is true (more than one output), the function also computes the virial stress tensor W:

$$\mathbf{W}_i = -\frac{1}{\mathrm{vol}} \sum_j \vec{r}_{ij}^{\,T} \times \vec{F}_{ij}$$

Where vol is the volume of the summation spherical neighborhood, calculated using the smoothing length $h$and density ρ.

```
volmin = mean([config.m],'omitnan')/mean([config.rho],'omitnan');
hmean = mean([config.h],'omitnan');
vol = 4/3 * pi * hmean^3;
if vol<volmin
      error('the value of h (%0.4g) leads to a volume smaller than atoms (%0.4g)',hmean,volmin
end
% [ ... ]
if askvirialstress
     stresstensor(:) = 0;
     for ineigh = 1:length(find(iscontact))
           % -rij' * Fij is the outerproduct (source: doi:10.1016/j.ijsolstr.2008.03.016)
                stresstensor = stresstensor - ( rij(ineigh,:)' * Fij(ineigh,:) ) /vol
      end
      W(i,:) = stresstensor(:)';
end
```

2.2.3. Function `ForceHertz` Inputs and Outputs

Inputs

- X: The $n \times 3$ matrix of atomic coordinates.
- V: The $n \times 1$ cell array representing the Verlet list.
- $config$: A 1×2 structure containing (fields) parameters like radius R, elastic modulus E, density ρ, and mass m.

Outputs

- F: The $n \times 3$ matrix of Hertz forces for each atom.
- W: The $n \times 9$ matrix representing the virial stress tensor.
- n: The $n \times 3$ matrix of normal vectors.

By utilizing a Verlet list and computing both the Hertzian forces and virial stress tensors, this function can efficiently be used in multiscale simulations for complex fluid-solid interactions.

2.3. Cauchy stress tensor from grid-projected forces: `Interp3Cauchy`

The code `Interp3Cauchy` calculates the Cauchy stress tensor at each point on a 3D grid, onto which Landshoff and Hert contact forces are projected. It takes into account the force vectors and coordinates at each grid point to accomplish this.

2.3.1. Classical Definition of Cauchy Stress Tensor

In classical continuum mechanics, the Cauchy stress tensor σ at a point is given by:

$$\sigma_{ij} = \lim_{\Delta A \to 0} \frac{\Delta F_j}{\Delta A_i}$$

where ΔF_j is the force in the j direction acting on an infinitesimal area ΔA_i oriented along the i direction.

2.3.2. Calculation in the Code

The code calculates a `local_tensor`, which serves as the Cauchy stress tensor for a particular grid cell. The components of this tensor are given by:

$$\text{local\_tensor}(\alpha, \beta) = \frac{F_{\text{avg}}(\alpha)}{A(\beta)}$$

where $F_{\text{avg}}(\alpha)$ is the average force in the α direction (either $x, y,$ or z) acting on the grid cell, and $A(\beta)$ is the area of the face of the grid cell in the β direction.

2.3.3. Code Algorithm

1. The code checks for input errors and inconsistencies, making sure all dimensions match.
2. A 4D array named `stress` is initialized to store the local stress tensors.
3. Three nested loops traverse the 3D grid. At each point, the code:

- Initializes a `local_tensor` to zero.
- Calculates dx, dy, dz which represent the increments in each direction.
- Calculates A, the area for each face of the grid cell.
- Loops through each face (`beta`) of the grid cell and:
- - Calculates the average force (F_{avg}) for the vertices constituting the face.
- - Updates the `local_tensor` components according to $\frac{F_{\text{avg}}(\alpha)}{A(\beta)}$.

4. This local tensor is then stored in the 4D `stress` array.

```
for iy = 1:ny
    for ix = 1:nx
        for iz = 1:nz
            % Initialize local tensor
            local_tensor = zeros(3, 3);
            % Edge cases for dx, dy, dz
            dx = Xw(iy, min(ix+1,nx), iz) - Xw(iy, ix, iz);
            dy = Yw(min(iy+1,ny), ix, iz) - Yw(iy, ix, iz);
            dz = Zw(iy, ix, min(iz+1,nz)) - Zw(iy, ix, iz);
            % Calculate area of each face of this cell
            A = [dy * dz, dx * dz, dx * dy];  % Face areas
            % Calculate tensor components for each face considering only the four vertices of
the face
            for beta = 1:3 % -> direction beta=1 (x-face), beta=2 (y-face), beta=3 (z-face
                % Define the indices for the 4 vertices constituting each face
                if beta == 1 && iy < ny && iz < nz
                    vert_idx = repmat(ix, 1, 4);
                    vert_idy = [iy, iy, iy+1, iy+1];
                    vert_idz = [iz, iz+1, iz, iz+1];
                elseif beta == 2 && ix < nx && iz < nz
                    vert_idy = repmat(iy, 1, 4);
                    vert_idx = [ix, ix+1, ix, ix+1];
                    vert_idz = [iz, iz, iz+1, iz+1];
                elseif beta == 3 && ix < nx && iy < ny
                    vert_idz = repmat(iz, 1, 4);
                    vert_idx = [ix, ix+1, ix, ix+1];
                    vert_idy = [iy, iy, iy+1, iy+1];
                else
                    continue;  % Skip, as it's the edge of the grid
                end
                % Translate to 1D indices for force matrices
                ind = sub2ind([ny, nx, nz], vert_idy, vert_idx, vert_idz);
                for alpha = 1:3 % -> force component
                    % Compute average force on vertices
                    if alpha == 1
                        F_alpha_avg = mean(FXw(ind), 'omitnan');
                    elseif alpha == 2
                        F_alpha_avg = mean(FYw(ind), 'omitnan');
                    elseif alpha == 3
                        F_alpha_avg = mean(FZw(ind), 'omitnan');
                    end
                    if isnan(F_alpha_avg), continue; end
                    % Update the stress tensor component
                    local_tensor(beta, alpha) = F_alpha_avg / A(beta);
                end % next alpha
            end % next beta
            % [ ... ]
        end % next iz
    end % next ix
end % next iy
```

2.3.4. Alignment with Classical Definition

The formula for the components of `local_tensor` essentially represents the Cauchy stress tensor components $\sigma_{\alpha\beta}$ where α, β can be x, y, or z.

3. Workshop Initialization

The workshop requires one or two frames. All calculations have been designed to work with a 8-core laptop with 16 GB RAM or more.

3.1. File structure and input files

```
% File Structure (change your local path to reflect the content)
% ├─── example2bis.m (main developing file)
% ├─── notebook
% │     ├─── example2bis.mlx <-- this file to be run from here
% └─── ...
% └─── data folder (dumps/pub1/)
```

Please run this file from `notebook\`, add the parent folder, where dependency files are located, to Matlab path (use `pathtool` to check path). You do not need to save the path.

```
addpath(rootdir(pwd),'-begin')
datafolder = '../dumps/pub1/';
dumpfile =
'dump.ulsphBulk_hertzBoundary_referenceParameterExponent+1_with1SuspendedPart
icle';
```

3.2 Global definitions and check the dump file

```
%% Definitions
% We assume that the dump file has been preprocessed (see example1.m and
example2.m)
statvec = @(f,before,after) dispf('%s: %s%d values | average = %0.5g %s',
before, ...
    cell2mat(cellfun(@(x) sprintf(' %0.1f%%> %10.4g | ', x, prctile(f, x)),
{2.5, 25, 50, 75, 97.5}, 'UniformOutput', false)), ...
    length(f), mean(f),after); % user function to display statistics on
vectors (usage:  statvec(f,'myvar','ok'))
coords = {'x','y','z'};
vcoords = cellfun(@(c) ['v',c],coords,'UniformOutput',false); % vx, vy, vz
% dump file and its parameterization
datafolder = lamdumpread2(fullfile(datafolder,dumpfile),'search'); % fix
datafolder based on initial guess
```

```
Look for 'dump.ulsphBulk_hertzBoundary_referenceParameterExponent+1_with1SuspendedParticle'... (be patient
...found in '../dumps/pub1/numericalViscosimeter_reference_ulsphBulk_hertzBoundary'

The dumpfile 'dump.ulsphBulk_hertzBoundary_referenceParameterExponent+1_with1SuspendedParticle' has been f
    in the folder: ../dumps/pub1/numericalViscosimeter_reference_ulsphBulk_hertzBoundary
the original search started in ../dumps/pub1
The frame (split) folder is: ../dumps/pub1/numericalViscosimeter_reference_ulsphBulk_hertzBoundary/PREFETC
The first frame (split) is located in: ../dumps/pub1/numericalViscosimeter_reference_ulsphBulk_hertzBounda
The prefetch is split in several files.
1014 TIMESTEPS are availble:
  Column 01      Column 02      Column 03      Column 04      Column 05      Column 06      Column 07
        0       102000       204000       306000       408000       510000       612000       714000
     1000       103000       205000       307000       409000       511000       613000       715000
     2000       104000       206000       308000       410000       512000       614000       716000
     3000       105000       207000       309000       411000       513000       615000       717000
     4000       106000       208000       310000       412000       514000       616000       718000
     5000       107000       209000       311000       413000       515000       617000       719000
     6000       108000       210000       312000       414000       516000       618000       720000
     7000       109000       211000       313000       415000       517000       619000       721000
     8000       110000       212000       314000       416000       518000       620000       722000
     9000       111000       213000       315000       417000       519000       621000       723000
    10000       112000       214000       316000       418000       520000       622000       724000
    11000       113000       215000       317000       419000       521000       623000       725000
    12000       114000       216000       318000       420000       522000       624000       726000
    13000       115000       217000       319000       421000       523000       625000       727000
    14000       116000       218000       320000       422000       524000       626000       728000
    15000       117000       219000       321000       423000       525000       627000       729000
    16000       118000       220000       322000       424000       526000       628000       730000
    17000       119000       221000       323000       425000       527000       629000       731000
    18000       120000       222000       324000       426000       528000       630000       732000
    19000       121000       223000       325000       427000       529000       631000       733000
    20000       122000       224000       326000       428000       530000       632000       734000
    21000       123000       225000       327000       429000       531000       633000       735000
    22000       124000       226000       328000       430000       532000       634000       736000
    23000       125000       227000       329000       431000       533000       635000       737000
    24000       126000       228000       330000       432000       534000       636000       738000
    25000       127000       229000       331000       433000       535000       637000       739000
    26000       128000       230000       332000       434000       536000       638000       740000
    27000       129000       231000       333000       435000       537000       639000       741000
    28000       130000       232000       334000       436000       538000       640000       742000
    29000       131000       233000       335000       437000       539000       641000       743000
    30000       132000       234000       336000       438000       540000       642000       744000
    31000       133000       235000       337000       439000       541000       643000       745000
    32000       134000       236000       338000       440000       542000       644000       746000
    33000       135000       237000       339000       441000       543000       645000       747000
    34000       136000       238000       340000       442000       544000       646000       748000
    35000       137000       239000       341000       443000       545000       647000       749000
    36000       138000       240000       342000       444000       546000       648000       750000
    37000       139000       241000       343000       445000       547000       649000       751000
    38000       140000       242000       344000       446000       548000       650000       752000
    39000       141000       243000       345000       447000       549000       651000       753000
    40000       142000       244000       346000       448000       550000       652000       754000
    41000       143000       245000       347000       449000       551000       653000       755000
    42000       144000       246000       348000       450000       552000       654000       756000
    43000       145000       247000       349000       451000       553000       655000       757000
    44000       146000       248000       350000       452000       554000       656000       758000
    45000       147000       249000       351000       453000       555000       657000       759000
    46000       148000       250000       352000       454000       556000       658000       760000
    47000       149000       251000       353000       455000       557000       659000       761000
    48000       150000       252000       354000       456000       558000       660000       762000
    49000       151000       253000       355000       457000       559000       661000       763000
    50000       152000       254000       356000       458000       560000       662000       764000
    51000       153000       255000       357000       459000       561000       663000       765000
    52000       154000       256000       358000       460000       562000       664000       766000
    53000       155000       257000       359000       461000       563000       665000       767000
    54000       156000       258000       360000       462000       564000       666000       768000
    55000       157000       259000       361000       463000       565000       667000       769000
    56000       158000       260000       362000       464000       566000       668000       770000
    57000       159000       261000       363000       465000       567000       669000       771000
    58000       160000       262000       364000       466000       568000       670000       772000
    59000       161000       263000       365000       467000       569000       671000       773000
    60000       162000       264000       366000       468000       570000       672000       774000
    61000       163000       265000       367000       469000       571000       673000       775000
    62000       164000       266000       368000       470000       572000       674000       776000
    63000       165000       267000       369000       471000       573000       675000       777000
    64000       166000       268000       370000       472000       574000       676000       778000
    65000       167000       269000       371000       473000       575000       677000       779000
    66000       168000       270000       372000       474000       576000       678000       780000
    67000       169000       271000       373000       475000       577000       679000       781000
    68000       170000       272000       374000       476000       578000       680000       782000
    69000       171000       273000       375000       477000       579000       681000       783000
    70000       172000       274000       376000       478000       580000       682000       784000
    71000       173000       275000       377000       479000       581000       683000       785000
    72000       174000       276000       378000       480000       582000       684000       786000
    73000       175000       277000       379000       481000       583000       685000       787000
    74000       176000       278000       380000       482000       584000       686000       788000
    75000       177000       279000       381000       483000       585000       687000       789000
    76000       178000       280000       382000       484000       586000       688000       790000
    77000       179000       281000       383000       485000       587000       689000       791000
    78000       180000       282000       384000       486000       588000       690000       792000
    79000       181000       283000       385000       487000       589000       691000       793000
    80000       182000       284000       386000       488000       590000       692000       794000
    81000       183000       285000       387000       489000       591000       693000       795000
    82000       184000       286000       388000       490000       592000       694000       796000
    83000       185000       287000       389000       491000       593000       695000       797000
    84000       186000       288000       390000       492000       594000       696000       798000
    85000       187000       289000       391000       493000       595000       697000       799000
    86000       188000       290000       392000       494000       596000       698000       800000
    87000       189000       291000       393000       495000       597000       699000       801000
    88000       190000       292000       394000       496000       598000       700000       802000
    89000       191000       293000       395000       497000       599000       701000       803000
    90000       192000       294000       396000       498000       600000       702000       804000
    91000       193000       295000       397000       499000       601000       703000       805000
    92000       194000       296000       398000       500000       602000       704000       806000
    93000       195000       297000       399000       501000       603000       705000       807000
    94000       196000       298000       400000       502000       604000       706000       808000
    95000       197000       299000       401000       503000       605000       707000       809000
    96000       198000       300000       402000       504000       606000       708000       810000
    97000       199000       301000       403000       505000       607000       709000       811000
    98000       200000       302000       404000       506000       608000       710000       812000
    99000       201000       303000       405000       507000       609000       711000       813000
   100000       202000       304000       406000       508000       610000       712000       814000
   101000       203000       305000       407000       509000       611000       713000       815000
Choose the time step you are interested in.
Only the first one is returned for now.
Use the prefetch (split: TIMESTEP 0) folder (instead of '../dumps/pub1/numericalViscosimeter_reference_uls
    TIMESTEP_000000000.mat        27-Jul-2023 20:52:07        1.1 MBytes        ../dumps/pub1/numericalVis
...loaded in 0.816 s
```

```
X0 = lamdumpread2(fullfile(datafolder,dumpfile)); % default frame
```

```
The prefetch is split in several files.
1014 TIMESTEPS are availble:
  Column 01      Column 02      Column 03      Column 04      Column 05      Column 06      Column 07
        0       102000       204000       306000       408000       510000       612000       714000
     1000       103000       205000       307000       409000       511000       613000       715000
     2000       104000       206000       308000       410000       512000       614000       716000
     3000       105000       207000       309000       411000       513000       615000       717000
     4000       106000       208000       310000       412000       514000       616000       718000
     5000       107000       209000       311000       413000       515000       617000       719000
     6000       108000       210000       312000       414000       516000       618000       720000
     7000       109000       211000       313000       415000       517000       619000       721000
     8000       110000       212000       314000       416000       518000       620000       722000
     9000       111000       213000       315000       417000       519000       621000       723000
    10000       112000       214000       316000       418000       520000       622000       724000
```

11000	113000	215000	317000	419000	521000	623000	725000
12000	114000	216000	318000	420000	522000	624000	726000
13000	115000	217000	319000	421000	523000	625000	727000
14000	116000	218000	320000	422000	524000	626000	728000
15000	117000	219000	321000	423000	525000	627000	729000
16000	118000	220000	322000	424000	526000	628000	730000
17000	119000	221000	323000	425000	527000	629000	731000
18000	120000	222000	324000	426000	528000	630000	732000
19000	121000	223000	325000	427000	529000	631000	733000
20000	122000	224000	326000	428000	530000	632000	734000
21000	123000	225000	327000	429000	531000	633000	735000
22000	124000	226000	328000	430000	532000	634000	736000
23000	125000	227000	329000	431000	533000	635000	737000
24000	126000	228000	330000	432000	534000	636000	738000
25000	127000	229000	331000	433000	535000	637000	739000
26000	128000	230000	332000	434000	536000	638000	740000
27000	129000	231000	333000	435000	537000	639000	741000
28000	130000	232000	334000	436000	538000	640000	742000
29000	131000	233000	335000	437000	539000	641000	743000
30000	132000	234000	336000	438000	540000	642000	744000
31000	133000	235000	337000	439000	541000	643000	745000
32000	134000	236000	338000	440000	542000	644000	746000
33000	135000	237000	339000	441000	543000	645000	747000
34000	136000	238000	340000	442000	544000	646000	748000
35000	137000	239000	341000	443000	545000	647000	749000
36000	138000	240000	342000	444000	546000	648000	750000
37000	139000	241000	343000	445000	547000	649000	751000
38000	140000	242000	344000	446000	548000	650000	752000
39000	141000	243000	345000	447000	549000	651000	753000
40000	142000	244000	346000	448000	550000	652000	754000
41000	143000	245000	347000	449000	551000	653000	755000
42000	144000	246000	348000	450000	552000	654000	756000
43000	145000	247000	349000	451000	553000	655000	757000
44000	146000	248000	350000	452000	554000	656000	758000
45000	147000	249000	351000	453000	555000	657000	759000
46000	148000	250000	352000	454000	556000	658000	760000
47000	149000	251000	353000	455000	557000	659000	761000
48000	150000	252000	354000	456000	558000	660000	762000
49000	151000	253000	355000	457000	559000	661000	763000
50000	152000	254000	356000	458000	560000	662000	764000
51000	153000	255000	357000	459000	561000	663000	765000
52000	154000	256000	358000	460000	562000	664000	766000
53000	155000	257000	359000	461000	563000	665000	767000
54000	156000	258000	360000	462000	564000	666000	768000
55000	157000	259000	361000	463000	565000	667000	769000
56000	158000	260000	362000	464000	566000	668000	770000
57000	159000	261000	363000	465000	567000	669000	771000
58000	160000	262000	364000	466000	568000	670000	772000
59000	161000	263000	365000	467000	569000	671000	773000
60000	162000	264000	366000	468000	570000	672000	774000
61000	163000	265000	367000	469000	571000	673000	775000
62000	164000	266000	368000	470000	572000	674000	776000
63000	165000	267000	369000	471000	573000	675000	777000
64000	166000	268000	370000	472000	574000	676000	778000
65000	167000	269000	371000	473000	575000	677000	779000
66000	168000	270000	372000	474000	576000	678000	780000
67000	169000	271000	373000	475000	577000	679000	781000
68000	170000	272000	374000	476000	578000	680000	782000
69000	171000	273000	375000	477000	579000	681000	783000
70000	172000	274000	376000	478000	580000	682000	784000
71000	173000	275000	377000	479000	581000	683000	785000
72000	174000	276000	378000	480000	582000	684000	786000
73000	175000	277000	379000	481000	583000	685000	787000
74000	176000	278000	380000	482000	584000	686000	788000

```
    75000       177000       279000       381000       483000       585000       687000       789000
    76000       178000       280000       382000       484000       586000       688000       790000
    77000       179000       281000       383000       485000       587000       689000       791000
    78000       180000       282000       384000       486000       588000       690000       792000
    79000       181000       283000       385000       487000       589000       691000       793000
    80000       182000       284000       386000       488000       590000       692000       794000
    81000       183000       285000       387000       489000       591000       693000       795000
    82000       184000       286000       388000       490000       592000       694000       796000
    83000       185000       287000       389000       491000       593000       695000       797000
    84000       186000       288000       390000       492000       594000       696000       798000
    85000       187000       289000       391000       493000       595000       697000       799000
    86000       188000       290000       392000       494000       596000       698000       800000
    87000       189000       291000       393000       495000       597000       699000       801000
    88000       190000       292000       394000       496000       598000       700000       802000
    89000       191000       293000       395000       497000       599000       701000       803000
    90000       192000       294000       396000       498000       600000       702000       804000
    91000       193000       295000       397000       499000       601000       703000       805000
    92000       194000       296000       398000       500000       602000       704000       806000
    93000       195000       297000       399000       501000       603000       705000       807000
    94000       196000       298000       400000       502000       604000       706000       808000
    95000       197000       299000       401000       503000       605000       707000       809000
    96000       198000       300000       402000       504000       606000       708000       810000
    97000       199000       301000       403000       505000       607000       709000       811000
    98000       200000       302000       404000       506000       608000       710000       812000
    99000       201000       303000       405000       507000       609000       711000       813000
   100000       202000       304000       406000       508000       610000       712000       814000
   101000       203000       305000       407000       509000       611000       713000       815000
Choose the time step you are interested in.
Only the first one is returned for now.
Use the prefetch (split: TIMESTEP 0) folder (instead of '../dumps/pub1/numericalViscosimeter_reference_uls
    TIMESTEP_000000000.mat        27-Jul-2023 20:52:07        1.1 MBytes        ../dumps/pub1/numericalVis
...loaded in 0.413 s
```

```matlab
boxdims = X0.BOX(:,2) - X0.BOX(:,1);                % box dims
natoms = X0.NUMBER;                                 % number of atoms
timesteps = X0.TIMESTEPS;                           % time steps
ntimesteps = length(timesteps);                     % number of time steps
T = X0.ATOMS.type;                                  % atom types
% === IDENTIFICATION OF ATOMS ===
atomtypes = unique(T);                              % list of atom types
natomspertype = arrayfun(@(t) length(find(T==t)),atomtypes);
[~,fluidtype] = max(natomspertype); % fluid type
[~,solidtype] = min(natomspertype); % solid type
walltypes = setdiff(atomtypes,[fluidtype,solidtype]); % wall types
nfluidatoms = natomspertype(fluidtype);
nsolidatoms = natomspertype(solidtype);
% === FLOW DIRECTION ===
[~,iflow] = max(boxdims);
iothers = setdiff(1:size(X0.BOX,1),iflow);
% === GUESS BEAD SIZE ===
Vbead_guess = prod(boxdims)/natoms;
rbead_guess = (3/(4*pi)*Vbead_guess)^(1/3);
cutoff = 3*rbead_guess;
[verletList,cutoff,dmin,config,dist] =
buildVerletList(X0.ATOMS(T==fluidtype,coords),cutoff);
```

```
Build Verlet list by searching in blocks...
... done in 0.367 s with 729 search blocks | minimum distance 2.083e-05
     Sort the Verlet list...
     ... done in 0.503 s
buildVerletList: all done in 23.8 s for 215025 atoms
```

```
rbead = dmin/2; % based on separation distance
```

3.3. Frame for stress analysis

```
%% Frame and Corresponding ROI for Stress Analysis
list_timestepforstess = unique(timesteps(ceil((0.1:0.1:0.9)*ntimesteps)));
timestepforstress = list_timestepforstess(end);
% stress frame
Xstress = lamdumpread2(fullfile(datafolder,dumpfile),'usesplit',
[],timestepforstress); % middle frame
```

```
Use the prefetch (split: TIMESTEP 912000) folder (instead of '../dumps/pub1/numericalViscosimeter_referenc
    TIMESTEP_000912000.mat        27-Jul-2023 21:35:16        8.8 MBytes        ../dumps/pub1/numericalVis
...loaded in 0.353 s
```

```
Xstress.ATOMS.isfluid = Xstress.ATOMS.type==fluidtype;
Xstress.ATOMS.issolid = Xstress.ATOMS.type==solidtype;
Xstress.ATOMS.iswall  = ismember(Xstress.ATOMS.type,walltypes);
% == control ==
% average bead volume and bead radius (control, min separation distance was
used in the previous section)
fluidbox =
[min(Xstress.ATOMS{Xstress.ATOMS.isfluid,coords});max(Xstress.ATOMS{Xstress.A
TOMS.isfluid,coords})]';
vbead_est = prod(diff(fluidbox,1,2))/(length(find(Xstress.ATOMS.isfluid))
+length(Xstress.ATOMS.issolid));
rbead_est = (3*vbead_est/(4*pi))^(1/3);
mbead_est = vbead_est*1000;
% Verlet List Construction with Short Cutoff
% Builds a Verlet list with a short cutoff distance, designed to identify
only the closest neighbors.
[verletList,cutoff,dmin,config,dist] =
buildVerletList(Xstress.ATOMS,3*rbead);
```

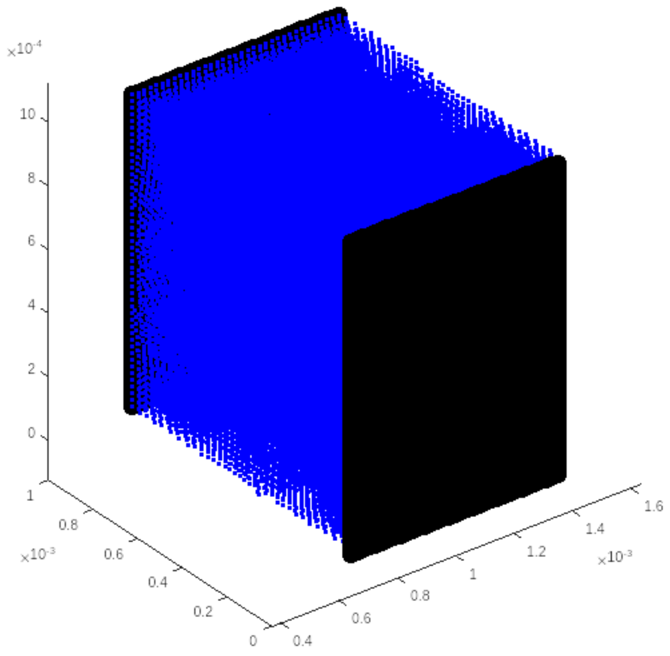
```
Build Verlet list by searching in blocks...
... done in 0.2145 s with 1584 search blocks | minimum distance 1.461e-05
     Sort the Verlet list...
     ... done in 0.676 s
buildVerletList: all done in 26.9 s for 313344 atoms
```

```
% Partition Verlet List Based on Atom Types
% This Verlet list is partitioned based on atom types, distinguishing
between interactions
% that are exclusively fluid-fluid, solid-fluid, or solid-solid.
verletListCross = partitionVerletList(verletList,Xstress.ATOMS);
% Identify Contacting Atoms
```

```
Xstress.ATOMS.isincontact = ~cellfun(@isempty,verletListCross);
Xstress.ATOMS.contacttypes = cellfun(@(v)
Xstress.ATOMS.type(v)',verletListCross,'UniformOutput',false);
% Identify Atoms in Contact with Solids and Fluids
Xstress.ATOMS.isincontactwithsolid = cellfun(@(c) ismember(solidtype,c),
Xstress.ATOMS.contacttypes);
Xstress.ATOMS.isincontactwithfluid = cellfun(@(c) ismember(fluidtype,c),
Xstress.ATOMS.contacttypes);
Xstress.ATOMS.isincontactwithwalls = cellfun(@(c)
~isempty(intersect(walltypes,c)), Xstress.ATOMS.contacttypes);
% Flag Fluid Atoms in Contact with Solid and Vice Versa
Xstress.ATOMS.fluidincontactwithsolid = Xstress.ATOMS.isfluid &
Xstress.ATOMS.isincontactwithsolid;
Xstress.ATOMS.solidincontactwithfluid = Xstress.ATOMS.issolid &
Xstress.ATOMS.isincontactwithfluid;
Xstress.ATOMS.fluidincontactwithwalls = Xstress.ATOMS.isfluid &
Xstress.ATOMS.isincontactwithwalls;
Xstress.ATOMS.wallsincontactwithfluid = Xstress.ATOMS.iswall &
Xstress.ATOMS.isincontactwithfluid;
% Identify Indices for Analysis
ROI = [
    min(Xstress.ATOMS{Xstress.ATOMS.fluidincontactwithsolid,coords})
    max(Xstress.ATOMS{Xstress.ATOMS.fluidincontactwithsolid,coords})
    ]';
ROI(iflow,:) = mean(ROI(iflow,:)) + [-1 1] * diff(ROI(iflow,:));
for j = iothers
    ROI(j,:) = Xstress.BOX(j,:) ;
end
inROI = true(size(Xstress,1),1);
for c=1:length(coords)
    inROI = inROI & (Xstress.ATOMS{:,coords{c}}>=ROI(c,1)) &
(Xstress.ATOMS{:,coords{c}}<=ROI(c,2));
end
ifluid = find(inROI & (Xstress.ATOMS.isfluid));
isolid = find(inROI & (Xstress.ATOMS.issolid));
iwall = find(inROI & (Xstress.ATOMS.iswall));
isolidcontact = find(Xstress.ATOMS.solidincontactwithfluid);
iwallcontact = find(inROI & Xstress.ATOMS.wallsincontactwithfluid);
% control
figure, hold on
plot3D(Xstress.ATOMS{ifluid,coords},'bo','markersize',3,'markerfacecolor','b'
)
plot3D(Xstress.ATOMS{iwallcontact,coords},'ko','markersize',12,'markerfacecol
or','k')
plot3D(Xstress.ATOMS{isolidcontact,coords},'ro','markersize',16,'markerfaceco
lor','r')
axis equal, view(3), drawnow

% triangulation of the solid
```

```
DT = delaunayTriangulation(double(Xstress.ATOMS{isolidcontact,
{'x','y','z'}}));
K = convexHull(DT);
plotsolid = @()trisurf(K, DT.Points(:,1), DT.Points(:,2), DT.Points(:,3),
'FaceColor', 'w','Edgecolor','k','FaceAlpha',0.6);
plotsolid()
```



4. Landshoff forces and stresses in the fluid

4.1. Computation part

```
%% Landshoff forces and stresses in the fluid
% these forces are the viscous forces in the fluid
% we calculate:
%   Flandshoff(i,:) the local Landshoff force (1x3 vector) for the atom i
%   Wlandshoff(i,:) is the local virial stress tensor (3x3 matrix stored as
1x9 vector with Matlab conventions)
%       row-wise: force component index
%       columm-wise: coord component index
% === Build the Verlet list consistently with the local Virial Stress Tensor
===
% Changing h can affect the value of viscosity for Landshoff foces and shear
stress.
```

```
% While keeping the same hLandshoff, the results can be rescaled to the
value of h
% applied in the simulations.
Xfluid  = Xstress.ATOMS(ifluid,:);
hLandshoff = 4*rbead; %1.25e-5; % m
Vfluid = buildVerletList(Xfluid,hLandshoff);
```

```
Build Verlet list by searching in blocks...
... done in 0.05878 s with 80 search blocks | minimum distance 1.461e-05
     Sort the Verlet list...
     ... done in 0.282 s
buildVerletList: all done in 8.27 s for 77394 atoms
```

```
configLandshoff = struct( ...
     'gradkernel', kernelSPH(hLandshoff,'lucyder',3),...kernel gradient
     'h', hLandshoff,...smoothing length (m)
     'c0',0.32,...speed of the sound (m/s)
     'q1',30,... viscosity coefficient (-)
     'rho', 1000, ...density
     'm', 9.04e-12 ...
     );
mu = configLandshoff.q1*configLandshoff.c0*configLandshoff.h/10; % viscosity
estimate
dispf('Atificial viscosity: %0.4g Pa.s',mu)
```

```
Atificial viscosity: 3.999e-05 Pa.s
```

```


% Landshoff forces and local virial stress
[Flandshoff,Wlandshoff] = forceLandshoff(Xfluid,[],Vfluid,configLandshoff);
```

```
Calculate Landshoff forces + virial stress between [77394 x 3] atoms...
     ... done with virial stress in 1.88 s
```

```
flandshoff = sqrt(sum(Flandshoff.^2,2));
statvec(flandshoff,' Force Landshoff',sprintf('<-- TIMESTEP:
%d',timestepforstress))
```

```
 Force Landshoff:  2.5%>     0.3038 |  25.0%>      1.204 |  50.0%>      2.663 |  75.0%>      6.623 |  97.5
```

```
statvec(Wlandshoff(:,2),'Virial Landshoff',sprintf('<-- TIMESTEP:
%d',timestepforstress))
```

```
Virial Landshoff:  2.5%> -9.653e+08 |  25.0%> -2.323e+08 |  50.0%>  4.029e+06 |  75.0%>  2.423e+08 |  97.5
```

```


% number of grid points along the largest dimension
fluidbox = [ min(Xfluid{:,coords}); max(Xfluid{:,coords}) ]';
boxcenter = mean(fluidbox,2);
resolution = ceil(50 * diff(fluidbox,[],2)'./max(diff(fluidbox,[],2)));
xw = linspace(fluidbox(1,1),fluidbox(1,2),resolution(1));
yw = linspace(fluidbox(2,1),fluidbox(2,2),resolution(2));
zw = linspace(fluidbox(3,1),fluidbox(3,2),resolution(3));
[Xw,Yw,Zw] = meshgrid(xw,yw,zw);
```

```
XYZgrid = [Xw(:),Yw(:),Zw(:)];
hLandshoff = 5*rbead; %1.25e-5; % m
mbead = 9.04e-12; % repported by Billy
Vbead = mbead_est/1000; % use estimated value instead (less dicrepancy)

% Build the Grid Verlet list
VXYZ = buildVerletList({XYZgrid Xfluid{:,coords}},1.001*hLandshoff); %
special grid syntax
```

```
Build Verlet list by searching in blocks...
... done in 0.2235 s with 80 search blocks | minimum distance 9.08e-08
     Sort the Verlet list...
     ... done in 0.505 s
buildVerletList: all done in 14.9 s for 168044 grid points
```

```
% Interpolate Landshoff forces, extract components for plotting
W = kernelSPH(hLandshoff,'lucy',3); % kernel for interpolation (not
gradkernel!)
FXYZgrid =
interp3SPHVerlet(Xfluid{:,coords},Flandshoff,XYZgrid,VXYZ,W,Vbead);
```

```
INTERP3SPHVERLET interpolates 90650 x 3 grid points with a Verlet list including from 0 to 75 neighbors...
...done in 9.95 s. INTERP3SPHVerlet completed the interpolation of 90650 points with 77394 kernels
```

```
FXYZgridx = reshape(FXYZgrid(:,1),size(Xw));
FXYZgridy = reshape(FXYZgrid(:,2),size(Yw));
FXYZgridz = reshape(FXYZgrid(:,3),size(Zw));
% Interpolate local virial stress tensor, extract s12 which is stored as
s(2,1)
WXYZgrid =
interp3SPHVerlet(Xfluid{:,coords},Wlandshoff,XYZgrid,VXYZ,W,Vbead);
```

```
INTERP3SPHVERLET interpolates 90650 x 3 grid points with a Verlet list including from 0 to 75 neighbors...
...done in 25.19 s. INTERP3SPHVerlet completed the interpolation of 90650 points with 77394 kernels
```

```
s12grid = reshape(WXYZgrid(:,2),size(Xw)); % extract \sigma_{xy} i.e. forces
along x across y
% Alternative estimation of the virial from the Cauchy stress tensor
WXYZgrid2 = interp3cauchy(Xw,Yw,Zw,FXYZgridx,FXYZgridy,FXYZgridz);
```

```
Calculate the local Cauchy stress from a [49 x 37 x 50] grid...
     ... done in 3.4 s
```

```
s12grid2 = reshape(WXYZgrid2(:,:,:,2),size(Xw));
% Interpolate the velocities, extract components for plotting
vXYZgrid =
interp3SPHVerlet(Xfluid{:,coords},Xfluid{:,vcoords},XYZgrid,VXYZ,W,Vbead);
```

```
INTERP3SPHVERLET interpolates 90650 x 3 grid points with a Verlet list including from 0 to 75 neighbors...
...done in 9.893 s. INTERP3SPHVerlet completed the interpolation of 90650 points with 77394 kernels
```

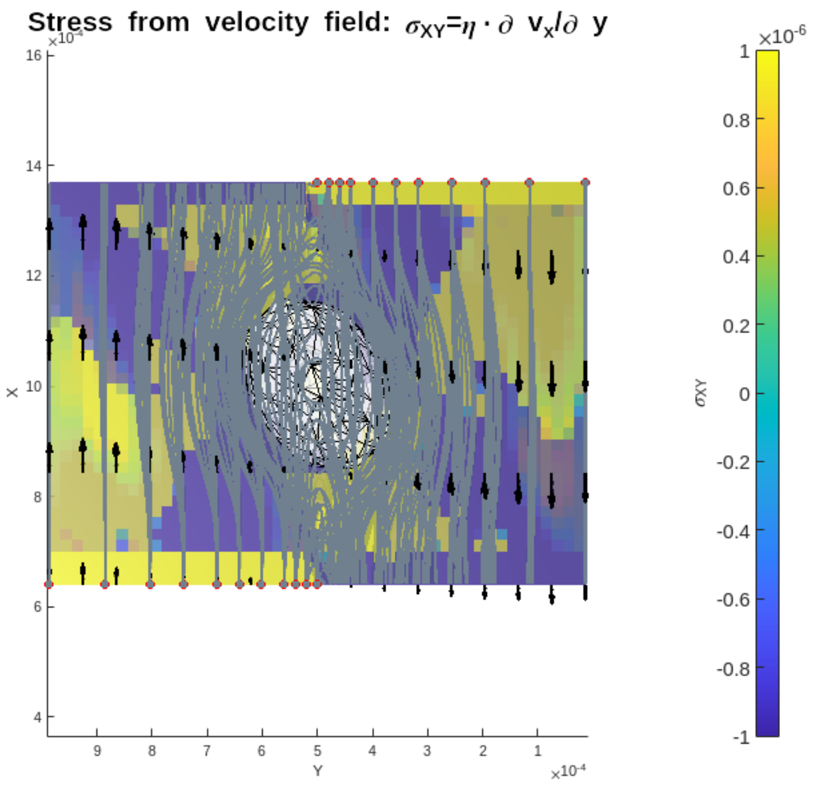
```
vXYZgridx = reshape(vXYZgrid(:,1),size(Xw));
vXYZgridy = reshape(vXYZgrid(:,2),size(Yw));
vXYZgridz = reshape(vXYZgrid(:,3),size(Zw));
vdXYZgridxdy = gradient(vXYZgridx,xw(2)-xw(1),yw(2)-yw(1),zw(2)-zw(1));
```

```
s12grid_est = mu * vdXYZgridxdy;
```

4.2 Plot the "reference" shear stress

The reference shear stress $\tau_{xy} = \mu\left(\frac{\partial u}{\partial y}\right)$ is estimated from vdXYZgridxdy and μ (mu).

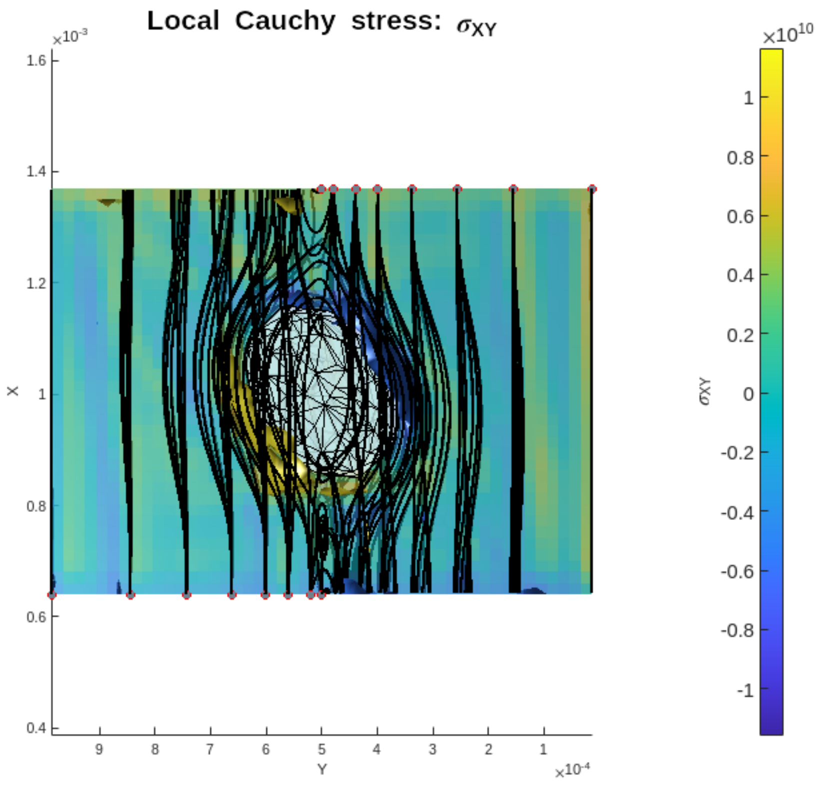
```
% === Plot Estimated shear stress (it should be the theoretical value)
figure, hold on
hs = slice(Xw,Yw,Zw,s12grid_est,[boxcenter(1) xw(end)],...
    [boxcenter(2) yw(end)],...
    [fluidbox(3,1) boxcenter(3)]);
set(hs,'edgecolor','none','facealpha',0.6)
plotsolid()
lighting gouraud, camlight('left'), axis equal, view(3) % shading interp
hc = colorbar('AxisLocation','in','fontsize',14); hc.Label.String =
'\sigma_{XY}';
title('Stress from velocity field: \sigma_{XY}=\eta\cdot\partial v_x/
\partial y','fontsize',20)
xlabel('X'), ylabel('Y'), zlabel('Z')
step = [3 10 5];
quiver3( ...
    Xw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Yw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Zw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    vXYZgridx(1:step(1):end,1:step(2):end,1:step(3):end), ...
    vXYZgridy(1:step(1):end,1:step(2):end,1:step(3):end), ...
    vXYZgridz(1:step(1):end,1:step(2):end,1:step(3):end) ...
    ,1,'color','k','LineWidth',2)
npart = 30;
[startX,startY,startZ] = meshgrid( ...
    double(xw(1)), ...
    double(yw(unique(round(1+(1+
(linspace(-1,1,npart).*linspace(-1,1,npart).^2))*floor(length(yw)/2))))),...
    double(yw(unique(round(1+(1+
(linspace(-1,1,npart).*linspace(-1,1,npart).^2))*floor(length(yw)/2))))) ...
  );
vstart = interp3(Xw,Yw,Zw,vXYZgridx,startX,startY,startZ); startX(vstart<0)
= double(xw(end));
hsl =
streamline(double(Xw),double(Yw),double(Zw),vXYZgridx,vXYZgridy,vXYZgridz,sta
rtX,startY,startZ);
set(hsl,'linewidth',2,'color',[0.4375    0.5000    0.5625])
plot3(startX(:),startY(:),startZ(:),'ro','markerfacecolor',[0.4375
0.5000    0.5625])
% fix the view
view(-90,90), clim([-1e-6 1e-6])
```



The figure is plotting the shear-stress, the velocity field and stream lines. The top view is showing the local turbulence and heterogenieity in shear. The pattern at the entrance and outlet of the channel are numerical errors on the gradient.

4.3. Plot the shear stress from Cauchy tensor

```
% === Plot Estimated shear stress from Cauchy Tensor
figure, hold on
s1299 = prctile(abs(s12grid2(~isnan(s12grid2))),99);
isosurface(Xw,Yw,Zw,s12grid2,s1299)
isosurface(Xw,Yw,Zw,s12grid2,-s1299)
hs = slice(Xw,Yw,Zw,s12grid2,[boxcenter(1) xw(end)],...
    [boxcenter(2) yw(end)],...
    [fluidbox(3,1) boxcenter(3)]);
set(hs,'edgecolor','none','facealpha',0.6)
plotsolid()
lighting gouraud, camlight('left'), axis equal, view(3) % shading interp
hc = colorbar('AxisLocation','in','fontsize',14); hc.Label.String =
'\sigma_{XY}';
title('Local Cauchy stress: \sigma_{XY}','fontsize',20)
xlabel('X'), ylabel('Y'), zlabel('Z')
npart = 20;
[startX,startY,startZ] = meshgrid( ...
    double(xw(1)), ...
    double(yw(unique(round(1+(1+
(linspace(-1,1,npart).*linspace(-1,1,npart).^2))*floor(length(yw)/2))))),...
    double(yw(unique(round(1+(1+
(linspace(-1,1,npart).*linspace(-1,1,npart).^2))*floor(length(yw)/2))))) ...
  );
vstart = interp3(Xw,Yw,Zw,vXYZgridx,startX,startY,startZ); startX(vstart<0)
= double(xw(end));
hsl =
streamline(double(Xw),double(Yw),double(Zw),vXYZgridx,vXYZgridy,vXYZgridz,sta
rtX,startY,startZ);
set(hsl,'linewidth',2,'color','k')
plot3(startX(:),startY(:),startZ(:),'ro','markerfacecolor',[0.4375
0.5000    0.5625])
view(-90,90)
```



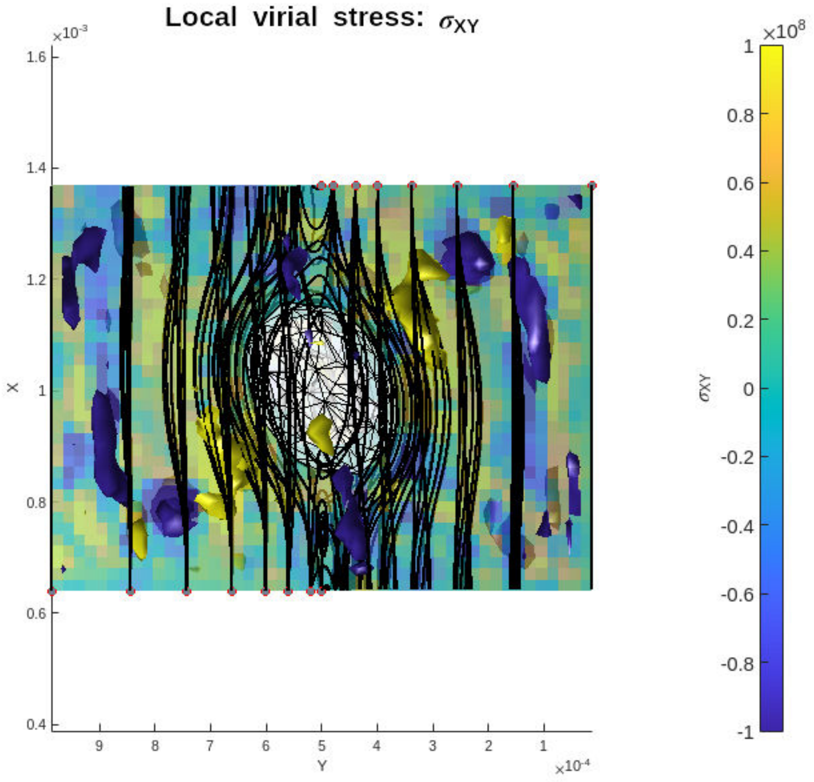
The top view shows a symmetric structure with two regions (isosurfaces corresponding to 1^{st} (blue) and 99^{th} (yellow)) with very high shear. They are reponsible of the deformation and the alignment of the particle. The solid particle presents an angle with the main axes of the channel. This angle reduces the cross-section of passage for the flow.

For a further discusssion on orientation effects, read: Computers & Mathematics with Applications 2020, 79(3) 539-554.

```
==========================================================================
    The scale of Landshoff stress does not match expected shear stress.
     It should be investigated at the first place.
==========================================================================
```

4.4. Plot the local virial stress

```
% === Plot local virial stess: s12 (s12 is stored as s(2,1))
figure, hold on
s1299 = prctile(abs(s12grid(~isnan(s12grid))),99);
isosurface(Xw,Yw,Zw,s12grid,s1299)
isosurface(Xw,Yw,Zw,s12grid,-s1299)
hs = slice(Xw,Yw,Zw,s12grid,[boxcenter(1) xw(end)],...
    [boxcenter(2) yw(end)],...
    [fluidbox(3,1) boxcenter(3)]);
set(hs,'edgecolor','none','facealpha',0.6)
plotsolid()
lighting gouraud, camlight('left'), axis equal, view(3) % shading interp
hc = colorbar('AxisLocation','in','fontsize',14); hc.Label.String =
'\sigma_{XY}';
title('Local virial stress: \sigma_{XY}','fontsize',20)
xlabel('X'), ylabel('Y'), zlabel('Z')
npart = 20;
[startX,startY,startZ] = meshgrid( ...
    double(xw(1)), ...
    double(yw(unique(round(1+(1+
(linspace(-1,1,npart).*linspace(-1,1,npart).^2))*floor(length(yw)/2))))),...
    double(yw(unique(round(1+(1+
(linspace(-1,1,npart).*linspace(-1,1,npart).^2))*floor(length(yw)/2))))) ...
  );
vstart = interp3(Xw,Yw,Zw,vXYZgridx,startX,startY,startZ); startX(vstart<0)
= double(xw(end));
hsl =
streamline(double(Xw),double(Yw),double(Zw),vXYZgridx,vXYZgridy,vXYZgridz,sta
rtX,startY,startZ);
set(hsl,'linewidth',2,'color','k')
plot3(startX(:),startY(:),startZ(:),'ro','markerfacecolor',[0.4375
0.5000    0.5625])
view(-90,90)
clim([-1 1]*1e8)
```

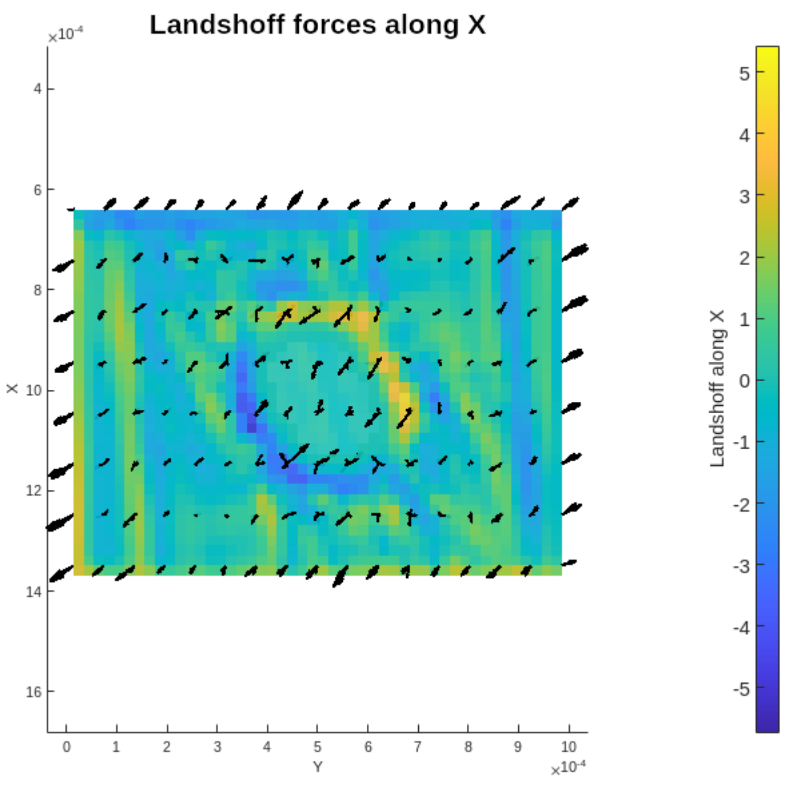


The scales of virial estimates despite being poorly resolved are consistent with Cauchy stress reconstruction.

```
==========================================================================================
     The scale of Landshoff stress does not match expected shear stress.
      It should be investigated at the first place.
==========================================================================================
```

4.5. Plot the Landschoff forces (not stress) - for control

```
% === PLot Landshoff forces only
figure, hold on
hs = slice(Xw,Yw,Zw,FXYZgridx,[xw(1) boxcenter(1)],...
    [yw(2) boxcenter(2) yw(end)],...
    [fluidbox(3,1) boxcenter(3)]);
set(hs,'edgecolor','none','facealpha',0.9)
axis equal, view(3),
hc = colorbar('AxisLocation','in','fontsize',14); hc.Label.String =
'Landshoff along X';
xlabel('X'), ylabel('Y'), zlabel('Z')
title('Landshoff forces along X','fontsize',20)
step = [3 5 5];
quiver3( ...
    Xw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Yw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Zw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridx(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridy(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridz(1:step(1):end,1:step(2):end,1:step(3):end) ...
    ,1,'color','k','LineWidth',2)
view(90,90)
```



The top view illustrate the symmetry of the forces. The color shows the extent of the x-component and the arrows the direction and magnitude of the forces.,The previously presented stress estimates are consistent with forces.

5. Hertz contact stresses interpreted as fluid-solid shear stress

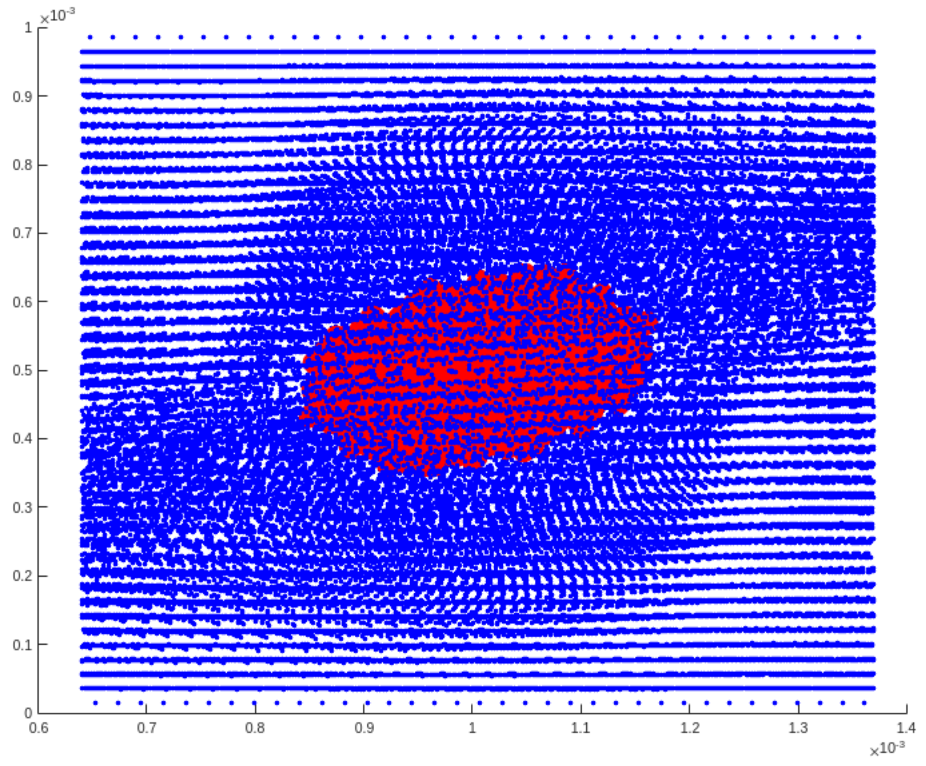
This section repeats analyzes introduced in `example2` on triangular meshes.

5.1 Computation part

The values `Rfluid` and `Rsolid` deserve clarificaiton for more accurate Hertz contact determination.

```
%% Hertz contact solid-fluid
% ===========================
XFluidSolid = Xstress.ATOMS(union(ifluid,isolid),:);
figure, hold on
```

```
plot3D(XFluidSolid{XFluidSolid.isfluid,coords},'bo','markersize',3,'markerfac
ecolor','b')
plot3D(XFluidSolid{XFluidSolid.issolid,coords},'ro','markersize',16,'markerfa
cecolor','r')
```



```
Rfluid = 1.04e-5; % m
Rsolid = 1.56e-5; % m
Rfluid = Rsolid;
hhertz = 2*Rsolid;
[Vcontactsolid,~,dmincontact] = buildVerletList(XFluidSolid,hhertz,[],[],
[],XFluidSolid.isfluid,XFluidSolid.issolid);
```

```
Build Verlet list by searching in blocks...
... done in 0.1321 s with 80 search blocks | minimum distance Inf
     Sort the Verlet list...
     ... done in 0.115 s
buildVerletList: all done in 0.291 s for 78945 atoms
```

```
configHertz = struct('R',{Rsolid
Rfluid},'E',2000,'rho',1000,'m',9.04e-12,'h',hhertz);
[FHertzSolid,WHertzSolid] =
forceHertz(XFluidSolid,Vcontactsolid,configHertz);
fhertz = sqrt(sum(FHertzSolid.^2,2));
statvec(fhertz(fhertz>0),'Hertz',sprintf('<-- TIMESTEP:
%d\n\tsubjected to Rsolid=[%0.4g %0.4g] dmin/
2=%0.4g',timestepforstress,configHertz(1).R,configHertz(2).R,dmincontact/2))
```

```
Hertz:  2.5%>  1.667e-08 |  25.0%>  4.027e-08 |  50.0%>  6.772e-08 |  75.0%>  8.849e-08 |  97.5%>   1.05e-
    subjected to Rsolid=[1.56e-05 1.56e-05] dmin/2=Inf
```

```
% project Hertz contact on accurate grid (Cartesian)
soliddbox = [ min(Xstress.ATOMS{isolid,coords})-4*rbead
              max(Xstress.ATOMS{isolid,coords})+4*rbead ]';
boxcenter = mean(soliddbox,2);
resolution = ceil(50 * diff(soliddbox,[],2)'./max(diff(soliddbox,[],2)));
xw = linspace(soliddbox(1,1),soliddbox(1,2),resolution(1));
yw = linspace(soliddbox(2,1),soliddbox(2,2),resolution(2));
zw = linspace(soliddbox(3,1),soliddbox(3,2),resolution(3));
[Xw,Yw,Zw] = meshgrid(xw,yw,zw);
XYZgrid = [Xw(:),Yw(:),Zw(:)];
VXYZ = buildVerletList({XYZgrid XFluidSolid{:,coords}},1.1*hhertz);
```

```
Build Verlet list by searching in blocks...
... done in 0.001586 s with 80 search blocks | minimum distance 3.774e-07
     Sort the Verlet list...
     ... done in 0.38 s
buildVerletList: all done in 8.71 s for 191745 grid points
```

```
W = kernelSPH(hhertz,'lucy',3); % kernel for interpolation (not gradkernel!)
FXYZgrid =
interp3SPHVerlet(XFluidSolid{:,coords},FHertzSolid,XYZgrid,VXYZ,W,Vbead);
```

```
INTERP3SPHVERLET interpolates 112800 x 3 grid points with a Verlet list including from 11 to 25 neighbors.
...done in 5.573 s. INTERP3SPHVerlet completed the interpolation of 112800 points with 78945 kernels
```

```
FXYZgridx = reshape(FXYZgrid(:,1),size(Xw));
FXYZgridy = reshape(FXYZgrid(:,2),size(Yw));
FXYZgridz = reshape(FXYZgrid(:,3),size(Zw));
WXYZgrid2 = interp3cauchy(Xw,Yw,Zw,FXYZgridx,FXYZgridy,FXYZgridz);
```

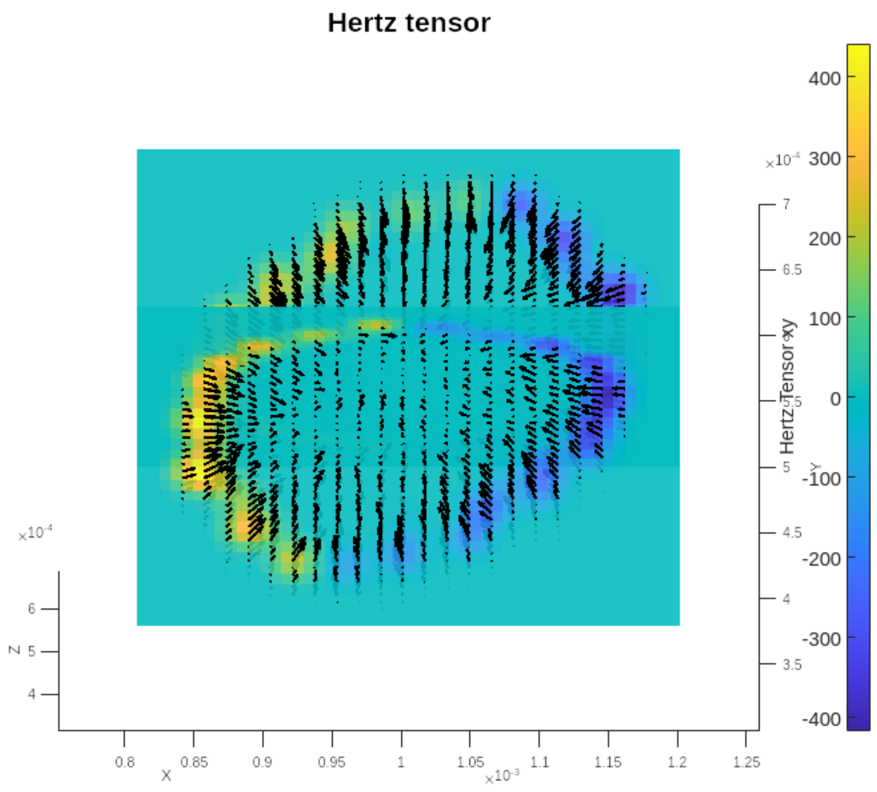
```
Calculate the local Cauchy stress from a [47 x 50 x 48] grid...
     ... done in 4.24 s
```

```
s12grid2 = reshape(WXYZgrid2(:,:,:,2),size(Xw));
```

5.2 Rapid plot of Hertz forces around the solid particle

```
%% Rough plot of Hertz Contact Tensor (component xy) - SOLID-FLUID
figure, hold on
hs = slice(Xw,Yw,Zw,s12grid2, ...
    boxcenter(1),...
    boxcenter(2),...
    boxcenter(3));
set(hs,'edgecolor','none','facealpha',0.9)
axis equal, view(3),
hc = colorbar('AxisLocation','in','fontsize',14); hc.Label.String = 'Hertz
Tensor xy';
```

```
xlabel('X'), ylabel('Y'), zlabel('Z')
title('Hertz tensor','fontsize',20)
step = [2 2 2];
quiver3( ...
    Xw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Yw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Zw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridx(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridy(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridz(1:step(1):end,1:step(2):end,1:step(3):end) ...
    ,1.5,'color','k','LineWidth',2)
view(0,72)
```



The tensor component *xy* is shown as colors. The arrows represent the Hertz forces.

5.3. Advanced shear-stress projected onto the solid particle

```
%% Advanced plot for Hertz Contacts
% almost same code as example2:

% === STEP 1/5 === Original Delaunay triangulation and the convex hull
% this step has been already done as the begining of example2bis.m
% DT = delaunayTriangulation(double(Xstress.ATOMS{isolidcontact, coords}));
% K = convexHull(DT);
% === STEP 2/5 === refine the initial mesh by adding midpoints
% Extract the convex hull points and faces
```

```
hullPoints = DT.Points;
hullFaces = DT.ConnectivityList(K, :);
% Initialize a set to keep track of midpoints to ensure they are unique
midpointSet = zeros(0, 3);
% Calculate midpoints for each edge in each triangle and add to the point
list
for faceIdx = 1:size(hullFaces, 1)
    face = hullFaces(faceIdx, :);
    for i = 1:3
        for j = i+1:3
            midpoint = (hullPoints(face(i), :) + hullPoints(face(j), :)) /
2;
            if isempty(midpointSet) || ~ismember(midpoint, midpointSet,
'rows')
                midpointSet = [midpointSet; midpoint]; %#ok<AGROW>
            end
        end
    end
end
% Merge the original points and the new midpoints, update the the Delaunay
triangulation
newDT = delaunayTriangulation([hullPoints; midpointSet]);
newK = convexHull(newDT);
% === STEP 3/5 ===  Laplacian Smoothing
points = newDT.Points;          % === Extract points and faces
faces = newDT.ConnectivityList(newK, :);
n = size(points, 1);            % === Initialize new points
newPoints = zeros(size(points));
neighbors = cell(n, 1);         % List of neighbors
for faceIdx = 1:size(faces, 1) % === Find the neighbors of each vertex
    face = faces(faceIdx, :);
    for i = 1:3
        vertex = face(i);
        vertex_neighbors = face(face ~= vertex);
        neighbors{vertex} = unique([neighbors{vertex}; vertex_neighbors(:)]);
    end
end
for i = 1:n                    % === Laplacian smoothing
    neighbor_indices = neighbors{i};
    if isempty(neighbor_indices) % Keep the point as is if it has no
neighbors
        newPoints(i, :) = points(i, :);
    else % Move the point to the centroid of its neighbors
        newPoints(i, :) = mean(points(neighbor_indices, :), 1);
    end
end
% Update the Delaunay triangulation with the smoothed points
newDT = delaunayTriangulation(newPoints);
newK = convexHull(newDT);
% === STEP 4/5 === Interpolate the Hertz forces on the triangular mesh
```

```
XYZhtri = newDT.Points;
XYZh = XFluidSolid{:,coords}; % kernel centers
VXYZh = buildVerletList({XYZhtri XFluidSolid{:,coords}},1.1*hhertz); %
special grid syntax
```

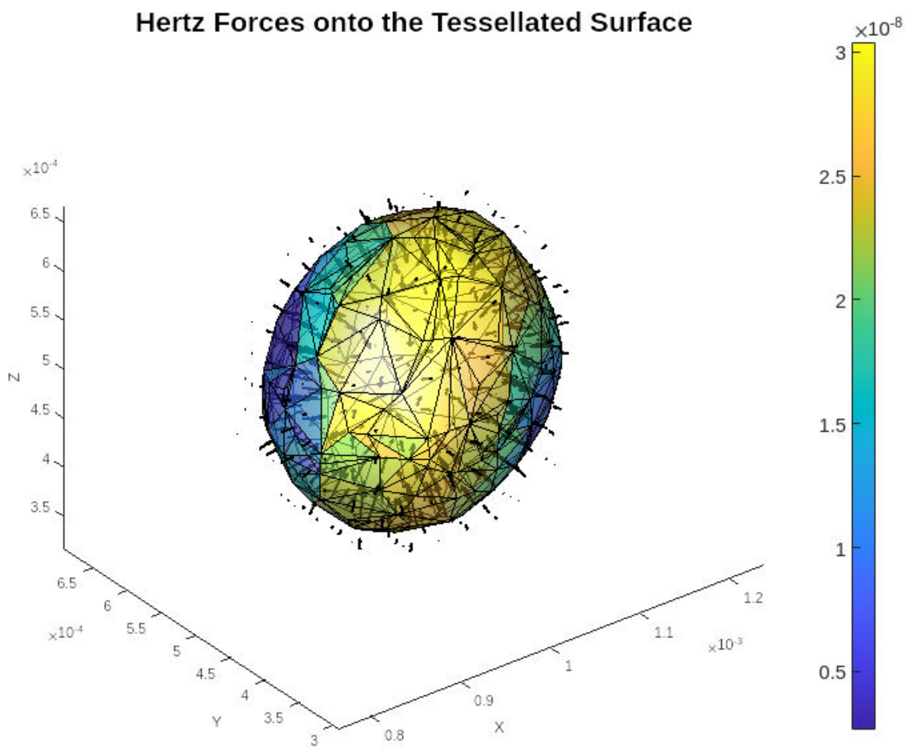
```
Build Verlet list by searching in blocks...
... done in 0.1746 s with 80 search blocks | minimum distance 0
     Sort the Verlet list...
     ... done in 0.122 s
buildVerletList: all done in 0.346 s for 80026 grid points
```

```
W = kernelSPH(hhertz,'lucy',3); % kernel expression
FXYZtri =
interp3SPHVerlet(XFluidSolid{:,coords},FHertzSolid,XYZhtri,VXYZh,W,Vbead);
```

```
INTERP3SPHVERLET interpolates 1081 x 3 grid points with a Verlet list including from 13 to 23 neighbors...
...done in 0.05759 s. INTERP3SPHVerlet completed the interpolation of 1081 points with 78945 kernels
```

```
% === STEP 5/5 === Extract tagential forces
% Calculate face normals and centroids
points = newDT.Points;
faces = newK;
v1 = points(faces(:, 1), :) - points(faces(:, 2), :);
v2 = points(faces(:, 1), :) - points(faces(:, 3), :);
faceNormals = cross(v1, v2, 2);
faceNormals = faceNormals ./ sqrt(sum(faceNormals.^2, 2));
centroids = mean(reshape(points(faces, :), size(faces, 1), 3, 3), 3);
% Interpolate force at each centroid using scatteredInterpolant for each
component
FInterp_x = scatteredInterpolant(XYZhtri, double(FXYZtri(:,1)), 'linear',
'nearest');
FInterp_y = scatteredInterpolant(XYZhtri, double(FXYZtri(:,2)), 'linear',
'nearest');
FInterp_z = scatteredInterpolant(XYZhtri, double(FXYZtri(:,3)), 'linear',
'nearest');
FXYZtri_at_centroids = [FInterp_x(centroids), FInterp_y(centroids),
FInterp_z(centroids)];
% Calculate normal and tangential components of the force at each face
centroid
normalComponent = dot(FXYZtri_at_centroids, faceNormals, 2);
normalForce = repmat(normalComponent, 1, 3) .* faceNormals;
tangentialForce = FXYZtri_at_centroids - normalForce;
tangentialMagnitude = sqrt(sum(tangentialForce.^2, 2));
% Do the figure
figure, hold on
trisurfHandle = trisurf(newK, newDT.Points(:, 1), newDT.Points(:, 2),
newDT.Points(:, 3), 'Edgecolor', 'k', 'FaceAlpha', 0.6);
set(trisurfHandle, 'FaceVertexCData', tangentialMagnitude, 'FaceColor',
'flat');
colorbar;
% Quiver plot to show the forces with an adjusted step
step = [2 2 2]*2;
```

```
quiver3( ...
    Xw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Yw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Zw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridx(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridy(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridz(1:step(1):end,1:step(2):end,1:step(3):end) ...
    ,1.5,'color','k','LineWidth',2)
axis equal; view(3), camlight('headlight'); camlight('left'); lighting
phong; colorbar;
title('Hertz Forces onto the Tessellated Surface','fontsize',20);
xlabel('X'); ylabel('Y'); zlabel('Z')
hc = colorbar('AxisLocation','in','fontsize',14);
```



The colors represent the component tangential to the surface. The arrows represent the forces at the refined mesh as estimated from the grid interpolation.

6. Hertz contact stress interpreted as shear-stress at walls

6.1. Computation part

```
%% Hertz contact wall-fluid
% ==========================
XFluidWall = Xstress.ATOMS(union(ifluid,iwall),:);
figure, hold on
```

```
plot3D(XFluidWall{XFluidWall.isfluid,coords},'bo','markersize',3,'markerfacec
olor','b')
plot3D(XFluidWall{XFluidWall.iswall,coords},'ko','markersize',16,'markerfacec
olor','k')
Rfluid = 1.04e-5; % m
Rsolid = 1.56e-5; % m
Rfluid = Rsolid;
hhertz = 2*Rsolid;
[Vcontactwall,~,dmincontact] = buildVerletList(XFluidWall,hhertz,[],[],
[],XFluidWall.isfluid,XFluidWall.iswall);
```

```
Build Verlet list by searching in blocks...
... done in 0.0343 s with 180 search blocks | minimum distance Inf
     Sort the Verlet list...
     ... done in 0.174 s
buildVerletList: all done in 0.735 s for 112674 atoms
```

```
configHertz = struct('R',{Rsolid
Rfluid},'E',2000,'rho',1000,'m',9.04e-12,'h',hhertz);
[FHertzWall,WHertzWall] = forceHertz(XFluidWall,Vcontactwall,configHertz);
fhertz = sqrt(sum(FHertzWall.^2,2));
statvec(fhertz(fhertz>0),'Hertz',sprintf('<-- TIMESTEP:
%d\n\tsubjected to Rsolid=[%0.4g %0.4g] dmin/
2=%0.4g',timestepforstress,configHertz(1).R,configHertz(2).R,dmincontact/2))
```

```
Hertz:  2.5%>   2.44e-07 |  25.0%>  3.923e-07 |  50.0%>  3.924e-07 |  75.0%>  3.926e-07 |  97.5%>   3.93e-
    subjected to Rsolid=[1.56e-05 1.56e-05] dmin/2=Inf
```

```
figure, stem3(XFluidWall{:,'x'},XFluidWall{:,'y'},FHertzWall(:,1),'k.')

% project Hertz contact on accurate grid (Cartesian)
[wallbox1,wallbox2] = deal(fluidbox);
wallbox1(2,1) = fluidbox(2,1) - 1.5*hhertz;
wallbox1(2,2) = fluidbox(2,1) + 1.5*hhertz;
wallbox2(2,1) = fluidbox(2,2) - 1.5*hhertz;
wallbox2(2,2) = fluidbox(2,2) + 1.5*hhertz;
boxcenter = mean(fluidbox,2);
boxcenter1 = mean(wallbox1,2);
boxcenter2 = mean(wallbox2,2);
resolution1 = ceil(120 * diff(wallbox1,[],2)'./max(diff(wallbox1,[],2)));
resolution2 = ceil(120 * diff(wallbox2,[],2)'./max(diff(wallbox2,[],2)));
xw = linspace(wallbox1(1,1),wallbox1(1,2),resolution1(1));
yw1 = linspace(wallbox1(2,1),wallbox1(2,2),resolution1(2));
yw2 = linspace(wallbox2(2,1),wallbox2(2,2),resolution2(2));
zw = linspace(wallbox1(3,1),wallbox1(3,2),resolution1(3));
[Xw,Yw,Zw] = meshgrid(xw,[yw1 boxcenter(2) yw2],zw);
XYZgrid = [Xw(:),Yw(:),Zw(:)];
VXYZ = buildVerletList({XYZgrid XFluidWall{:,coords}},1.1*hhertz);
```

```
Build Verlet list by searching in blocks...
... done in 0.4589 s with 180 search blocks | minimum distance 9.889e-08
     Sort the Verlet list...
     ... done in 0.812 s
buildVerletList: all done in 25.5 s for 376674 grid points
```

```
W = kernelSPH(hhertz,'lucy',3); % kernel for interpolation (not gradkernel!)
FXYZgrid =
interp3SPHVerlet(XFluidWall{:,coords},FHertzWall,XYZgrid,VXYZ,W,Vbead);
```

```
INTERP3SPHVERLET interpolates 264000 x 3 grid points with a Verlet list including from 0 to 27 neighbors..
...done in 13.38 s. INTERP3SPHVerlet completed the interpolation of 264000 points with 112674 kernels
```

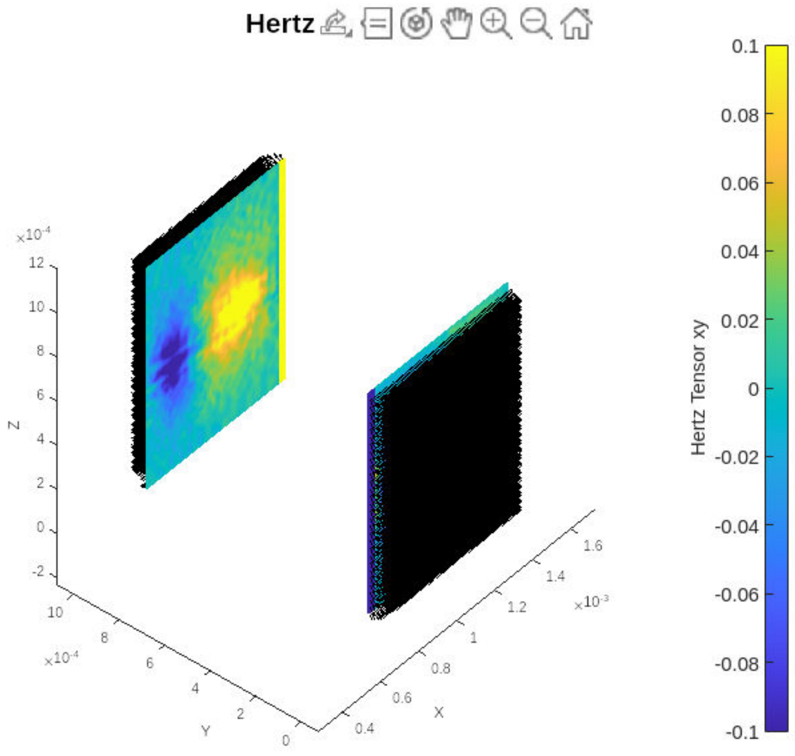
```
FXYZgridx = reshape(FXYZgrid(:,1),size(Xw));
FXYZgridy = reshape(FXYZgrid(:,2),size(Yw));
FXYZgridz = reshape(FXYZgrid(:,3),size(Zw));
WXYZgrid2 = interp3cauchy(Xw,Yw,Zw,FXYZgridx,FXYZgridy,FXYZgridz);
```

```
Calculate the local Cauchy stress from a [25 x 88 x 120] grid...
     ... done in 9.99 s
```

```
s12grid2 = reshape(WXYZgrid2(:,:,:,2),size(Xw));
```

6.2 Rough plot of Hertz stress at walls

```
%% Rough plot of Hertz Contact Tensor (component xy)
figure, hold on
hs = slice(Xw,Yw,Zw,s12grid2, ...
    single([]),...
    [fluidbox(2,1)-[-.1 0 .1 0.5 1]*rbead/2 ,fluidbox(2,2)+[-.1 0 .1 0.5
1]*rbead/2],...
    single([]));
set(hs,'edgecolor','none','facealpha',0.9)
axis equal, view(3),
hc = colorbar('AxisLocation','in','fontsize',14); hc.Label.String = 'Hertz
Tensor xy';
xlabel('X'), ylabel('Y'), zlabel('Z')
title('Hertz tensor','fontsize',20)
step = [4 1 4];
quiver3( ...
    Xw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Yw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    Zw(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridx(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridy(1:step(1):end,1:step(2):end,1:step(3):end), ...
    FXYZgridz(1:step(1):end,1:step(2):end,1:step(3):end) ...
    ,.5,'color','k','LineWidth',.1)
caxis([-.1 .1]), view([-50,42])
```



The xy Hertz resulting tensor stress is plotted as color for both wall surfaces with forces shown as quiver plots.