This document[1] describes SMOOTH MACH DYNAMICS, a meshless simulation package for solving problems in continuum mechanics. SMOOTH MACH DYNAMICS is provided as the USER-SMD package within the LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator particle code [2]. This document and the SMOOTH MACH DYNAMICS software are distributed under the GNU General Public License.

✧ ✧ ✧

```
/* ----------------------------------------------------------------------
 *
 *                    *** Smooth Mach Dynamics ***
 *
 * This file is part of the USER-SMD package for LAMMPS.
 * Copyright (2014) Georg C. Ganzenmueller, georg.ganzenmueller@emi.fhg.de
 * Fraunhofer Ernst-Mach Institute for High-Speed Dynamics, EMI,
 * Eckerstrasse 4, D-79104 Freiburg i.Br, Germany.
 * This software is distributed under the GNU General Public License.
 *
 * ---------------------------------------------------------------------- */
```

✧ ✧ ✧

Acknowledgements

Acknowledgements go to Prof. Dr. Stefan Hiermaier at the Fraunhofer Ernst-Mach Institute for High-Speed Dynamics, who provided the resources and motivation for developing SMOOTH MACH DYNAMICS.

TODO

This document is a preliminary draft version as of now. A lot remains to be done.

[1] Draft Version 0.12, February 18, 2016
[2] http://lammps.sandia.gov

Contents

1

Smooth-Particle Hydrodynamics Theory review

Smooth-Particle Hydrodynamics (SPH) was first developed in the seventies by Lucy [1] and Gingold and Monaghan [2] in the astrophysical context as a means to simulate the formation of stars. The principle of SPH is to approximate a continuous field using a discrete set of kernel functions which are centred about so-called particles, where the physical properties of the system, e.g. mass, internal energy, or velocity, are located. When the SPH approximation is applied to fluid flow or solid body deformation, solutions to the underlying set of partial differential equations are obtained in terms of simple algebraic equations. As no auxiliary computational grid is used to construct the solution, SPH is termed a mesh-free method. The absence of a mesh implies that arbitrarily large deformations and instability phenomena such as fracture can be handled with ease when compared to mesh-based techniques such as the Finite Element method (FEM). FEM requires geometrically well-defined mesh cells and certain assumptions regarding the smoothness of the field within these cells. Because both of these requirements are typically violated in the simulation of important engineering applications such as impact, explosion, or machine cutting, a continued interest in the development of meshfree methods prevails.

The application of SPH to fluid problems with free boundaries has been very successful. In particular, solutions to large-scale gas dynamic problems have been obtained in astrophysics [3], and fluid-structure interaction in civil engineering applications, such as the impact of a water wave on coastal structures, has been treated with success [4]. However, for solid body deformations, the situation is not equally satisfying. In 1991, Libersky [5] was the first to simulate a body with material strength using SPH. Strong numerical instability issues appeared which prevented SPH from becoming a serious competitor to mesh-based methods for solid continua. Different sources of instability were identified by early works: Swegle [6] noted that the interaction of the second derivative of the kernel and the tensile stress resulted in nonphysical clumping of particles, which he termed *tensile instability*. Dyka *et al* [7] observed that the nodal integration approach inherent to SPH incurs instabilities. In essence, the number of integration points is too small such that the solution to the underlying equilibrium equation becomes non-unique due to rank deficiency. They proposed to eliminate the rank-deficiency by introducing additional integration points at other locations that the particles themselves and noted that this kind of instability is also observed in FEM, when elements with a reduced number of integration points are used. In FEM, the instability emerging from this rank-deficiency problem is termed hour-glassing. Rank deficiency occurs regardless of the state of stress and in addition to the tensile instability. A number of different schemes were devised to increase the stability of SPH. Artificial viscosity and Riemann-type solvers increase numerical stability by dissipating high-frequency modes. Conservative smoothing [8] and the XSPH time integration scheme [9] are dispersive rather than dissipative but also work by removing high-frequency modes. Randles *et al* [10] elaborated on the idea of introducing additional stress points to remove the rank-deficiency problem. The clumping problem associated with tensile instability was addressed by Gray *et al.* [11] by adding repulsive forces between SPH particles if the principal stresses are tensile. However, none of these approaches turned SPH into a simulation method that is generally stable for a broad range of applications.

A turning point was achieved by Belytschko *et al.* [12], who showed that the Eulerian character of the kernel function (other particles pass through a particle's kernel domain as the simulation proceeds) in combination with the Lagrangian character of the moving SPH particles (they move in a

fixed frame of reference) is the cause of the tensile instability. They proposed a Lagrangian formulation where the kernel approximation is performed in the initial, undeformed reference coordinates of the material. In this Lagrangian formulation, the tensile instability is absent, however, other instabilities due to rank-deficiency caused by the collocation method remain. Belytschko *et al* also showed that the remaining instabilities can be removed by the addition of stress points, but the locations of the stress points needs to be carefully chosen. The invention of Lagrangian SPH has prompted a revived interest in this particular meshless method, resulting in several studies which confirm its enhanced stability [13, 14, 15, 16, 17]. However, the idea of using additional integration points appears not to have found widespread usage in the SPH community. Possible causes might be related to the increased computational effort required for evaluating the stresses and a lack of information as to where stress points should be placed if irregular particle positions are employed for the reference configuration.

In this work, it is demonstrated how instabilities caused by the rank-deficiency can be directly controlled. The inspiration for taking such an approach to stabilise the solution originates from ideas developed for the Finite Element Method (FEM). There, elements with a reduced number of integration points are routinely employed because they are computationally very effective and avoid the shear locking problems of fully integrated elements. Such reduced-integrated elements are susceptible to so-called hourglass modes, which are zero-energy modes in the sense that the element deforms without an associated increase of the elastic (potential) energy. These modes cannot be detected if a reduced number of integration points is used, and can therefore be populated with arbitrary amounts of kinetic energy, such that the solution is entirely dominated by theses modes. A common approach to suppress the hour-glassing modes is to identify them as the non-linear part of the velocity field and penalise them by appropriate means. It is difficult in general to seek analogies between the SPH collocation method and FEM. However, in the case of the so-called mean (or constant) stress element [18], which uses only one integration point to represent the average stress state within the entire element, there exists a clear analogy to the weighted average over the neighbouring particles that is obtained in SPH. It is this analogy that will be exploited in order to develop a zero-energy mode suppression algorithm for SPH.

The remainder of this article is organised as follows. In the next section, a brief review of the Lagrangian SPH formalism is given. This is followed by the details as to how an SPH analogue of the non-linear part of the deformation field can be used to obtain an algorithm which effectively suppresses zero-energy modes. The usefulness of the stabilisation algorithm is subsequently demonstrated with a number of large strain deformation examples, that are difficult, if not impossible, to obtain using Lagrangian SPH without additional stress points. Finally, the implications of this particular type of stabilisation technique are discussed, and an outlook is given regarding possible improvements.

1.1 The total Lagrangian formulation

In the Total Lagrangian formulation, conservation equations and constitutive equations are expressed in terms of the reference coordinates $\boldsymbol{X}$, which are taken to be the coordinates of the initial, undeformed reference configuration. A mapping ϕ between the current coordinates, and the reference coordinates describes the body motion at time t:

$$\boldsymbol{x} = \phi(\boldsymbol{X}, t), \tag{1.1}$$

Here, $\boldsymbol{x}$ are the current, deformed coordinates and $\boldsymbol{X}$ the reference (Lagrangian) coordinates. The displacement $\boldsymbol{u}$ is given by

$$\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{X}, \tag{1.2}$$

Note that bold mathematical symbols like the preceding ones denote vectors or tensors, while the same mathematical symbol in non-bold font refers to their respective Euclidean norm, e.g. $x = |\boldsymbol{x}|$. The conservation equations for mass, impulse, and energy in the total Lagrangian formulation are

given by

$$\rho J = \rho_0 \tag{1.3}$$

$$\ddot{\boldsymbol{u}} = \frac{1}{\rho_0}\boldsymbol{\nabla}_0 \cdot \boldsymbol{P}^T \tag{1.4}$$

$$\dot{e} = \frac{1}{\rho_0}\dot{\boldsymbol{F}} : \boldsymbol{P}, \tag{1.5}$$

where ρ is the mass density, $\boldsymbol{P}$ is the first Piola-Kirchhoff stress tensor, e is the internal energy, and $\boldsymbol{\nabla}$ is the gradient or divergence operator. The subscript 0 indicates that a quantity is evaluated in the reference configuration, while the absence of this subscript means that the current configuration is to be used. J is the determinant of the deformation gradient $\boldsymbol{F}$,

$$\boldsymbol{F} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{X}} = \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{X}} + \boldsymbol{I}, \tag{1.6}$$

which can be interpreted as the transformation matrix that describes the rotation and stretch of a line element from the reference configuration to the current configuration.

1.2 The SPH Approximation in the total Lagrangian formulation

The SPH approximation for a scalar function f in terms of the reference coordinates can be written as

$$f(\boldsymbol{X}_i) = \sum_{j \in \mathscr{S}} V_j^0 f(\boldsymbol{X}_j) W_i\left(\boldsymbol{X}_{ij}\right) \tag{1.7}$$

The sum extends over all particles within the range of a scalar weight function W_i, which is centred at position $\boldsymbol{X}_i$ and depends only on the distance vector between coordinates $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$. Here, exclusively radially symmetric kernels are considered, i.e., $W_i\left(\boldsymbol{X}_{ij}\right) = W_i\left(X_{ij}\right)$ which depend only on the scalar distance between particles i and j. V^0 is the volume associated with a particle in the reference configuration. The weight function is chosen to have compact support, i.e., it includes only neighbours within a certain radial distance. This domain of influence is denoted $\mathscr{S}$.

The SPH approximation of a derivative of f is obtained by operating directly with the gradient operator on the kernel functions,

$$\boldsymbol{\nabla}_0 f(\boldsymbol{X}_i) = \sum_{j \in \mathscr{S}} V_j^0 f(\boldsymbol{X}_j) \boldsymbol{\nabla} W_i\left(X_{ij}\right), \tag{1.8}$$

where the gradient of the kernel function is defined as follows:

$$\boldsymbol{\nabla} W_i(X_{ij}) = \left(\frac{\mathrm{d}W(X_{ij})}{\mathrm{d}X_{ij}}\right)\frac{\boldsymbol{X}_j - \boldsymbol{X}_i}{X_{ij}} \tag{1.9}$$

It is of fundamental interest to characterise numerical approximation methods in terms of the order of completeness, i.e., the order of a polynomial that can be exactly approximated by the method. For solving the conservation equations with its differential operators, at least first-order completeness is required. In the case of the SPH approximation, the conditions for zeroth- and first-order completeness are stated as follows:

$$\sum_{j \in \mathscr{S}} V_j^0 W_i\left(X_{ij}\right) = 1 \tag{1.10}$$

$$\sum_{j \in \mathscr{S}} V_j^0 \boldsymbol{\nabla} W_i\left(X_{ij}\right) = 0 \tag{1.11}$$

The basic SPH approach given by equations (1.7) and (1.8) fulfils neither of these completeness conditions. An *ad-hoc* improvement by Monaghan [19] consists of adding eqn. (1.11) to eqn. (1.8), such that a *symmetrized* approximation for the derivative of a function is obtained,

$$\boldsymbol{\nabla}_0 f(\boldsymbol{X}_i) = \sum_{j \in \mathscr{S}} V_j^0 \left(f(\boldsymbol{X}_j) - f(\boldsymbol{X}_i)\right) \boldsymbol{\nabla} W_i\left(X_{ij}\right) \tag{1.12}$$

The symmetrisation does not result in first-order completeness, however, it yields zeroth-order completeness for the derivatives of a function, even in the case of irregular particle arrangements [15].

1.3 First-Order Completeness

In order to fulfil first-order completeness, the SPH approximation has to reproduce the constant gradient of a linear field. A number of correction techniques [10, 20, 21] exploit this condition as the basis for correcting the gradient of the SPH weight function,

$$\sum_{j \in \mathscr{S}} V_j^0 (\boldsymbol{X}_j - \boldsymbol{X}_i) \otimes \boldsymbol{\nabla} W_i(X_{ij}) \stackrel{!}{=} \boldsymbol{I}, \tag{1.13}$$

where $\boldsymbol{I}$ is the diagonal unit matrix. Based on this expression, a corrected kernel gradient can be defined:

$$\tilde{\boldsymbol{\nabla}} W_i(X_{ij}) = \boldsymbol{L}_i^{-1} \boldsymbol{\nabla} W_i(X_{ij}), \tag{1.14}$$

which uses the correction matrix $\boldsymbol{L}$, given by:

$$\boldsymbol{L}_i = \sum_{j \in \mathscr{S}} V_j^0 \boldsymbol{\nabla} W_i(X_{ij}) \otimes (\boldsymbol{X}_j - \boldsymbol{X}_i). \tag{1.15}$$

By construction, the corrected kernel gradient now satisfy eqn. (1.13),

$$\sum_{j \in \mathscr{S}} V_j^0 (\boldsymbol{X}_j - \boldsymbol{X}_i) \otimes \boldsymbol{L}_i^{-1} \boldsymbol{\nabla} W_i(X_{ij}) = \boldsymbol{I}, \tag{1.16}$$

resulting in first-order completeness.

1.4 Total-Lagrangian SPH expressions for Solid Mechanics

For calculating the internal forces of a solid body subject to deformation, expressions are required for (i) the deformation gradient, (ii) a constitutive equation which provides a stress tensor as function of the deformation gradient, and (iii) an expression for transforming the stresses into forces acting on the nodes which serve as the discrete representation of the body.

1.4.1 the Deformation Gradient and its time derivative

The deformation gradient is obtained by calculating the spatial derivative of the displacement field, i.e. by using the symmetrized SPH derivative approximation, eqn. (1.12), for eqn. (1.6):

$$\boldsymbol{F}_i = \sum_{j \in \mathscr{S}} V_j^0 (\boldsymbol{u}_j - \boldsymbol{u}_i) \otimes \boldsymbol{L}_i^{-1} \boldsymbol{\nabla} W_i(X_{ij}) + \boldsymbol{I}. \tag{1.17}$$

Note that in the above equation, the corrected kernel gradients have been introduced. Similarly, the time derivative of $\boldsymbol{F}$ is obtained by considering the spatial derivative of the velocity field:

$$\dot{\boldsymbol{F}}_i = \sum_{j \in \mathscr{S}} V_j^0 (\boldsymbol{v}_j - \boldsymbol{v}_i) \otimes \boldsymbol{L}_i^{-1} \boldsymbol{\nabla} W_i(X_{ij}). \tag{1.18}$$

1.4.2 Constitutive models and time integration of the stress rate

The constitutive model is independent of the numerical discretization and therefore no essential part of the SPH method. However, some important relations are quoted for clarity and the reader is referred to the excellent textbooks by Bonet and Wood [22] or Belytschko *et al* [23]. In `USER-SMD`, constitutive models are expressed using a strain rate and the Cauchy stress rate. The Cauchy stress is obtained by integrating the stress rate in time. This approach allows for a proper handling of non-linear

material behaviour, such as plasticity or damage. However, the time integration of the stress rate is plagued by the problem of *non-objectivity*, which can be summarised by the following inequality:

$$\boldsymbol{\sigma} \neq \int \dot{\boldsymbol{\sigma}} \mathrm{d}t. \tag{1.19}$$

The problem is caused by the presence of finite rotations in the stress rate. As a solution, we adopt the approach detailed in [24] and time-integrate an unrotated stress rate, which is obtained by subtracting the rotation components of the current deformation state. The correct Cauchy stress is subsequently rotated back to to agree with the current deformation of the system. To this end, we obtain the velocity gradient as

$$\boldsymbol{L} = \dot{\boldsymbol{F}}\boldsymbol{F}^{-1}, \tag{1.20}$$

from which we compute the rate-of-deformation tensor,

$$\boldsymbol{D} = \frac{1}{2}(\boldsymbol{L} + \boldsymbol{L}^T). \tag{1.21}$$

The rotation is obtained from a polar decomposition of the deformation gradient,

$$\boldsymbol{F} = \boldsymbol{R}\boldsymbol{U}, \tag{1.22}$$

where $\boldsymbol{R}$ is a pure rotation tensor, i.e. $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$ and $\det(\boldsymbol{R}) = 1$. With $\boldsymbol{R}$ known, the unrotated part of the rate-of-deformation tensor, i.e. the stretch rate tensor is computed:

$$\boldsymbol{d} = \boldsymbol{R}^T\boldsymbol{D}\boldsymbol{R}. \tag{1.23}$$

A constitutive model relates the unrotated stress rate to the stretch rate tensor. For illustratory purposes we consider Hookean linear elasticity, with the Lamé parameters λ and μ:

$$\dot{\boldsymbol{\sigma}}^u = \lambda \mathrm{Tr}\{\boldsymbol{d}\} + 2\mu\boldsymbol{d} \tag{1.24}$$

Note that $\mathrm{Tr}\{\boldsymbol{d}\} = d_{ii}/3$ denotes the trace of $\boldsymbol{d}$. The unrotated stress is the time integral of the unrotated stress rate,

$$\boldsymbol{\sigma}^u = \int \dot{\boldsymbol{\sigma}}^u \mathrm{d}t, \tag{1.25}$$

and we obtain the correct Cauchy stress by rotating the unrotated Cauchy stress back to the current deformation state:

$$\boldsymbol{\sigma} = \boldsymbol{R}\boldsymbol{\sigma}^u\boldsymbol{R}^T. \tag{1.26}$$

The Cauchy stress is the proper stress measure for the deformed configuration. However, in the total Lagrangian Formulation, nodal forces are applied by a stress integration method which is formulated using the reference coordinates. Therefore, a stress measure is required which links the stress in the reference configuration to the current configuration. This stress measure is the First Piola-Kirchhoff stress, given by

$$\boldsymbol{P} = J\boldsymbol{\sigma}\boldsymbol{F}^{-T}. \tag{1.27}$$

1.4.3 Nodal Forces

Nodal forces are obtained from an SPH approximation of the stress divergence, eqn. (1.4). Several different approximations can be obtained [25], depending on how the discretization is performed. The most frequently used expression, which is variationally consistent in the sense that it minimises elastic energy [20], is the following,

$$\boldsymbol{f}_i = \sum_{j \in \mathscr{S}} V_i^0 V_j^0 \left(\boldsymbol{P}_j + \boldsymbol{P}_i\right) \boldsymbol{\nabla} W_i(X_{ij}), \tag{1.28}$$

where the stress tensors are added to each other rather than subtracted from each other. For a radially symmetric kernel which depends only on distance, the anti-symmetry property $\nabla W_i(X_{ij}) = -\nabla W_j(X_{ji})$ holds. Therefore, the above force expression will conserve linear momentum exactly, as $\boldsymbol{f}_{ij} = -\boldsymbol{f}_{ji}$. The anti-symmetry property of the kernel gradient is used to rewrite the force expression as follows:

$$\boldsymbol{f}_i = \sum_{j\in\mathscr{S}} V_i^0 V_j^0 \left(\boldsymbol{P}_i \nabla W_i(X_{ij}) + \boldsymbol{P}_j \nabla W_i(X_{ij})\right) \tag{1.29}$$

$$= \sum_{j\in\mathscr{S}} V_i^0 V_j^0 \left(\boldsymbol{P}_i \nabla W_i(X_{ij}) - \boldsymbol{P}_j \nabla W_j(X_{ji})\right). \tag{1.30}$$

Replacing the uncorrected kernel gradients with the corrected gradients (c.f. eqn. (1.14), the following expression is obtained:

$$\boldsymbol{f}_i = \sum_{j\in\mathscr{S}} V_i^0 V_j^0 \left(\boldsymbol{P}_i \boldsymbol{L}_i^{-1} \nabla W_i(X_{ij}) - \boldsymbol{P}_j \boldsymbol{L}_j^{-1} \nabla W_j(X_{ji})\right) \tag{1.31}$$

This first-order corrected force expression also conserves linear momentum due to its anti-symmetry with respect to interchange of the particle indices i and j, i.e., $\boldsymbol{f}_{ij} = -\boldsymbol{f}_{ji}$. The here constructed anti-symmetric force expression is usually not seen in the literature. In contrast, it seems to be customary [10, 20, 21] to directly insert the corrected kernel gradient into eqn. (1.28), which destroys the local conservation of linear momentum. This section is summarised by noting that all expressions for Total-Lagrangian SPH have now been defined. The next section will introduce an SPH analogue of the hour-glassing control mechanism used in FEM.

1.5 Updating the reference configuration

The total Lagrangian formulation can be used for elastic deformations with significantly large strains. Under an elastic deformation, the topology, i.e., how material points are connected with each other, does not change. However, if plastic flow is considered, the topology does change. To reflect this change in topology, the reference configuration must be updated at certain time intervals. This is achieved in re-initialising the reference coordinates with the current coordinates, and updating the particle volume.

$$\boldsymbol{X} \leftarrow \boldsymbol{x} \tag{1.32}$$

$$V \leftarrow JV \tag{1.33}$$

Following these changes, the next evaluation of the deformation gradient will result in the identity matrix, as all displacements w.r.t the new reference configuration are zero. Nevertheless, the system has not lost its memory of its deformation, as the stress state is still known: In an attempt to minimise the stress, the system will try to return to its initial configuration. However, the strain information, which is computed from $\boldsymbol{F}$ would be lost. To circumvent this problem, we keep track of the deformation gradient before performing an update of the reference configuration:

$$\boldsymbol{F}_0 \leftarrow \boldsymbol{F}_0 \boldsymbol{F}.$$

Note that initially, $\boldsymbol{F}_0 = \boldsymbol{I}$. The current total deformation gradient, reflecting the deformation since the last update ($\boldsymbol{F}$) and all updates before the last update ($\boldsymbol{F}_0$) is then given by

$$\boldsymbol{F}_t = \boldsymbol{F}_0 \boldsymbol{F}. \tag{1.34}$$

1.6 The updated Lagrangian formulation

... This is work in progress.

2

Material Point Method Theory review

SMD provides an implementation of the *Generalized Interpolation Material Point Method* (GIMP), but we will refer to this simply as the *Material Point Method* (MPM) in this document. In short, MPM makes use of both particles and an auxiliary background grid. As such, features from Eularian grid based methods such as *Computational Fluid Dynamics* (CFD) and Lagrangian particle methods like Smooth Particle Hydrodynamics (SPH) are present in MPM. Note, however, that MPM still classifies as a meshfree method, becasue the background only serves as a computational scratchpad during one timestep and is completely deleted at the end of the timestep. In particular, different grids could be chosen between timesteps. All history-dependent variables are stored only on the particles. For more background information, the reader is referred to the works [26]
SMD utilizes a regular quadratic (or cubic in 3d) grid with node spacing h. Nodal positions are denoted by $\boldsymbol{x}_n$, while particle positions are denoted $\boldsymbol{x}_p$. An interpolation function W transfers information between grid and particles. W is of compact support, meaning that a particle can only interact with nearby nodes up to a distance of $2\,h$ along every cartesian direction. To this end, W is given by a dyadic product of of one-dimensional cubic B-splines:

$$W(\boldsymbol{x}_n - \boldsymbol{x}_p) = W(\boldsymbol{x}_p n) = W(x_n - x_p) \times W(y_n - y_p) \times W(z_n - z_p)$$

Here, (x_n, y_n, z_n) and (x_p, y_p, z_p) correspond to the entries of the vectors $\boldsymbol{x}_n$ and $\boldsymbol{x}_p$, respectively. The B-spline itself is defined in units of h:

$$W(r = x/h) = \begin{cases} \frac{1}{2}|r|^3 - |r|^2 + \frac{2}{3} & \text{if } 0 \le |r| < 1 \\ \frac{1}{2} - \frac{1}{6}|r|^3 + |r|^2 - 2|r| + \frac{4}{3} & \text{if } 1 \le |r| < 2 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

2.1 MPM workflow

At the beginning of a single MPM timestep at time t, the following information is located at the particles:

- velocities $\boldsymbol{v}_p^t$
- stress $\boldsymbol{\sigma}_p^t$
- deformation gradient $\boldsymbol{F}_p^t$
- volume V_p^t
- heat (thermal energy) q_p^t

These quantities are evolved in time according to the following steps. This particular MPM scheme is known as MUSL (Modified Update Stress Last) **cite Wallstedt time integration paper**.

1. Points to grid: Interpolate particle mass, velocity, and heat to nearby grid nodes.

2. Compute grid forces: Evlaute the divergence of the particle stress field with respect to nearby nodes.
3. Compute heat rate: Determine the second derivative of the spatial heat distribution.
4. Update grid: time-integrate grid velocities using grid forcess and grid heat using heat rate
5. Grid to points: Interpolate grid velocities and accelerations back to particles
6. Particle time integration: update particle positions and velocities using interpolated grid values.
7. Interpolate updated particle velocities a second time to grid nodes.
8. Compute velocity gradient at particle locations.
9. Update particle stress and strain states.

These steps are detailed below.

2.1.1 Points to grid

This is the initial stage of the MPM algorithm. Mass, velocities, and heat are interpolated to the grid nodes:

$$m_n^t = \sum_p W_{pn} m_p \tag{2.2}$$
$$q_n^t = \frac{1}{m_n} \sum_p W_{pn} m_p q_p^t \tag{2.3}$$
$$\boldsymbol{v}_n^t = \frac{1}{m_n} \sum_p W_{pn} m_p \boldsymbol{v}_n^t \tag{2.4}$$

Note that both heat and velocity are weighted by mass, which is neccessary for conservation of impulse and total energy.

2.1.2 Compute grid forces

$$\boldsymbol{f}_n^t = \sum_p V_p^t \sigma_p^t \boldsymbol{\nabla} W_{pn}$$

2.1.3 Compute heat rate

The heat rate $\dot{q}_n^t$ is given by the product of the thermal conduction coefficient, κ, and the spatial second derivative $\boldsymbol{\nabla} \cdot \boldsymbol{\nabla} q_n^t$ of the heat distribution.. The latter quantity is computed directly on the grid using central differences and a three point stencil.

2.1.4 Update grid quantities

Now that the grid node forces and the rate of heat are known, velocities and heat are time-integrated using a simple forward Euler update.

$$q_n^{t+\Delta t} = q_n^t + \Delta t \, \dot{q}_n^t \tag{2.5}$$
$$\boldsymbol{v}_n^{t+\Delta t} = \boldsymbol{v}_n^t + \frac{\Delta t}{m_p} \boldsymbol{f}_n^t \tag{2.6}$$

2.1.5 Grid to points

Interpolate the updated grid velocity, updated grid heat, and grid forces back to the particles.

$$q_p^{t+\Delta t} = \sum_n W_{pn} q_n^{t+\Delta t} \tag{2.7}$$

$$\tilde{\boldsymbol{v}}_p^{t+\Delta t} = \sum_n W_{pn} \boldsymbol{v}_n^{t+\Delta t} \tag{2.8}$$

$$\boldsymbol{f}_p^t = \sum_n W_{pn} \boldsymbol{f}_n^t \tag{2.9}$$

Note that the reverse interpolation of the velocity creates a new particle velocity $\tilde{\boldsymbol{v}}_p^{t+\Delta t}$ as indicated by the *tilde* decorator.

2.1.6 Particle time integration

Particle positions are updated using the interpolated grid velocities:

$$\boldsymbol{x}_p^{t+\Delta t} = \boldsymbol{x}_p^t + \Delta t\, \tilde{\boldsymbol{v}}_p^{t+\Delta t}$$

The update of particle positions can be performed in two different ways:

$$\boldsymbol{v}_p^{t+\Delta t} = \tilde{\boldsymbol{v}}_p^{t+\Delta t} \tag{2.10}$$

$$\boldsymbol{v}_p^{t+\Delta t} = \boldsymbol{v}_p^t + \frac{\Delta t}{m_p} \boldsymbol{f}_p^t \tag{2.11}$$

The first option is known as the *pure PIC* update, which stems from the fact that the predecessor of MPM, the Particle-In-Cell Method, used this velocity update. This update leads to very stable simulation at the price of strong dissipation. The second option is known as a *pure FLIP* update, which was introduced with the Fluid Implicit Particle method **cite this.**. Pure FLIP is nearly free from dissipation, but introduces numerical noise into the simulation. `SMD` offers a combined PIC/FLIP update according to the rule:

$$\boldsymbol{v}_p^{t+\Delta t} = (1-\beta)\, \tilde{\boldsymbol{v}}_p^{t+\Delta t} + \beta \left(\boldsymbol{v}_p^t + \frac{\Delta t}{m_p} \boldsymbol{f}_p^t \right)$$

The parameter β defaults to 0.99 and can be tuned with the `FLIP` keyword of the MPM pair style.

2.1.7 Interpolation of updated particle velocities to the grid

The new particle velocities are interpolated again to the grid. Note that the weights $W(\boldsymbol{x}_{pn}^{t+\Delta t})$ are different from the old weights because the particles have moved with respect to the grid in the meantime.

$$m_n^{t+\Delta t} = \textstyle\sum_p W_{pn}^{t+\Delta t} m_p \tag{2.12}$$

$$\boldsymbol{v}_n^{t+\Delta t} = \tfrac{1}{m_n^{t+\Delta t}} \textstyle\sum_p W_{pn}^{t+\Delta t} m_p \boldsymbol{v}_n^{t+\Delta t} \tag{2.13}$$

2.1.8 Compute velocity gradient

The velocity gradient at the particle locations is computed according to:

$$\boldsymbol{L}_p^{t+\Delta t} = \sum_n \boldsymbol{v}_n^{t+\Delta t} \otimes \boldsymbol{\nabla} W_{pn}^{t+\Delta t}$$

2.1.9 Update particle stress and strain state

The update of strains and stresses starts with the update of the deformation gradient.

$$\boldsymbol{F}_p^{t+\Delta t} = \exp\left(\Delta t\, \boldsymbol{L}_p^{t+\Delta t}\right) \boldsymbol{F}_p^t$$

From the updated deformation gradient, the updated volume is calculated:

$$V_p^{t+\Delta t} = J\, V_p^0$$

Here, J is the determinant of $\boldsymbol{F}_p^{t+\Delta t}$ and V_p^0 is the initial particle volume.

`SMD` employs an updated stress formulation in which the stress is accumulated from the strain increments. The strain increment is given by

$$\boldsymbol{\epsilon}_p^{inc} = \Delta t\, \frac{1}{2}\left(\boldsymbol{L}^T + \boldsymbol{L}\right)$$

Note that the time superscript in the above line is omitted to reduce clutter. It is implicitely assumed that we refer to time $t + \Delta t$. Using the strain increment, a new value of the stress is computed via a constitutive law f:

$$\boldsymbol{\sigma}_p^{t+\Delta t} \leftarrow f\left[\boldsymbol{\sigma}_p^t, \boldsymbol{\epsilon}_p^{inc}\right]$$

To illustrate the very general concept of what such a constitutive law could look like, we consider a linear elastic solid with Lamë parameters λ and μ:

$$\boldsymbol{\sigma}_p^{t+\Delta t} = \boldsymbol{\sigma}_p^t + \lambda\, \mathrm{Tr}\left[\boldsymbol{\epsilon}_p^{inc}\right] + 2\,\mu\, \boldsymbol{\epsilon}_p^{inc}$$

Here, Tr [] denotes the trace operator.

3

Installation instructions

3.1 Obtaining the code

SMOOTH MACH DYNAMICS can be downloaded from Github.

https://github.com/ganzenmg/lammps_current/releases.

Download the newest release, e.g., Feb172016.tar.gz. In the following, this user guide assumes that work in a Linux shell. Extract the archive in any convenient folder, e.g. like this:

```
shell $> tar -xzf Feb172016.tar.gz
```

You will end up with a folder named "lammps-current-Feb172016" or similar.

3.2 Building the code

For compilation, a standard recent Linux installation should suffice (Ubuntu 14.04 has been tested ok). Change to the lammps-current-Feb172016/src directory and issue the following commands:

```
shell $> make stubs
shell $> make yes-user-smd
shell $> make serial
```

This will create the executable lmp_serial with all USER-SMD capabilities enabled. Additional executables for multiprocessor systems may be generated using the mpi or similar make targets, however, such a build setup is outside the scope of this document. See http://lammps.sandia.gov/doc/Section_start.ht for more info.

4

Simulation examples

4.1 MPM examples

4.1.1 Elastic collision of two discs

4.1.2 Funnel flow

4.2 SPH examples

4.2.1 Tensile loading of a rubber strip

. This example serves to illustrate the basic features of Total-Lagrangian SPH. A 2d strip of an elastic material is elongated by pulling two opposite edges apart at a quasi-static velocity. Due to Poisson's effect, the material contracts and a non-homogeneous distribution of stress is established. The following input script (located within the `examples/USER/smd/rubber_strip_pull`-directory) creates an initial geometry, applies boundary conditions, defines output quantities and invokes the required pair style and and time integration fixes.

The script starts with the definition of material parameters. This includes the Young's modulus, Poisson ratio, and mass density. Additionally, viscosity and hourglass control coefficients are defined her. Following this, the simulation is initialized with the `atom_style smd`, which provides the neccesary data structures for SPH.

The initial geometry of the rubber strip is defined as a quadratic lattice with spacing 1 mm. The SPH kernel diameter is set to three times this value, such that approximately 20 neighbours interact with each particle. Note that the argument of the `set diameter` command is really the radius of the SPH smmothing kernel, and not the diameter as one might be tempted to think. The use of the "si" unit system does not imply that physical units of kilogrammes, meters, and second need to be employed. Rather, any physical system of units which is *consistent*, may be used. Here, we opt for GPa, mm, and ms. A velocity boundary condition is implemented by creating individual groups for the top and bottom rows of particles. These groups of particles are subsequently pulled apart via the `fix smd/setvel` command to effect tensile loading.

The material is modelled using the Total-Lagrangian pair style `smd/tlsph`. The `*COMMON` keyword is mandatory and defines quantities which are not related to a specific material model. This also include Young's modulus and Poisson ratio, which are infinitesimal strain quantities and thus indeed independent from the functional form of the material model. The parameters `Q1` and `Q2` define the coefficients for the linear and quadratic part of the artificial viscosity. In most cases, the values 0.06 and 0, respectively, provide good results. The parameter `hg` is the dimensionless hourglass control coefficient, which should be chosen in the range of 10 ... 100. Finally, the parameter `Cp` defines the specific heat capacity, which is used by some material models to calculate a temperature from the particle's internal energy. Here, its value does not matter. The second and third keywords activate a linear material model, i.e., Hookean linear elasticity, both for deviatoric (shear) and dilational (volumetric) deformation. The required parameters are taken from the `*COMMON` keyword section. Note that the pressure relation can be chosen completely independent from the material strength model, i.e., the `USER-SMD` code performs a decomposition of the material behaviour into the equation of state (*EOS...) and the shear response (*STRENGTH...) model.

The Cauchy stress and the number of interacting neighbours for each particle are obtained using `compute` commands and written to the dump file. The engineering strain and stress, i.e., the global measures for these quantities are calculated as variables and written to a separate file using the `fix print` command.

Time integration of the system is performed via the `fix smd/integrate_tlsph` command. In the given form with no arguments, the reference configuration for the Total-Lagrangian is fixed at the initial coordinates of time-step zero and never changed during the course of the simulation. At every time-step, a stable time increment for the explicit Velocity-Verlet integration approach used by `LAMMPS` is computed by the `fix smd/adjust_dt` command.

Input file for tensile simulation

```
#########################################################################################################
#
# TLSPH example:  elongate a 2d strip of a linear elastic material py pulling its ends apart
#
# unit sytem: GPa / mm / ms
#
#########################################################################################################
```

```
##########################################################################################################
# MATERIAL PARAMETERS
##########################################################################################################
variable        E equal 1.0 # Young's modulus
variable        nu equal 0.3 # Poisson ratio
variable        rho equal 1 # initial mass density
variable        q1 equal 0.06 # standard artificial viscosity linear coefficient
variable        q2 equal 0.0  # standard artificial viscosity quadratic coefficient
variable        hg equal 10.0 # hourglass control coefficient
variable        cp equal 1.0 # heat capacity of material -- not used here

##########################################################################################################
# INITIALIZE LAMMPS
##########################################################################################################
dimension       2
units           si
boundary        sm sm p # simulation box boundaries
atom_style      smd
atom_modify     map array
comm_modify     vel yes
neigh_modify    every 10 delay 0 check yes # re-build neighbor list every 10 steps
newton          off

##########################################################################################################
# CREATE INITIAL GEOMETRY
##########################################################################################################
variable        l0 equal 1.0 # lattice spacing for creating particles
lattice         sq ${l0}
region          box block -10 10 -10 10 -0.1 0.1 units box
create_box      1 box
create_atoms    1 box
group           tlsph type 1


##########################################################################################################
# DISCRETIZATION PARAMETERS
##########################################################################################################
variable        h equal 2.01*${l0} # SPH smoothing kernel radius
variable        vol_one equal ${l0}^2 # volume of one particle -- assuming unit thickness
variable        skin equal ${h} # Verlet list range
neighbor        ${skin} bin
set             group all volume ${vol_one}
set             group all smd_mass_density ${rho}
set             group all diameter ${h} # set SPH kernel radius

##########################################################################################################
# DEFINE VELOCITY BOUNDARY CONDITIONS
##########################################################################################################
variable        vel0 equal 0.005 # pull velocity
region          top block EDGE EDGE 9.0 EDGE EDGE EDGE units box
region          bot block EDGE EDGE EDGE -9.1 EDGE EDGE units box
group           top region top
group           bot region bot
variable        vel_up equal ${vel0}*(1.0-exp(-0.01*time))
variable        vel_down equal -v_vel_up
fix             veltop_fix top smd/setvelocity 0 v_vel_up 0
fix             velbot_fix bot smd/setvelocity 0 v_vel_down 0

##########################################################################################################
# INTERACTION PHYSICS / MATERIAL MODEL
##########################################################################################################
pair_style      smd/tlsph
pair_coeff      1 1 *COMMON ${rho} ${E} ${nu} ${q1} ${q2} ${hg} ${cp} &
                *STRENGTH_LINEAR &
                *EOS_LINEAR &
                *END

##########################################################################################################
# TIME INTEGRATION
##########################################################################################################
fix             dtfix tlsph smd/adjust_dt 0.1 # dynamically adjust time increment every step
fix             integration_fix tlsph smd/integrate_tlsph
```

```
##########################################################################################################
# SPECIFY TRAJECTORY OUTPUT
##########################################################################################################
compute         S all smd/tlsph_stress # Cauchy stress tensor
compute         E all smd/tlsph_strain # Green-Lagrange strain tensor
compute         nn all smd/tlsph_num_neighs # number of neighbors for each particle
dump            dump_id all custom 10 dump.LAMMPS id type x y z vx vy vz &
                        c_S[1] c_S[2] c_S[4] c_nn &
                        c_E[1] c_E[2] c_E[4] &
                        vx vy vz
dump_modify     dump_id first yes

##########################################################################################################
# STATUS OUTPUT
##########################################################################################################
variable        stress equal 0.5*(f_velbot_fix[2]-f_veltop_fix[2])/20 # stress = force / initial width
variable        length equal xcm(top,y)-xcm(bot,y)
variable        strain equal (v_length-${length})/${length} # engineering strain
fix             stress_curve all print 10 "${strain} ${stress}" file stress_strain.dat screen no

thermo          100
thermo_style    custom step dt f_dtfix v_strain

##########################################################################################################
# RUN SIMULATION
##########################################################################################################
run             2500
```

From within the `examples/USER/smd/rubber_strip_pull`-directory, the simulation should be executed with the following shell command:

```
shell $> PATH-TO-LAMMPS-EXECUTABLES/lmp_serial < rubber_strip_pull.lmp
```

Figure (4.1) shows the final state of the simulation at an achieved engineering strain of ≈ 0.15. The visualisation is obtained using the program `OVITO` [**?**], which is recommended due to its ease of use and the speed at which large visualisations of simulations can be rendered. A check for the accuracy of the simulation is reported in Figure (4.2), which displays the observed stress-strain relationship. Note that the effective Young's modulus under 2d plane-strain conditions is given by

$$E_{2d} = \frac{E}{1-\nu^2} \approx 1.1\,\text{GPa}.$$

This value is indeed well reproduced by the slope of the stress-strain relation, at least for small strains. For larger strains, deviations are expected because the engineering stress output by the simulation becomes invalid.

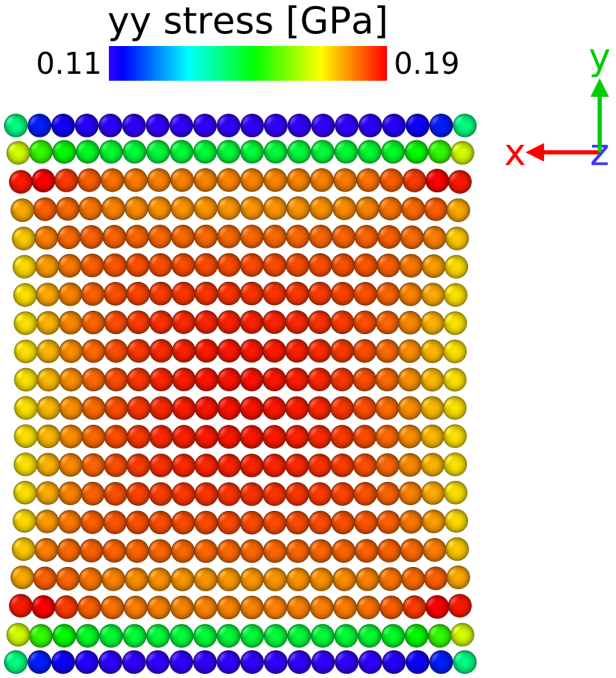

Figure 4.1: Rubber strip pull simulation. The colour coding shows the final state of the yy-stress distribution (units: GPa).

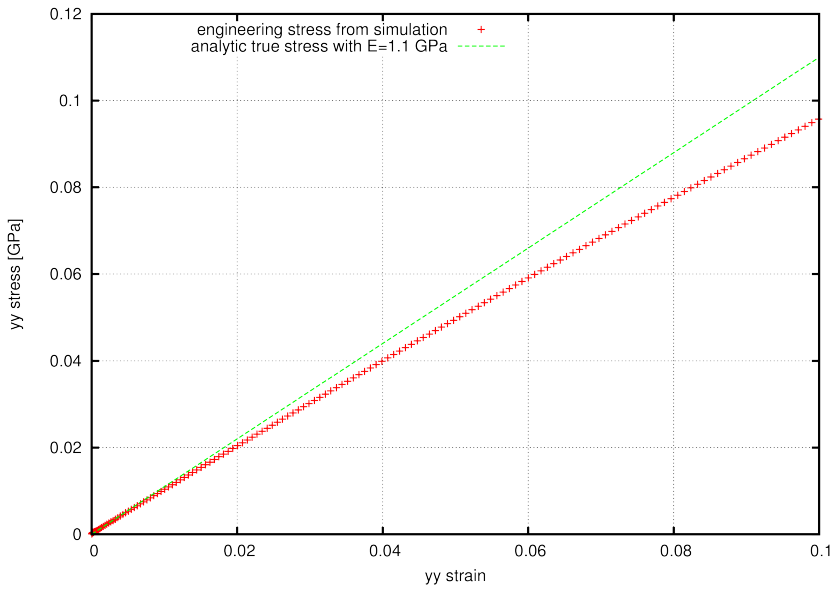

Figure 4.2: Rubber strip pull simulation. The graph shows the simulation data (symbols) and an analytic stress-strain relation.

4.2.2 Tensile loading of an aluminum strip

. This example illustrates modelling of real materials, in this case aluminum, with a complicated material model including failure.

4.2.3 Fluid-Structure interaction

. This example illustrates how a fluid and deformable solids can be mutually coupled.

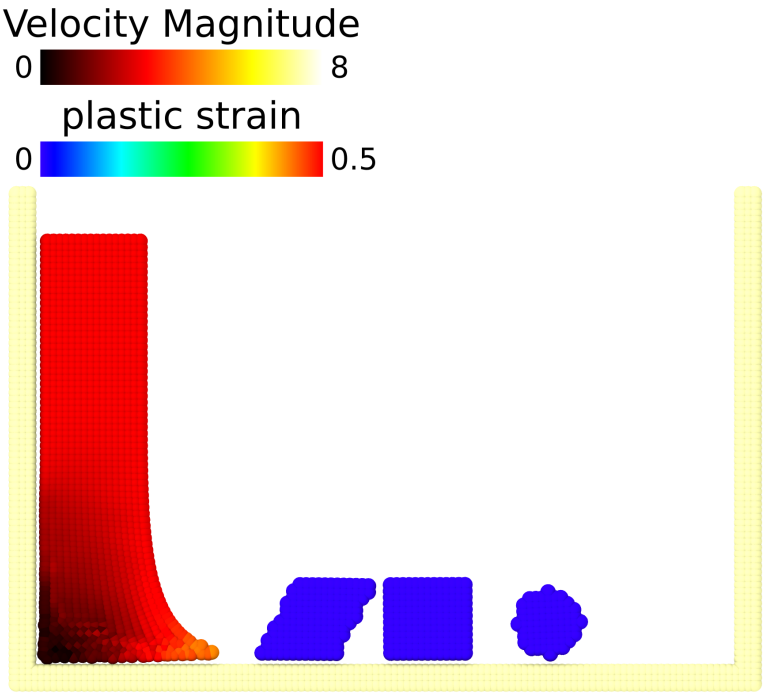

Figure 4.3: Fluid-Structure interaction snapshot 1

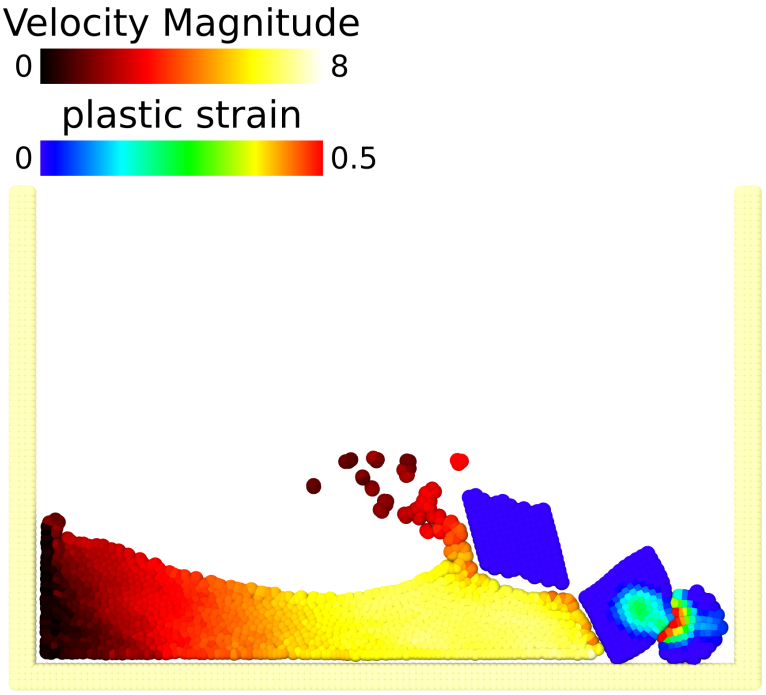

Figure 4.4: Fluid-Structure interaction snapshot 2

4.2.4 Funnel flow

.

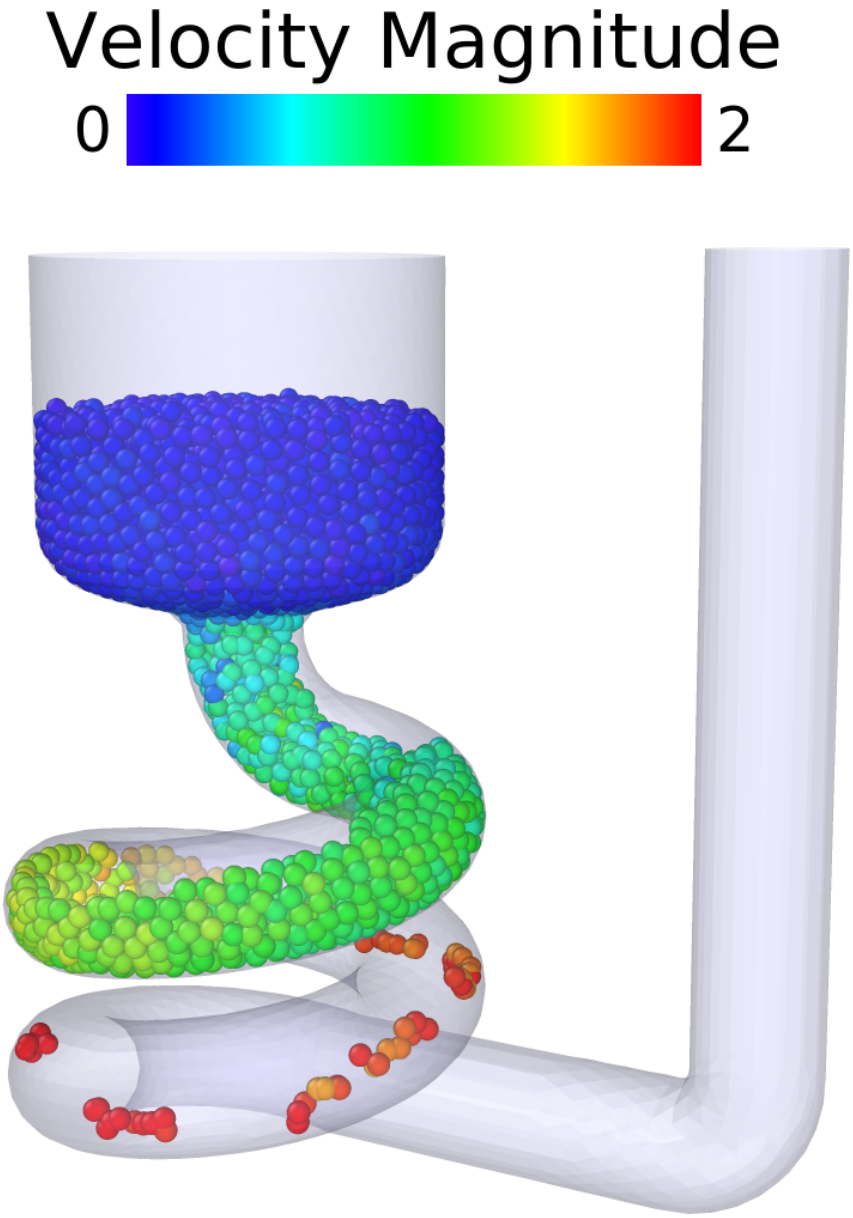

Figure 4.5: Gravity-driven fluid flow in a complex geometry.

5

Creating particles

5.1 The USER-SMD data structure

It is mandatory to use the `atom_vec smd` command for SPH particles, which allocates memory for the basic data which are associated with an SPH particle. These data are:

variable	description	value upon initialisation
`int tag`	atom id	1 ... N
`int type`	atom type	1 ... ntypes
`int mol`	re-use of molecule to group collections of particles,	1
`float x[3]`	particle coordinates	position
`float x0[3]`	reference particle coordinates	same as `x`
`float v[3]`	velocity used for Verlet time integration algorithm, correct only at half-time-steps	zero vector
`float vest[3]`	extrapolated velocity at full time-steps	zero vector
`float vfrac`	particle volume	1.0
`float rmass`	particle mass	1.0
`float radius`	SPH kernel radius	1.0
`float contact_radius`	contact radius	1.0
`float e`	internal energy	0.0
`float tlsph_fold[9]`	deformation gradient of a particle in the reference configuration	unit matrix
`float tlsph_stress[6]`	unrotated Cauchy stress	zero matrix
`float eff_plastic_strain`	effective plastic strain	0.0
`float eff_plastic_strain_rate`	effective plastic strain rate	0.0
`float damage`	damage status of a particle	0.0

5.2 Creating geometries and reading USER-SMD data files

Two different methods exist for creating SPH particles. From within the `LAMMPS` command file, simple geometries can be defined using `create_atoms` command in combination with `lattice`, `region` commands.

Alternatively, arbitrarily complex geometries may be read from disk via the `read_data` command. The format of a single line in the `Atoms` section, which defines the initial properties for an SPH particle, is as follows:

```
tag type mol vfrac rmass radius contact_radius x y z x0 y0 z0
```

5.3 Initialising particle properties from the LAMMPS command file

Particle properties can be initialised using the LAMMPS `set` command. The general syntax for the `set` is:

```
set style ID keyword values ...
-------------------------------
style = atom or type or mol or group or region
ID = atom ID range or type range or mol ID range or group ID or region ID
```

For a complete description of the `set` command, see the LAMMPS documentation. Here, only the `keywords` specific to the `USER-SMD` package are detailed. Note that the setup of a SPH simulation requires the proper setting of the variables mass, volume, mass density, kernel radius, contact radius, and internal energy. If a LAMMPS data file is read, the variables volume, mass, kernel radius, and contact radius are taken from that file.

5.3.1 Particle volume

The volume of a particle is set by:

```
set style ID volume v
```

Note that the sum of all particles' volumes should equal the volume of the body which is to be simulated. This can be checked using the output of `compute smd/vol`. Also note that the quantities mass, volume, and mass density should be set mutually consistent.

5.3.2 Particle mass

The mass of a particle can be initialised directly:

```
set style ID mass m
```

Here, `m` is the particle's mass. Assuming that the particle's volume V has already been correctly set, the mass can also be initialized by specifying a mass density:

```
set style ID smd/mass_density rho
```

This command computes the particle mass using $m = \rho \times V$, where ρ is the specified mass density.

5.3.3 Particle radius

There are two different radii values associated with each particle. The `diameter` equals the SPH kernel range. The `contact_radius` is the radius used for computing contact forces which prevent independent bodies from penetrating. Typically, the contact radius corresponds to one half of the particle spacing, whereas the `diameter` is approximately 3 times the particle spacing. These quantities are set using:

```
set style ID smd/contact_radius value
set style ID diameter value
```

Note that the argument of the `set diameter` command is the radius of the SPH smoothing kernel, and not the diameter as one might be tempted to think.

5.3.4 Particle heat energy

The heat energy of a particle is set by:

```
set style ID smd_heat value
```

Note that you can set a particle's temperature T this way via the relation

$$q = T\, c_p\, m, \tag{5.1}$$

Where q is heat energy, c_p is heat capacity (defined in file `materials.ini`), and m is the particle's mass.

6

SPH pair styles

Two different SPH pair styles are implemented in USER-SMD. The pair style smd/tlsph utilises a total Lagrangian formulation which relates the deformation to particle displacements relative to a fixed reference configuration. The pair style smd/ulsph is an updated Lagrangian formulation which does away with the reference configuration and computes all deformations from time integration of the velocities. The total Lagrangian formulation is more apt to the simulation of solid bodies due to better accuracy, while the updated Lagrangian formulation is better suited to fluid flow problems as arbitrarily large deformations are more easily described. Both pair styles are activated with the usual LAMMPS commands pair_style and pair_coeff, however, due to the large number of possible parameters, the concept of keyword cards, similar to the input decks of established Finite Element solver packages is used.

6.1 The total Lagrangian pair style

This pair style is invoked with the following command:

```
pair_style smd/tlsph
pair_coeff <i> <j> *COMMON <rho0> <E> <nu> <Q1> <Q2> <hg> <Cp> &
                   *END
```

Here, i and j denote the LAMMPS particle types for which this pair style is defined. Note that i and j must be equal, i.e., no smd/tlsph cross interactions between different particle types are allowed.. In contrast to the usual LAMMPS pair coeff definitions, which are given solely a number of floats and integers, the smd/tlsph pair coeff definition is organised using keywords. These keywords mark the beginning of different sets of parameters for particle properties, material constitutive models, and damage models. The pair coeff line must be terminated with the *END keyword. The use the line continuation operator & is recommended. A typical invocation of the smd/tlsph for a solid body would consist of an equation of state for computing the pressure (the diagonal components of the stress tensor), and a material model to compute shear stresses (the off-diagonal components of the stress tensor). Damage and failure models can also be added.
The *COMMON keyword is mandatory.

```
*COMMON <rho0> <E> <nu> <Q1> <Q2> <hg> <Cp>
```

This keyword must be followed by 7 numbers:

parameter	meaning	recommended value
rho0	reference density ρ_0	initial mass density
E	reference Young's modulus E	initial Young's modulus
nu	reference Poisson ratio ν	initial Poisson ratio
Q1	linear artificial viscosity coefficient $Q1$	0.05 to 1.0
Q2	not used	
hg	hourglass control coefficient γ	1 to 20
Cp	specific heat capacity C_p, units: energy / (mass * temperature)	must be > 0

From these parameters, some useful quantities are pre-computed, which are accessed by the various equations of state and material models:

parameter	description
bulk modulus	$K = \frac{E}{3(1-2\nu)}$
Lamé parameter λ	$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$
shear modulus G	$G = \frac{E}{2(1+\nu)}$
p-wave speed modulus M	$M = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$

6.2 The updated Lagrangian pair style

This pair style is invoked with the following command:

```
pair_style smd/ulsph <*DENSITY_SUMMATION   or *DENSITY_CONTINUITY> &
                     <*VELOCITY_GRADIENT   or *NO_VELOCITY_GRADIENT> &
                     <*GRADIENT_CORRECTION or *NO_GRADIENT_CORRECTION>
pair_coeff <i> <j> *COMMON <rho0> <c0> <Q1> <Cp> <Hg> &
                   *END
```

Following `smd_ulsph`, three keywords are expected:

keyword	description
*DENSITY_SUMMATION	Recompute the mass density every timestep from scratch.
*DENSITY_CONTINUITY	Time-integrate the mass density using the rate-of deformation tensor.
*VELOCITY_GRADIENT	Enable computation of the velocity gradient. This is needed for *DENSITY_CONTINUITY, viscosity constitutive models, and strength constitutive models.
*NO_VELOCITY_GRADIENT	Save computation time by not computing the velocity gradient. This is only useful for pure fluid simulations.
*GRADIENT_CORRECTION	Compute first-order corrected derivatives. This makes the SPH scheme more accurate but also susceptible to numerical instabilities.
*NO_GRADIENT_CORRECTION	Do not compute first-order corrected derivatives

i and j denote the LAMMPS particle types for which this pair style is defined. Note that i and j can be different, i.e., smd/ulsph cross interactions between different particle types are allowed. In contrast to the usual LAMMPS pair coeff definitions, which are given solely a number of floats and integers,

the smd/ulsph pair coeff definition is organised using keywords. These keywords mark the beginning of different sets of parameters for particle properties, material constitutive models, and damage models. The pair coeff line must be terminated with the *END keyword. The use the line continuation operator & is recommended. A typical invocation of the smd/ulsph for a solid body would consist of the *COMMON keyword and an equation of state for computing the pressure (the diagonal components of the stress tensor).
The *COMMON keyword is mandatory.

```
*COMMON <rho0> <c0> <Q1> <Cp> <Hg>
```

This keyword must be followed by 5 numbers:

parameter	meaning	recommended value
rho0	reference density ρ_0	initial mass density
c0	reference speed of sound $c0$	
Q1	linear artificial viscosity coefficient $Q1$	0.05 to 1.0
Cp	specific heat capacity C_p, units: energy / (mass * temperature)	must be > 0
Hg	hourglass control coefficient	0

From these parameters, some useful quantities are pre-computed, which are accessed by the various equations of state and material models:

parameter	description
bulk modulus	$K = c_0^2 \rho_0$

If a cross interaction between different particle types is required, i.e., between a highly viscous fluid and a low viscosity fluid, the *CROSS keyword can be specified, but only after the i – i and j – j interactions have been defined as shown in the example below:

```
pair_style smd/ulsph *DENSITY_CONTINUITY *VELOCITY_GRADIENT *GRADIENT_CORRECTION
pair_coeff 1 1 *COMMON <rho0> <c0> <Q1> <Cp> <Hg>&
               *EOS_LINEAR &
               *END
pair_coeff 2 2 *COMMON <rho0> <c0> <Q1> <Cp> <Hg> &
               *EOS_LINEAR &
               *VISCOSITY_NEWTON <mu> &
               *END
pair_coeff 1 2 *CROSS
```

Note: the kernel gradient correction can lead to severe instabilities if the neighborhood of a particle is ill defined (too few neighbors or coplanar / colinear arrangements). The reason for this is that the shape matrix needs to be inverted, which becomes impossible if it is rank deficient. This problem is minimized due to the use of a stable SVD algorithm for the inversion. Nevertheless, if your updated Lagrangian simulation crashes, try using *NO_GRADIENT_CORRECTION.

7

The Material Point Method Pair Style

This pair style is invoked with the following command:

```
pair_style smd/mpm_linear <cellsize> <optional args>
pair_coeff <i> <j>
```

Here, `i` and `j` denote the LAMMPS particle types for which this pair style is defined. Note that cross interactions between particle types `i` and `j` are allowed if an `i–i` and `j–j` interaction with this pair style are also defined. Note that you explicitely need to give `i` and `j` in numbers: A typical LAMMPS shortcut like `* *` is not allowed. `cellsize` denotes the cell spacing of the regular background grid which is used as asn auxialiary scratchpad to compute the stress and strain fields.
The following optional arguments may be appended to the `pair_style smd/mpm_linear` command

- `limit_velocity <float value>`
  This will limit the maximum velcoity of any particle interacting according to the MPM pair style to `value`.
- `FLIP <float value>`
  This command sets the ratio of PIC/FLIP update of the velocities. `value` must be in the range of 0(pure PIC) to 1 (pure FLIP). Note that PIC is dissipative, but a little PIC is often needed to stabilize the simulation. The default is 0.99 FLIP update and 0.01 PIC update.
- `2d`
  If a 2d simulation is performed, zero out velocities and accelerations in z-direction.
- `exclude_region <string region_name> <float vx> <float vy> <float vz>` Particles contained in the specified region will have zero acceleration and move at the specified constant velocity.
- `corotated`
  Employ a corotated formulation for the stress update. Here, all rigid body rotations are subtracted from the velocity gradient. Stress updates are performed in the unrotated frame of reference, which makes the stress rate objective. The updated stresses are subsequently rotated forward to the current configuration. The corotated stress formulation is more accuarate than the default Jaumann update, but increases simulation time by roughly 30
- `true_deformation`
  Use the real velocity field, i.e., the time derivative of the coordinates, for the computation of the velocity gradient. Per default, a different velocity field is used in MPM, which is similar, but not identical to the real velocity field. Fur pure fluid simulation, this command will provide better volume conservation (incompressibility). For solid bodies, the default MPM approach works better.

- `sym_y_+ <float location>`
  Create a symmetry plane at y-coordinate `location`. Particles may only exist above the symmetry plane, as indicated by the + sign. Completely analogous symmetry planes may be specified with `sym_x_+` and `sym_z_+`. Each of the `sym_x_+`, and `sym_y_+`, and `sym_z_+` commands may only appear once.

- `sym_y_- <float location>`
  Create a symmetry plane at y-coordinate `location`. Particles may only exist below the symmetry plane, as indicated by the − sign. Completely analogous symmetry planes may be specified with `sym_x_-` and `sym_z_-`. Each of the `sym_x_-`, and `sym_y_-`, and `sym_z_-` commands may only appear once.

- `thermal`
  Enable heat conduction between all particles which interact via the MPM pair style. . This is the transport equation which is solved:

$$\dot{q} = \alpha \nabla^2 q \tag{7.1}$$

  Here, q is heat (units of energy), and α is the thermal diffusion constant with units of length^2/time. α is calculated from the heat conduction coefficients, which are definined in file `materials.ini` Note that the thermal diffusion constant is related to the thermal conductivity κ, which has units of energy / (time × length × temperature)

$$\alpha = \frac{\kappa}{\rho\, c_p} \tag{7.2}$$

Note that no material behavior has yet been specified. For this purpose, the file `materials.ini` is parsed. This file *must* provide sections for *all* particle types, regardless whether they interact via the MPM pair style or not.

8

MPM Material Models

Material models for in USER-SMD are decomposed into isotropic and deviatoric parts, corresponding to volumetric and shear deformations. The relationship between the scalar quantities volume change $\mu = \rho/\rho_0 - 1$ and pressure p is given by an *equation of state*, while the relation between a tensorial shear deformation $\boldsymbol{\epsilon}_d$ and the stress deviator tensor $\boldsymbol{\sigma}_d$ is given by a *material strength* model. The decomposition is additive, i.e.,

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\sigma}_d,$$

where $\boldsymbol{I}$ is the diagonal unit tensor.

Material models are read from the file `materials.ini` which must reside in the same directory as the input script. This file must provide a *general section* for each particle type, regardless whether it interacts via the MPM pair style or not. Each general section must have a section header given by the particle type number in square brackets.

This are the keywords of the general section. Note that SI units are indicated for clarification purposes, however, you are free to use any consistent systems of units you deem neccessary.

keyword	description
`rho0 = <float>`	reference mass density [kg / m^3]
`cp = <float>`	heat capacity [J/(kg K)]
`heat_conduction = <float>`	heat conduction coefficient [W/(m K)]
`EOS = <string>`	section name defining equation of state. Must be defined.
`STRENGTH = <string>`	section name defining strength model. Can be `NONE`
`VISCOSITY = <string>`	section name defining viscosity mode. Can be `NONE`

Each section name provided with the `EOS`, `STRENGTH`, or `VISC` keywords must exist, with the exception of `NONE`, in which case no strength model or no viscosity model is activated for that particle type.

For illustrative purposes, the following `material.ini` file defines a linear fluid for particle type 1, while particle type 2 is an elastic solid. Note that comments start with a semicolon (`;`).

```
; this simulation has 2 particle types, so we need two general sections, [1] and [2]

[1]
rho0 = 1000             ; mass density 1000 kg/m^3
cp = 4200               ; heat capacity, 4200 J/(kg K)
heat_conduction = 0.56  ; heat_conduction, 0.56 W/(m K)
EOS = Water_EOS         ; Water_EOS is a section name which is located further below.
STRENGTH = NONE         ; This is a fluid so no strength model is required.
VISCOSITY = NONE        ; No viscosity is used here.

[2]
rho0 = 2700             ; mass density 2700 kg/m^3
cp = 900                ; heat capacity, 900 J/(kg K)
heat_conduction = 204   ; heat_conduction, 204 W/(m K)
EOS = Alu_EOS           ; Alu_EOS is a section name which is located further below.
STRENGTH = Alu_Strength ; dito for Alu_Strength
VISCOSITY = NONE        ; No viscosity is used here.

[Water_EOS]
EosId = 1               ; 1 invokes Linear EOS
K = 2.0e9               ; bulk modulus of 2 GPa

[Alu_EOS]
EosId = 1               ; 1 invokes Linear EOS
K = 33.0e9              ; bulk modulus of 33 GPa

[Alu_Strength]
MatId = 1               ; 1 invokes Linear strength model
nu = 0.3                ; Poisson ratio
E = 70.0e9              ; Young's modulus of 70 GPa
```

8.1 Equation of State models

Note that J denotes the determinant of the deformation gradient.

8.1.1 Linear Equation of State

This EOS `#1`. It is defined with the following keywords:

keyword	description
`EosId = 1`	mandatory
`K = <float>`	bulk modulus

Pressure is computed according to the following relation:

$$\begin{aligned} \mu &= J-1 \\ p &= K\mu \end{aligned}$$

8.1.2 Murnaghan-Tait isothermal Equation of State

This EOS `#2`. It is defined with the following keywords:

keyword	description
`EosId = 2`	mandatory
`K = <float>`	bulk modulus
`n = <int>`	Tait exponent

$$p = \frac{K}{n}\left(J^n-1\right)$$

8.2 Material strength models

8.2.1 Linear strength model

This material `#1`. It is defined with the following keywords:

keyword	description
`MatId = 1`	mandatory
`E = <float>`	Young's modulus
`nu = <float>`	Poisson ratio

Deviatoric stresses are computed according to the following relation:

$$\boldsymbol{\sigma}_d = 2G\boldsymbol{\epsilon}_d,$$

where G is the shear modulus, which is computed from the provided parameters. In combination with EOS `#1`, Hookean linear elasticity is obatined.

8.2.2 Simple plasticity model

This material `#2`. It is defined with the following keywords:

keyword	description
`MatId = 2`	mandatory
`E = <float>`	Young's modulus
`nu = <float>`	Poisson ratio
`yield_stress = <float>`	Initial yield stress

This strength model implements the deviatoric stress part of linear elastic / plastic material behaviour. This is a history-dependent strength model. At each time-step n, an elastic trial update to the stress deviator is performed>,

$$\boldsymbol{\sigma}_d^{trial} = \boldsymbol{\sigma}_d^n + 2G\boldsymbol{d}_d,$$

where G is the shear modulus, and $\boldsymbol{d}_d$ is the deviatoric part of the strain rate tensor. The second invariant J_2 of the trial stress deviator is then compared to the current plastic yield stress, which is a linear function of the hardening parameter H and the equivalent plastic strain $\varepsilon_{plastic}^{eqv.}$.

$$\sigma_{yield} = \sigma_{yield}^0 + H\,\varepsilon_{plastic}^{eqv.}$$

If J_2 is below the yield stress, the elastic update is accepted. Otherwise, a limiting stress deviator with $J_2 = \sigma_{yield}$ is obtained by scaling the trial stress deviator using the radial return algorithm [27]. The increase in plastic strain, i.e., the amplitude by which the trial stress deviator has to be scaled back such that $J_2 = \sigma_{yield}$ is added to the `effective_plastic_strain` variable, which can be accessed via the `compute smd/plastic_strain` command.
Note: Documentation is incorrect here, some scalar factors are missing.
Note: Hardening is not implemented as of now (3 Feb 2016).

9

Contact pair styles

USER-SMD uses contact forces to prevent different physical entities, such as individual solid bodies or a fluid phase, from penetrating each other.

9.1 Hertzian particle contact force

This pair style is invoked with the following command:

```
pair_style smd/hertz scale_factor
pair_coeff <i> <j> contact_stiffness
```

Here, `i` and `j` denote the LAMMPS particle types for which this pair style is defined. Note that this contact force can be defined both between different particle types and for the same particle type. The latter is useful to model self-contact, e.g., when a flexible part bends and starts to interact with itself. The Hertzian potential between two particles with contact radii R_i and R_j and mutual distance r is defined as follows:

$$r_{cut} = R_i + R_j \tag{9.1}$$

$$\delta = r_{cut} - r \tag{9.2}$$

$$r_{geom} = \frac{R_i R_j}{r_{cut}} \tag{9.3}$$

$$f_{ij} = E\sqrt{\delta r_{geom}} \;\; \forall r < r_{cut} \tag{9.4}$$

These expressions are derived from the standard form of the Hertzian overlap potential between two elastic spheres with elastic modulus E. This elastic modulus is set via the argument `contact_stiffness`, which has units of pressure. A good choice for the contact stiffness is approximately one tenth to one half of the characteristic stiffness of the interacting particle types, i.e., the Young's modulus for a solid body or the bulk modulus for a liquid. Much larger values of the contact stiffness lead to instabilities with the standard magnitude of the CFL-stable time increment as computed by `fix smd/adjust_dt` and much smaller values allow for penetration. Note that the radii for this potential are derived from the contact radii, scaled with the argument `scale_factor`. It is recommended to set the scale factor to 1.5 if the contact radii are defined as one half of the initial distance between particles. This approach leads to a much smoother effective interaction surface, compared to `scale_factor`=1.0.

Note that in the case of self-contact, i.e., particle type `i` equals particle type `j`, an additional interaction check is performed: only particles which are separated by more than r_{cut} in the reference configuration are allowed to interact via the Hertzian potential. This check ensures that self contact forces may only appear between particles which cannot interact otherwise, e.g., via SPH.

9.2 Hertzian wall contact force for triangulated geometries

This pair style allows to use triangulated surfaces, read in from files in `.STL` format, as rigid walls. To use this feature, the following components must be activated:

1. Read a surface from file using the `smd_create/tri_boundary` command.
2. Define the `smd/tri_surface` pair style for interactions between the triangulated surface and other particles.
3. (optional) Define the `smd/move_tri_surf` fix to translate or rotate the triangulated surface during the course of a simulation.

9.2.1 Reading a surface

A surface is read in with the following command:

```
smd_create/tri_boundary file.stl type molID
```

Here, `file.stl` specifies the file holding the tesselated surface. This file has to adhere to the STL standard and must contain only a single surface. The triangles read in from this file are assigned a particle type `type` and a molecule tag `molID`, which has to be ≥ 65535. Note that multiple surfaces may be read in by invoking this command several times.

9.2.2 Triangulated surface pair style

This pair style causes particles which approach the surface to be repelled. Optionally, Coulomb friction or a tangential viscous interaction, as well as a temperature transfer from wall to particle may be specified.

```
pair_style smd/tri_surface scale_factor
pair_coeff <i> <j> contact_stiffness wall_temperature friction_coeff visc_coeff
```

Here, `i` and `j` denote the `LAMMPS` particle types for which this pair style is defined. One of these must correspond to the particly type given to the triangulated surface with the `smd_create/tri_boundary` command.
`contact_stiffness` influences the magnitude of the repulsive force which pushes a particle away from the wall. This force is given by a Hertzian-like potential, which is nonzero if the distance r between the particle and the closest point of a triangle is less than the particle's contact radius r_c.

$$\delta = r_c - r \tag{9.5}$$
$$\boldsymbol{f}_n = K\sqrt{\delta r_c}\,\boldsymbol{n} \tag{9.6}$$

K is the contact stiffnes which should be chosen roughly equal to the bulk modulus of the particle. $\boldsymbol{n}$ is the normal of the triangle.

A Coulomb friction force is computed according to

$$\boldsymbol{v}_{rel} = \boldsymbol{v}_{particle} - \boldsymbol{v}_{triangle} \tag{9.7}$$
$$\boldsymbol{v}_{tan} = \boldsymbol{v}_{rel} - (\boldsymbol{v}_{rel}\cdot\boldsymbol{n})\,\boldsymbol{n} \tag{9.8}$$
$$\boldsymbol{f}_{tan} = \mu|\boldsymbol{f}|\frac{\boldsymbol{v}_{tan}}{|\boldsymbol{v}_{tan}|} \tag{9.9}$$

Here, $\boldsymbol{v}_{tan}$ is the tangential velocity of the particle with respect to the triangle plane. μ is the dimensionless friction coefficient. The Coulomb friction force is proportional to the normal force.

If the particle is closer than its kernel radius to the triangle, a viscous friction force is computed.

$$\boldsymbol{f}_{tan} = \eta \, |\boldsymbol{v}_{tan}| \, \frac{\boldsymbol{v}_{tan}}{|\boldsymbol{v}_{tan}|} \tag{9.10}$$

Here, $\boldsymbol{v}_{tan}$ is the relative tangential velocity of the particle with respect to the triangle plane. η is the viscous friction coefficient with units of mass over time. The viscous friction force is proportional to the magnitude of the relative tangential velocity. Note that the kernel radius and the contact radius are two different things – see section 5.3.3. This allows to have a viscous force if a particle comes close to the wall, but before the frictional or repulsive contact becomes active.

If the particle is closer than its kernel radius to the triangle plane, its temperature is set to the specified value of `wall_temperature`, provided that this value is different from zero. This allows to approximate temperature transfer from the wall to the particles.

Note that the particles' contact radii may be scaled with a value of `scale_factor` different from unity.

9.2.3 Translating or rotating triangulated surfaces

This is work in progress.

10

SPH Material models

Material models for in USER-SMD are decomposed into isotropic and deviatoric parts, corresponding to volumetric and shear deformations. The relationship between the scalar quantities volume change $\mu = \rho/\rho_0 - 1$ and pressure p is given by an *equation of state*, while the relation between a tensorial shear deformation $\boldsymbol{\epsilon}_d$ and the stress deviator tensor $\boldsymbol{\sigma}_d$ is given by a *material strength* model. The decomposition is additive, i.e.,

$$\boldsymbol{\sigma} = p\boldsymbol{I} + \boldsymbol{\sigma}_d,$$

where $\boldsymbol{I}$ is the diagonal unit tensor.

10.1 Equation of state models

10.1.1 *EOS_LINEAR

The simplest EOS is activated by the keyword

```
*EOS_LINEAR
```

and computes pressure according to

$$\begin{aligned} \mu &= \frac{\rho}{\rho_0} - 1 \\ p &= K\mu \end{aligned}$$

This EOS is implemented both for `pair_style smd/ulsph` and `pair_style smd/tlsph`.

10.1.2 *EOS_SHOCK

This is a simple Hugoniot shock EOS. It is activated by the keyword

```
*EOS_SHOCK <c0> <S> <Gamma>
```

and computes pressure according to

$$\begin{aligned} \mu &= \frac{\rho}{\rho_0} - 1 \\ p_H &= \frac{\rho_0 c_0^2 \mu(1+\mu)}{(1.0 - (S-1.0) * \mu)^2} \\ p &= p_H + \rho * \Gamma * (e - e_0) \end{aligned}$$

where ρ is the mass density and e is the internal energy per unit mass. The subscript 0 refers to these values at the beginning of the simulation. This keyword must be followed by 3 numbers:

parameter	meaning
c0	reference speed of sound
S	Hugoniot parameter S, slope of u_s vs. u_p line
Gamma	Grueneisen parameter Γ

This EOS is implemented for `pair_style smd/tlsph`

10.1.3 *EOS_POLYNOMIAL

This a general polynomial expression for an EOS. It is activated by the keyword

```
*EOS_POLYNOMIAL <C0> <C1> <C2> <C3> <C4> <C5> <C6>
```

Pressure is computed according to

$$\begin{aligned} \mu &= \frac{\rho}{\rho_0} - 1 \\ p &= C_0 + C_1\mu + C_2\mu^2 + C_3\mu^3 + (C_4 + C_5\mu + C_6\mu^2)e \end{aligned}$$

where ρ is the mass density and e is the internal energy per unit mass. This keyword must be followed by 7 numbers:

parameter	meaning
C0	polynomial coefficient C_0
C1	polynomial coefficient C_1
C2	polynomial coefficient C_2
C3	polynomial coefficient C_3
C4	polynomial coefficient C_4
C5	polynomial coefficient C_5
C6	polynomial coefficient C_6

This EOS is implemented for `pair_style smd/tlsph`

10.1.4 *EOS_TAIT

This a general non-linear EOS which neglects thermal effects. It is activated by the keyword

```
*EOS_TAIT <n>
```

Pressure is computed according to

$$p = K\left[\left(\frac{\rho}{\rho_0}\right)^n - 1\right]$$

where ρ is the mass density and K is the bulk modulus, which is computed from the values passed with the `*COMMON` keyword. This keyword must be followed by 1 number:

parameter	meaning
n	Tait exponent n

Note that a typical value is $n = 7$ for hydraulic simulations of water.

This EOS is implemented for `pair_style smd/ulsph`

10.2 Material strength models

Material strength models are only defined for the `smd/tlsph` pair style.

10.2.1 *STRENGTH_LINEAR

The strength model implements the deviatoric stress part of linear, i.e., Hookean elasticity. It is activated by the following keyword:

```
*STRENGTH_LINEAR
```

The stress deviator is computed according to

$$\boldsymbol{\sigma}_d = 2G\boldsymbol{\epsilon}_d,$$

where G is the shear modulus, which is computed from the parameters provided with the `*COMMON` keyword.

This EOS is implemented both for `pair_style smd/ulsph` and `pair_style smd/tlsph`.

10.2.2 *STRENGTH_LINEAR_PLASTIC

The strength model implements the deviatoric stress part of linear elastic / ideal plastic material behaviour. It is activated by the following keyword:

```
*STRENGTH_LINEAR_PLASTIC <yield_stress0> <hardening parameter>
```

This is a history-dependent strength model. At each time-step n, an elastic trial update to the stress deviator is performed>,

$$\boldsymbol{\sigma}_d^{trial} = \boldsymbol{\sigma}_d^n + 2G\boldsymbol{d}_d,$$

where G is the shear modulus, and $\boldsymbol{d}_d$ is the deviatoric part of the strain rate tensor. The second invariant J_2 of the trial stress deviator is then compared to the current plastic yield stress, which is a linear function of the hardening parameter H and the equivalent plastic strain $\varepsilon_{plastic}^{eqv.}$.

$$\sigma_{yield} = \sigma_{yield}^0 + H\,\varepsilon_{plastic}^{eqv.}$$

If J_2 is below the yield stress, the elastic update is accepted. Otherwise, a limiting stress deviator with $J_2 = \sigma_{yield}$ is obtained by scaling the trial stress deviator using the radial return algorithm [27]. The increase in plastic strain, i.e., the amplitude by which the trial stress deviator has to be scaled back such that $J_2 = \sigma_{yield}$ is added to the `effective_plastic_strain` variable, which can be accessed via the `compute smd/plastic_strain` command.

This keyword must be followed by 2 numbers:

parameter	meaning
`yield_stress0`	plastic yield stress σ_{yield}^0
`hardening parameter`	hardening parameter H

This strength model is implemented both for `pair_style smd/ulsph` and `pair_style smd/tlsph`.

10.2.3 *STRENGTH_JOHNSON_COOK

This is a complex strength model developed for ductile metals. It is activated by the following keyword:

```
*STRENGTH_JOHNSON_COOK <A> <B> <a> <C> <epdot0> <T0> <Tmelt> <m>
```

This strength model performs linear trial updates of the shear stresses using the shear modulus G. The shear stresses are subsequently limited with a complex yield criterion which depends on the equivalent plastic strain ε_p, the equivalent plastic strain rate $\dot{\varepsilon}_p$ and temperature.

$$\sigma_{yield} = \left[A + B\,\varepsilon_p^a\right]\,\left[1 + C\,\ln\left(\frac{\dot{\varepsilon}_p}{\dot{\varepsilon}_{p,0}}\right)\right]\,\left[1 - T_H^m\right]$$

T_H is the homologeous temperature defined using current temperature T, reference temperature T_0 and melting temperature T_{melt} as:

$$T_H = \frac{T - T_0}{T_{melt} - T_0}$$

This keyword must be followed by 7 numbers:

parameter	meaning
A	initial yield stress A
B	prop. factor for plastic strain dependence B
a	exponent for plastic strain dependence a
C	prop. factor for plastic strain rate dependence C
epdot0	reference plastic strain rate $\dot{\varepsilon}_{p,0}$
T0	reference temperature T_0
Tmelt	melt temperature T_{melt}
m	exponent m for temperature dependence

This strength model is implemented for `pair_style smd/tlsph`

10.3 Damage and failure models

Damage and failure models are only defined for the `smd/tlsph` pair style. Classical continuum mechanics failure models work *at an integration point*, i.e., at a particle and typically degrade the stress that this integration point can bear according to some rule. An alternative way of introducing damage and failure, which is in much better agreement with the meshless spirit of SPH, is to degrade the interaction between specific particle pairs only according to some rule. This approach has been used as the visibility criterion before in other meshless methods such as the Element-Free Galerkin method [28] but is also extensively persued in meshless peridynamic implementations [29].
`SMOOTH MACH DYNAMICS` only features degradation of pairwise interactions, however, the degradation may be initiated using a classical damage/failure model.

10.3.1 *FAILURE_MAX_PLASTIC_STRAIN

This is a simple failure model based on equivalent plastic strain. It is activated by the following keyword:

```
*FAILURE_MAX_PLASTIC_STRAIN value
```

If the equivalent plastic strain of a particle exceeds the threshold `value`, its damage variable D is immediately set to unity. Degradation of the material is achieved by considering pairs $< i, j >$ of particles. As soon as an interacting particle pair has a geometric mean damage,

$$\bar{D} = \sqrt{D_i \, D_j},$$

of unity, a one-dimensional pairwise damage onset strain,

$$\varepsilon_{ij,0} = \frac{r - r_0}{r_0},$$

is defined and stored once. Here, r and r_0 are the current and initial distances of this particle pair. A pairwise damage state d_{ij} is subsequently evaluated from the current pairwise strain ε_{ij} and its onset damage strain:

$$d_{ij} = \begin{cases} \frac{\varepsilon_{ij} - \varepsilon_{ij,0}}{2\,\varepsilon_{ij,0}} & \text{if } \varepsilon_{ij} > \varepsilon_{ij,0} \\ 0 & \text{else} \end{cases}$$

All pairwise interactions are scaled with $1 - d_{ij}$ to effect damage. If the pairwise damage state reaches unity, the pairwise interaction is permanently deleted, similar to the visibility criterion mentioned above. Due to the requirement that ε_{ij} must be larger than $\varepsilon_{ij,0}$ to effect damage, failure can only result under tension or shear, but not under compression. The per-integration point damage variable can be accessed using the `compute smd/damage` command.

Note: this failure model can lead to severe instabilities if used with Total-Lagrangian SPH under compression.

10.3.2 *FAILURE_MAX_PAIRWISE_STRAIN

This is a simple failure model based on local strain. It is activated by the following keyword:

```
*FAILURE_MAX_PAIRWISE_STRAIN value
```

Degradation of the material is achieved by considering pairs $< i, j >$ of particles. If the one-dimensional pairwise strain,

$$\varepsilon_{ij} = \frac{r - r_0}{r_0},$$

where r and r_0 are the current and initial distances of this particle pair, exceeds the maximum supplied value `value`, a pairwise damage state d_{ij} is evaluated:

$$d_{ij} = \begin{cases} \frac{\varepsilon_{ij} - \varepsilon_{1D}^{max}}{2\,\varepsilon_{1D}^{max}} & \text{if } \varepsilon_{ij} > \varepsilon_{1D}^{max} \\ 0 & \text{else} \end{cases}$$

All pairwise interactions are scaled with $1 - d_{ij}$ to effect damage. If the pairwise damage state reaches unity, the pairwise interaction is permanently deleted, similar to the visibility criterion mentioned above. Due to the requirement that ε_{ij} must be larger than ε_{1D}^{max} to effect damage, failure can only result under tension or shear, but not under compression. The damage criterion is reversible as long as the bond is not permanently failed.

Note: this failure model is numerically far more stable than the plastic strain based failure criterion from above.

11

Boundary conditions

This is work in progress. See the documentation of the indvidual fixes mentioned below for some info.

11.1 Constraints imposed directly on the particles

The most straighforward way is to impose Dirichlet boundary conditions directly at the particles

- `fix setforce` for forces
- `fix setvel` for velocities

11.2 Rigid walls from surface tesselation geometries

`SMOOTH MACH DYNAMICS` supports loading triangulated surfaces from .STL format files. These surfaces can act as a rigid wall to constrain teh movement of SPH particles. See:

- `fix smd/wall_surface` for loading a surface
- `fix smd/move_tri_surf` for moving a surface during the course of a simulation.

Also look at the `funnel_flow` example. The current coordinates of the triangulated surfaces can be saved to a trajectory file using the `compute smd/triangle_vertices` command and the `dump custom` command. The so produced dump file can be converted into VTK file format using the `dump2vtk_tris` program available int the `tools` directory.

12 Time integration

12.1 Time integration for the Total-Lagrangian pair style

SPH particles for which the Total-Lagrangian pair style is defined should be time integrated with the `fix smd/integrate_tlsph` command. This fix performs time the usual Velocity-Verlet integration of position and velocity as well as Euler integration for the internal energy and mass density. Additionally, some logic is implemented to define and updated the reference configuration if needed. The fix is invoked with:

```
fix ID group-ID smd/integrate_tlsph keyword values
```

- ID, group-ID are documented in fix command
- `smd/integrate_tlsph` = style name of this fix command
- zero or more keyword/value pairs may be appended:
  - keyword `limit_velocity value`: reduce velocity of any particle if it exceeds `value`. This destroys conservation of total energy but can help when dealing with instabilities.

12.2 Time integration for the updated Lagrangian pair style

SPH particles for which the updated Lagrangian pair style is defined should be time integrated with the `fix smd/integrate_ulsph` command. This fix performs time the usual Velocity-Verlet integration of position and velocity as well as Euler integration for the internal energy and mass density. The fix is invoked with:

```
fix ID group-ID smd/integrate_ulsph keyword values
```

- ID, group-ID are documented in fix command
- `smd/integrate_ulsph` = style name of this fix command
- zero or more keyword/value pairs may be appended:
  - keyword `limit_velocity value`: reduce velocity of any particle if it exceeds `value`. This destroys conservation of total energy but can help when dealing with instabilities.
  - keyword `adjust_radius factor min_nn max_nn`: determine the SPH smoothing kernel radius h dynamically such that a number of neighbors between `min_nn` and `max_nn` is obtained.

12.3 Time integration for the Material Point Method pair style

You do not need an extra fix for performing time integration of particles which interact with the MPM pair style. Time integration of these particles is handled inside the pair style itself. If you run a combination of MPM particles and other particles, you only need to define a time integration fix for the other particles.

12.4 Stable timestep

For any explicit time integration scheme, the time increment δt must satisfy the CFL-criterion (Courant, Levy, and Friedrichs, see),

$$\delta t < \frac{h}{c_0}, \tag{12.1}$$

where h is a characteristic distance between integration points and c_0 is the speed at which information propagates. For a liquid this speed depends on the bulk modulus K and the mass density ρ, $c_0 = \sqrt{K/\rho}$. In a solid body, however, the additional shear stiffness G increases the speed to $c_0 = \sqrt{(K+4G/3)/\rho}$. It is only possible for linear equations of state or constitutive models to precompute the speed of information propagation. The `fix smd/adjust_dt` enables the computation of the stable time increment for each particle, considering the current effective bulk and shear moduli K and G, which may change over the course of a simulation if a non-linear material model is used. The resulting time increment is the smallest computed time increment of all particles for which the fix is defined. This fix computes stable time increments both for all `SMD` pair styles.

```
fix ID group-ID smd/adjust_dt factor
```

- ID, group-ID are documented in fix command
- `smd/adjust_dt` = style name of this fix command
- keyword `factor`: scalar prefactor for the CFL criterion, $\delta t =$ `factor` $\times h/c_0$

In the case of SPH, h is taken as the SPH smoothing kernel radius and `factor` should be approximately 0.1. In the case of the bond based Peridynamic model, `smd/peri_ipmb`, `factor` should be approximately 0.5. For the Material Point Method, pair style `smd/mpm_linear`, h is the cell width and `factor` should be approximately 0.5.

Fix Output:

- This fix returns a scalar which keeps track of the total time, i.e., the sum of all time increments.
- this fix returns a vector of values, which gives the computed stable time increment for the various `SMD` methods. If a method is not used in a simulation, the values reported in this vector will attain default values, either zero or a very large number. The purpose of this output information is to see which part of the simulation limits the current time step, e.g. a continuum method or a contact model. Each component of this vector belongs to the following pair style.
  - index 1: Total-Lagrangian SPH, pair style `smd/tlsph`
  - index 2: Updated SPH, pair style `smd/ulsph`
  - index 3: Triangulated repulsive walls, pair style `smd/tri_surface`
  - index 4: Hertzian contact force, pair style `smd/hertz`
  - index 5: bond based Peridynamic model, pair style `smd/peri_ipmb`
  - index 6: Material Point Method, pair style `smd/mpm_linear`

12.5 Run duration

The duration of a run may be limited with the `smd_run_duration` command.

```
smd_run_duration value
```

If this command is supplied before a run is started, the run will terminate if the requested length in physical time units, i.e., the sum of all time increments over all time steps, is reached. Note that you have to run the simulation for a sufficient number of time steps so the desired duration can actually be reached, i.e., the `run` command must be used with large number.

13 Access to particle quantities

To access particle quantities, e.g., stress and strain tensors, or the mass density, different mechanisms are available, depending on the actual quantity. Some scalar quantities are accessed via the `variable` command, others via a `compute` command. In the following, only the quantities specific to the `USER-SMD` package are discussed.

13.1 Particle volume and mass

Particle volumes and masses are accessed via the `variable` command:

```
variable name1 atom volume
variable name2 atom mass
```

Here, `name1` is a vector of length (total number of particles) and contains the particles' volumes. `name2` similarly holds the particles' masses.

13.1.1 Mass density

Mass density is accessed via a `compute` command:

```
compute name group smd/rho
```

This command creates a vector of length (total number of particles), named `name` which holds the particles' mass density values. Note that these values are only meaningful if the `smd/ulsph` pair style is used, as only this pair style performs time integration of the mass density.

13.1.2 Contact radius

Contact radii are accessed via a `compute` command:

```
compute name group smd/contact_radius
```

This command creates a vector of length (total number of particles), named `name` which holds the particles' contact radius values.

13.1.3 Deformed particle shape

The deformation of particles interacting with the Total-Lagrangian SPH scheme can be accessed via a `compute` command:

```
compute name group smd/tlsph_shape
```

This command creates an array of length (total number of particles) * 7, named `name` which holds the particles' stretch axes and rotation. The first three values for each particle correspond to `contact_radius`, scaled with the ε_{xx}, ε_{yy}, ε_{zz} strains. These values define the half-axes of an ellipsoid that is stretched according to the strain tensor of the particle. The next four entries give the rotation of the ellipsoid as a quaternion (component order: Q, X, Y, Z).

13.1.4 SPH kernel diameter

The SPH kernel diameter cannot be directly accessed via a `variable` or `compute` command. However, the `dump custom` command can be used to output the quantity `radius`, which is one half of the SPH kernel diameter.

13.1.5 Equivalent plastic strain

Equivalent plastic strain is accessed via a `compute` command:

```
compute name group smd/plastic_strain
```

This command creates a vector of length (total number of particles), named `name` which holds the particles' equivalent plastic strain. Note that these values are only meaningful if the `smd/tlsph` pair style is used, as only this pair style computes the equivalent plastic strain.

13.1.6 Equivalent plastic strain rate

Equivalent plastic strain rate is accessed via a `compute` command:

```
compute name group smd/plastic_strain_rate
```

This command creates a vector of length (total number of particles), named `name` which holds the particles' equivalent plastic strain rate. Note that these values are only meaningful if the `smd/tlsph` pair style is used, as only this pair style computes the equivalent plastic strain rate.

13.1.7 Energy and temperature

`SMD` distinguishes two types of mechanical energies: potential energy and heat. The potential energy corresponds to the elastic energy in the context of mechanics, while heat energy is proportional to the temperature. Temperature is calculated via the following expression, assuming a constant heat capacity as read in from the general section of the `materials.ini` file:

$$T = \frac{q}{m\, c_p}$$

Here, m is the mass of the particle, and c_p is the constant pressure heat capacity. Potential energy, heat and temperature are accessed via a `compute` command:

```
compute name group smd/energy
```

This command creates an array of length (total number of particles) * 3, named `name` which holds the particles' potential energy (index 1), heat (index 2), and temperature (index 3).

13.1.8 Damage

The scalar damage variable, which is used for some but but not all material models to degrade stiffness and/or strength, is accessed via a `compute` command:

```
compute name group smd/damage
```

This command creates a vector of length (total number of particles), named `name` which holds the particles' damage variable.

13.1.9 Number of Particles within the SPH kernel range

The number of particles interacting with a given particle, i.e., those particles which are spatially closer to the given particle than the SPH kernel radius, is accessed via a `compute` command:

```
compute name group smd/tlsph_num_neighs
compute name group smd/ulsph_num_neighs
```

This command creates a vector of length (total number of particles), named `name` which holds the particles' number of interacting neighbours. Note that `smd/tlsph_num_neighs` computes this quantity for the `smd/tlsph` pair style, while `smd/ulsph_num_neighs` needs to be used with the `smd/ulsph` pair style.

13.1.10 Hourglass error

The hourglass error, defined as the SPH average of the deviation between the linear displacement described by the deformation gradient and the actual displacement for a particle, is accessed via a `compute` command:

```
compute name group smd/hourglass_error
```

This command creates a vector of length (total number of particles), named `name` which holds the particles' hourglass error variable. Note that these values are only meaningful if the `smd/tlsph` pair style is used, as only this pair style computes the hourglass error.

13.1.11 Stress tensor

The Cauchy stress tensor is accessed via a `compute` command:

```
compute name group smd/stress
```

This command creates an array of length (total number of particles) * 7, named `name` which holds the particles' Cauchy stress components and the equivalent von Mises stres. Note that, if different SMD methods are used in a simulation for the same particle vis the `hybrid/overlay` pair style, e.g. MPM and TLSPH, the reported stress is the *sum* of the individual stress tensors computed by each method. The first six values for each particle correspond to the σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{xz}, and σ_{yz} components of the symmetric stress tensor, while the 7^{th} component ist the von Mises euqivalent stress, i.e., the second invariant of the stress tensor.

13.1.12 Strain tensor

The Green-Lagrange strain tensor for SPH particles is accessed via a `compute` command:

```
compute name group smd/tlsph_strain
```

This command creates an array of length (total number of particles) * 6, named `name` which holds the particles' Green-Lagrange strain components.

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\boldsymbol{F}_t^T \boldsymbol{F}_t - \boldsymbol{I}\right),$$

where $\boldsymbol{F}_t$ is the deformation gradient tensor and $\boldsymbol{I}$ is the diagonal unit matrix. The six values for each particle correspond to the ϵ_{xx}, ϵ_{yy}, ϵ_{zz}, ϵ_{xy}, ϵ_{xz}, and ϵ_{yz} components of the symmetric strain tensor. Note that the total deformation gradient is used here, as defined in section 1.5. These values are only meaningful is the `smd/tlsph` pair style is used, as only this pair style computes the deformation gradient.

13.1.13 Strain rate tensor

The time derivative of the strain tensor for SPH particles is accessed via a `compute` command:

```
compute name group smd/tlsph_strain_rate
compute name group smd/ulsph_strain_rate
```

This command creates an array of length (total number of particles) * 6, named `name` which holds the particles' strain rate tensor components. The strain rate is computed as the symmetric part of the velocity gradient $\boldsymbol{L}$,

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\left(\boldsymbol{L} + \boldsymbol{L}^T\right),$$

where

$$\boldsymbol{L} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{bmatrix}.$$

This quantity is computed both by the `smd/tlsph` and `smd/ulsph` pair styles. The six values for each particle correspond to the $\dot{\epsilon}_{xx}$, $\dot{\epsilon_{yy}}$, $\dot{\epsilon_{zz}}$, $\dot{\epsilon_{xy}}$, $\dot{\epsilon_{xz}}$, and $\dot{\epsilon_{yz}}$ components of the symmetric strain tensor.

Bibliography

[1] L. B. Lucy, A numerical approach to the testing of the fission hypothesis, The Astronomical Journal 82 (1977) 1013–1024. doi:10.1086/112164.

[2] R. A. Gingold, J. J. Monaghan, Smoothed particle hydrodynamics - Theory and application to non-spherical stars, Monthly Notices of the Royal Astronomical Society 181 (1977) 375–389.

[3] V. Springel, Smoothed Particle Hydrodynamics in Astrophysics, Annual Review of Astronomy and Astrophysics 48 (1) (2010) 391–430. doi:10.1146/annurev-astro-081309-130914.

[4] M. Gomez-Gesteira, B. D. Rogers, R. A. Dalrymple, A. J. C. Crespo, State-of-the-art of classical SPH for free-surface flows, JHR 48 (extra) (2010) 6–27. doi:10.3826/jhr.2010.0012.

[5] L. D. Libersky, A. G. Petschek, Smooth particle hydrodynamics with strength of materials, in: H. E. Trease, M. F. Fritts, W. P. Crowley (Eds.), Advances in the Free-Lagrange Method Including Contributions on Adaptive Gridding and the Smooth Particle Hydrodynamics Method, Vol. 395 of Lecture Notes in Physics, Berlin Springer Verlag, 1991, pp. 248–257. doi:10.1007/3-540-54960-9_58.

[6] J. Swegle, D. Hicks, S. Attaway, Smoothed Particle Hydrodynamics Stability Analysis, Journal of Computational Physics 116 (1) (1995) 123–134. doi:10.1006/jcph.1995.1010.

[7] C. T. Dyka, P. W. Randles, R. P. Ingle, Stress Points for Tension Instability in SPH, Int. J. Numer. Meth. Eng. 40 (1997) 2325–2341.

[8] D. L. Hicks, J. W. Swegle, S. W. Attaway, Conservative smoothing stabilizes discrete-numerical instabilities in SPH material dynamics computations, Applied Mathematics and Computation 85 (1997) 209–226. doi:http://dx.doi.org/10.1016/S0096-3003(96)00136-1.

[9] J. Monaghan, On the problem of penetration in particle methods, Journal of Computational Physics 82 (1) (1989) 1–15. doi:16/0021-9991(89)90032-6.

[10] P. Randles, L. Libersky, Smoothed Particle Hydrodynamics: Some recent improvements and applications, Computer Methods in Applied Mechanics and Engineering 139 (1) (1996) 375–408.

[11] J. P. Gray, J. J. Monaghan, R. P. Swift, SPH elastic dynamics, Computer Methods in Applied Mechanics and Engineering 190 (49-50) (2001) 6641–6662. doi:10.1016/S0045-7825(01)00254-7.

[12] T. Belytschko, Y. Guo, W. Kam Liu, S. Ping Xiao, A unified stability analysis of meshless particle methods, International Journal for Numerical Methods in Engineering 48 (9) (2000) 1359–1400. doi:10.1002/1097-0207(20000730)48:9<1359::AID-NME829>3.0.CO;2-U.

[13] J. Bonet, S. Kulasegaram, Remarks on tension instability of Eulerian and Lagrangian corrected smooth particle hydrodynamics (CSPH) methods, International Journal for Numerical Methods in Engineering 52 (2001) 1203–1220. doi:10.1002/nme.242.

[14] J. Bonet, S. Kulasegaram, Alternative Total Lagrangian Formulations for Corrected Smooth Particle Hydrodynamics (CSPH) Methods in Large Strain Dynamic Problems, Revue EuropÃl'enne des ÃL'lÃl'ments 11 (7-8) (2002) 893–912. doi:10.3166/reef.11.893-912.

[15] T. Rabczuk, T. Belytschko, S. Xiao, Stable particle methods based on Lagrangian kernels, Computer Methods in Applied Mechanics and Engineering 193 (12âĂŞ14) (2004) 1035–1063. doi:10.1016/j.cma.2003.12.005.

[16] R. Vignjevic, J. Reveles, J. Campbell, SPH in a Total Lagrangian Formalism, Computer Modelling in Engineering and Sciences 146 (2) (3) (2006) 181–198.

[17] S. Xiao, T. Belytschko, Material stability analysis of particle methods, Advances in Computational Mathematics 23 (1-2) (2005) 171–190. doi:10.1007/s10444-004-1817-5.

[18] D. P. Flanagan, T. Belytschko, A uniform strain hexahedron and quadrilateral with orthogonal hourglass control, Int. J. Numer. Meth. Eng. 17 (1981) 679–706.

[19] J. J. Monaghan, An introduction to SPH, Computer Physics Communications 48 (1) (1988) 89–96. doi:10.1016/0010-4655(88)90026-4.

[20] J. Bonet, T. S. L. Lok, Variational and momentum preservation aspects of Smooth Particle Hydrodynamic formulations, Computer Methods in Applied Mechanics and Engineering 180 (1-2) (1999) 97–115. doi:10.1016/S0045-7825(99)00051-1.

[21] R. Vignjevic, J. Campbell, L. Libersky, A treatment of zero-energy modes in the smoothed particle hydrodynamics method, Computer Methods in Applied Mechanics and Engineering 184 (1) (2000) 67–85. doi:10.1016/S0045-7825(99)00441-7.

[22] J. Bonet, R. D. Wood, Nonlinear Continuum Mechanics for Finite Element Analysis, Cambridge University Press, 2008.

[23] T. Belytschko, W. K. Liu, B. Moran, K. Elkhodary, Nonlinear Finite Elements for Continua and Structures, 2nd ed., John Wiley & Sons, 2013.

[24] L. M. Taylor, D. P. Flanagan, PRONTO 2d, A Two-Dimensional Transient Solid Dynamics Program, in: Sandia report SAND86-0594, 1987.

[25] R. Gingold, J. Monaghan, Kernel estimates as a basis for general particle methods in hydrodynamics, Journal of Computational Physics 46 (3) (1982) 429–453. doi:10.1016/0021-9991(82)90025-0.

[26] O. Buzzi, D. M. Pedroso, A. Giacomini, Caveats on the implementation of the generalized material point method, Computer Modeling in Engineering and Sciences 1 (2008) 1–21.

[27] A. F. Bower, Applied Mechanics of Solids, CRC Press, 2011.

[28] T. Belytschko, Y. Y. Lu, L. Gu, Element-free Galerkin methods, Int. J. Numer. Meth. Engng. 37 (2) (1994) 229–256. doi:10.1002/nme.1620370205.

[29] S. Silling, E. Askari, A meshfree method based on the peridynamic model of solid mechanics, Computers & Structures 83 (17-18) (2005) 1526–1535. doi:10.1016/j.compstruc.2004.11.026.

Index