Smooth Mach Dynamics – Manual[*]

Georg C. Ganzenmüller[†]

April 11, 2015

This document 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 [1]. This document and the SMOOTH MACH DYNAMICS software are distributed under the GNU General Public License.

✧ ✧ ✧

✧ ✧ ✧

Acknowledgement

Acknowledgement goes to Prof. Dr. Stefan Hiermaier, who provided the resources for developing SMOOTH MACH DYNAMICS.

TODO

I'm sure there is always something to do.

[*] Draft Version 0.12, April 11, 2015

[†] Email: georg.ganzenmueller@emi.fraunhofer.de

[1] http://lammps.sandia.gov

Contents

1 A little background

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

$$\rho J = \rho_0 \tag{1.3}$$
$$\ddot{u} = \frac{1}{\rho_0}\boldsymbol{\nabla}_0 \cdot \boldsymbol{P}^T \tag{1.4}$$
$$\dot{e} = \frac{1}{\rho_0}\dot{\boldsymbol{F}} : \boldsymbol{P}, \tag{1.5}$$

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,

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

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,

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

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:

$$\begin{aligned} \boldsymbol{f}_i &= \sum_{j\in\mathscr{S}} V_i^0 V_j^0 \left(\boldsymbol{P}_i \nabla W_i(X_{ij}) + \boldsymbol{P}_j \nabla W_i(X_{ij})\right) & (1.29)\\ &= \sum_{j\in\mathscr{S}} V_i^0 V_j^0 \left(\boldsymbol{P}_i \nabla W_i(X_{ij}) - \boldsymbol{P}_j \nabla W_j(X_{ji})\right). & (1.30)\end{aligned}$$

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.

$$\begin{aligned} \boldsymbol{X} &\leftarrow \boldsymbol{x} & (1.32)\\ V &\leftarrow JV & (1.33)\end{aligned}$$

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

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

2 Installation instructions

2.1 Obtaining the code

Note that the following method will change in the near future as soon as USER−SMD will be officially incorporated into the main LAMMPS release. For the meantime, the code (ca. 80 MB) can be downloaded from Github:

1. via downloading a Zip archive
   `https://github.com/ganzenmg/lammps/archive/master.zip`
2. via cloning with git using the following shell command:
   `shell $> git clone https://github.com/ganzenmg/lammps.git`

Depending on the chosen download method (and unzipping, of course), a main LAMMPS directory containing, amongst others, the subdirectories `/src` and `/examples` should be obtained.

2.2 Building the code

For compilation, a standard Linux installation (Ubuntu 12.04 and 14.04 have been tested) with the GNU compiler suite is fine. Change to the `/src` directory and issue the following commands:

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

This will create the executable `lmp_serial_debug` with all USER−SMD capabilities enabled. Additional executables for multiprocessor systems may be generated using the `openmpi` or similar make targets, however, such a build setup is outside the scope of this document.

3 Simulation examples

3.1 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 parameters for the simulation. Most noteworthy, a lattice spacing of 1 mm is used and particles are created on a quadratic lattice. The SPH kernel diameter is set to three times this value, such that approximately 20 neighbours interact with each particle. 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 to effect tensile loading.

The material is modelled using the Total-Lagrangian pair style `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 and the shear response 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. Such an approach is adequate for elastic deformations, but problems involving large amounts of plastic flow should employ regular updates of the reference configuration via the `update` keyword. A stable time increment for the explicit Velocity-Verlet integration approach used by `LAMMPS` is computed every fifth time-step by the `fix smd/adjust_dt` command.

Input file for tensile simulation

```
# TLSPH example: elongate a 2d strip of elastic material py pulling its ends apart

# define initial parameters
variable        l0 equal 1.0 # lattice spacing for creating particles
variable        rho equal 1 # initial mass density
variable        E equal 1.0 # Young's modulus
variable        nu equal 0.3 # Poisson ratio
variable        h equal 3.01*${l0} # SPH smoothing kernel diameter
variable        vol_one equal ${l0}^2 # volume of one particle
variable        mass_one equal ${vol_one}*${rho} # mass of one particle
variable        skin equal 0.1*${l0} # Verlet list range
variable        vel equal 0.001 # pull velocity

units           si # si means: a consistent system of units is used. Here: mm / GPa / ms
dimension       2 # this is a 2d plane-strain simulation
boundary        sm sm p # simulation box boundaries
atom_style      smd
neighbor        ${skin} bin
neigh_modify    every 10 delay 0 check yes # re-build neighbor list every 10 steps
newton          off # required for SMD simulations!
comm_modify     vel yes # required for SMD simulations!

# create simulation box and particles
lattice         sq ${l0}
region          box block -10 10 -10 10 -0.01 0.01 units box
create_box      1 box
create_atoms    1 box
set             group all mass ${mass_one}
set             group all volume ${vol_one}
set             group all diameter ${h} # set SPH kernel radius

# create particle groups for applying displacement boundary conditions
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 # top row of particles
group           bot region bot # bottom row of particles

pair_style      tlsph # activate Total-Lagrangian SPH
#                           rh0    E    nu    Q1    Q2   hg    Cp
pair_coeff      1 1 *COMMON ${rho} ${E} ${nu} 0.06  0.0  10 1.0 &
                *LINEAR_STRENGTH &
                *EOS_LINEAR &
                *END

compute         S all smd/tlsph_stress # Cauchy stress tensor
compute         nn all smd/tlsph_num_neighs # number of neighbors for each particle

dump            dump_id all custom 100 dump.LAMMPS id type x y z &
                c_S[1] c_S[2] c_S[3] c_S[4] c_S[5] c_S[6] c_nn
dump_modify     dump_id first yes

# apply velocity boundary condition to top and bottom rows of particles
velocity        top set 0.0  ${vel} 0.0 sum no units box
velocity        bot set 0.0 -${vel} 0.0 sum no units box
fix             topfix top setforce 0 0 0
fix             botfix bot setforce 0 0 0
```

```
# define variables for outputting stress/strain curve. obtain stress as average of top and
# bottom forces and divide by the initial cross section area (20 mm * 1 mm)
variable        stress equal 0.5*(f_botfix[2]-f_topfix[2])/20
variable        strain equal (2.0*${vel}*time/20) # strain: relative elongation
fix             integration_fix all smd/integrate_tlsph
fix             stress_curve all print 10 "${strain} ${stress}" file stress_strain.dat screen no
fix             dtfix all smd/adjust_dt 5 0.1

thermo 100
thermo_style    custom step dt time v_strain

run             10000
```

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_debug < rubber_strip_pull.lmp
```

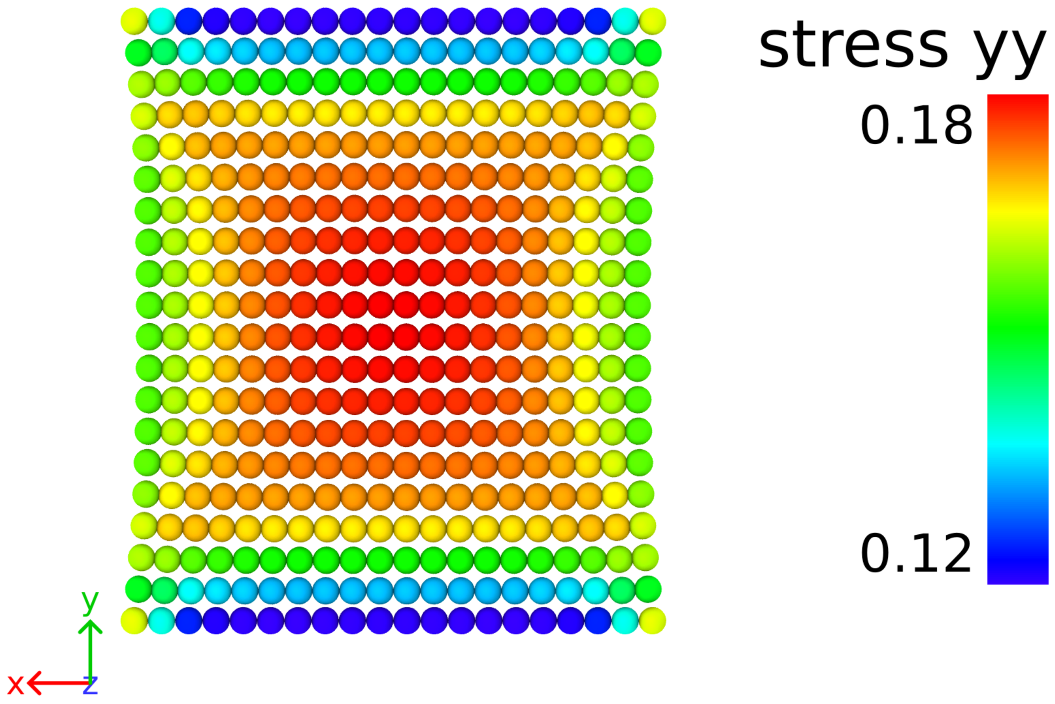

Figure 3.1: Rubber strip pull simulation. The colour coding shows the final state of the yy-stress distribution (units: GPa).

Figure (3.1) shows the final state of the simulation at an achieved engineering strain of ≈ 0.15. The visualisation is obtained using the program OVITO [26], 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 (3.2), 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.

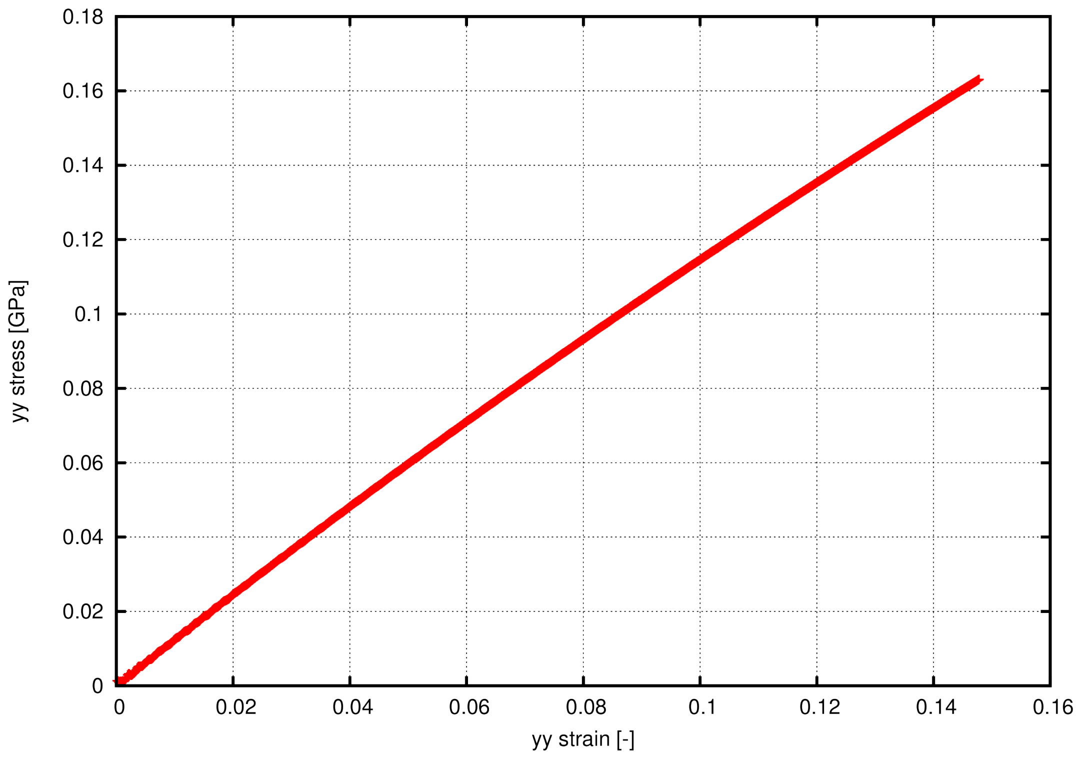

Figure 3.2: Rubber strip pull simulation. The graph shows the relation between the yy-components of stress and strain.

4 Creating particles

4.1 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

4.2 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
```

4.3 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, however, mass density and internal energy still need to be set in the input script.

4.3.1 Particle mass

The mass of a particle can be initialised directly:

```
set style ID mass m
```

Here, m is the particle's mass. Alternatively, the mass can be set by specifying a mass density rho:

```
set style ID density rho
```

Assuming that the particle volume V has already been set, the particle's mass is computed as m=V * rho. Note that the quantities mass, volume, and mass density should be set mutually consistent.

4.3.2 Particle volume

The volume of a particle is set by:

```
set style ID vfrac v
```

Note that the sum of all particles' volumes should equal the volume of the body which is represented by the particle. Also note that the quantities mass, volume, and mass density should be set mutually consistent.

4.3.3 Particle mass density

The mass density of a particle is set by:

```
set style ID rho value
```

Note that the quantities mass, volume, and mass density should be set mutually consistent.

4.3.4 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 three times the particle spacing. These quantities are set using:

```
set style ID contact_radius value
set style ID diameter value
```

4.3.5 Particle internal energy

The internal energy of a particle is set by:

```
set style ID internal_energy value
```

5

SPH pair styles

Two different SPH pair styles are implemented in USER-SMD. The pair style tlsph utilises a total Lagrangian formulation which relates the deformation to particle displacements relative to a fixed reference configuration. The pair style 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.

5.1 The total Lagrangian pair style

This pair style is invoked with the following command:

```
pair_style 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 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 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 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 0.5
Q2	not used	
hg	hourglass control coefficient γ	10 to 50
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	$M = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$

5.2 The updated Lagrangian pair style

This pair style is invoked with the following command:

```
pair_style tlsph
pair_coeff <i> <j> *COMMON <rho0> <c0> <Q1> <Q2> <Cp> &
                   *END
```

Here, 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., 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 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 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). No material strength models to calculate elastic deviatoric stresses are supported, however, viscosity models can be activated.
The *COMMON keyword is mandatory.

```
*COMMON <rho0> <c0> <Q1> <Q2> <Cp>
```

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 0.5
`Q2`	not used	
`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	$K = c_0^2 \rho_0$

6 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.

6.1 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:

$$r_{cut} = R_i + R_j \tag{6.1}$$
$$\delta = r_{cut} - r \tag{6.2}$$
$$r_{geom} = \frac{R_i R_j}{r_{cut}} \tag{6.3}$$
$$f_{ij} = \begin{cases} 1.067 K \delta \sqrt{\delta r_{geom}} \; \forall r < r_{cut} & \text{in 2d} \\ 0.168 K \sqrt{\delta r_{geom}} \; \forall r < r_{cut} & \text{in 3d} \end{cases} \tag{6.4}$$

These expressions are derived from the standard form of the Hertzian overlap potential between two elastic spheres with bulk modulus K and the assumption of a Poisson ratio of 1/3 in 2d and 1/4 in 3d. This bulk modulus is set via the argument contact_stiffness, which has units of pressure. A good choice for the contact stiffness is approximately one tenth of the characteristic stiffness of the interacting particle types, i.e., one tenth of the Young's modulus for a solid body or one tenth of 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.

7 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 ϵ_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.

7.1 Equation of state models

7.1.1 *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 for `pair_style tlsph`

7.1.2 *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 tlsph`

7.1.3 `*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 tlsph`

7.1.4 `*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 ulsph`

7.2 Material strength models

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

7.2.1 *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.

7.2.2 *LINEAR_PLASTICITY

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

```
*LINEAR_PLASTICITY yield_stress
```

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 plastic yield stress. 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 1 number:

parameter	meaning
`yield_stress`	plastic yield stress σ_{yield}

7.2.3 *STRENGTH_JOHNSON_COOK

7.3 Damage and failure models

Damage and failure models are only defined for the `tlsph` pair style.

7.3.1 *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`, the damage variable D starts to grow irreversibly at a rate given by

$$\frac{\mathrm{dD}}{\mathrm{d}t} = \frac{c_0}{100\,h},$$

where c_0 is the reference speed of sound, h is the particle's SPH kernel diameter, and the factor 100 is neccessary for numerical stability. The maximum value of D is limited to 1. The damage variable is used to scale the stress tensor of a particle:

$$\boldsymbol{\sigma} = \begin{cases} (1-D)(-p\boldsymbol{I} + \sigma_d) & \text{if } p < 0 \\ -p\boldsymbol{I} + (1-D)\sigma_d & \text{if } p \geq 0 \end{cases}$$

This scaling achieves that failed particles can still bear compressive load but no cohesion or shear load in case of tension ($p < 0$). The damage variable can be accessed using the `compute smd/damage` command.

7.3.2 *FAILURE_MAX_PRINCIPAL_STRAIN

7.3.3 *FAILURE_JOHNSON_COOK

8 Time integration

8.1 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 `reinit`: if defined, re-initialise the reference configuration from the current particle coordinates when this fix is invoked. This can be useful if several runs are started from one input script.
  - keyword `update`: if defined, periodically re-initialise the reference configuration from the current particle coordinates. This action is performed, whenever the relative displacement of an interacting pair of particles exceeds one half of the Verlet skin distance. This option is useful if problems with large amounts of plastic flow are simulated.
  - keyword `xsph`: use Monaghan's XSPH time integration scheme for velocities
  - 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`: adjust the SPH smoothing kernel radius h between updates according to: $h \leftarrow h \times J^{(1/d)}$, where J is the determinant of the incremental deformation tensor and d is the dimensionality of the system. This keyword can only be used together with the `update` keyword.
- default: None

8.2 Time integration for the updated Lagrangian pair style

SPH particles which 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 `xsph`: use Monaghan's XSPH time integration scheme for velocities
  - 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 value`: determine the SPH smoothing kernel radius h dynamically according to: $h = \texttt{value} \times (m/\rho)^{(1/d)}$, where m and ρ are particle mass and mass density , respectively, and d St the dimensionality of the system.
- default: None

8.3 Stable timestep

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

$$\delta t < \frac{h}{c_0}, \tag{8.1}$$

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 values of K and G. The resulting time increment is the smallest computed time increment of all particles for which the fix is defined. This fix computes stable timesteps both for `ulsph` and `tlsph` 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, $\delta t = \texttt{factor} \times h/c_0$

h is taken as the SPH smoothing kernel radius. Typically, `factor=0.1`.

Fix Output:

- This fix returns a scalar which keeps track of the total time, i.e., the sum of all time increments.

9 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.

9.1 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.

9.1.1 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.

9.1.2 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.

9.1.3 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.

9.1.4 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.

9.1.5 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.

9.1.6 Internal energy

The thermodynamic internal energy (the sum of work and heat energies) is accessed via a compute command:

```
compute name group smd/internal_energy
```

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

9.1.7 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.

9.1.8 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 `tlsph` pair style, while `smd/ulsph_num_neighs` needs to be used with the `ulsph` pair style.

9.1.9 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.

9.1.10 Stress tensor

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

```
compute name group smd/tlsph_stress
compute name group smd/ulsph_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 `smd/tlsph_stress` computes this quantity for the `tlsph` pair style, while `smd/ulsph_stress` needs to be used with the `ulsph` pair style. 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.

9.1.11 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.

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

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.

9.1.12 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.

[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.

[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, Journal of Hydraulic Research 48 (extra) (2010) 6–27.

[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.

[6] J. Swegle, D. Hicks, S. Attaway, Smoothed particle hydrodynamics stability analysis, Journal of Computational Physics 116 (1) (1995) 123–134.

[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.

[9] J. Monaghan, On the problem of penetration in particle methods, Journal of Computational Physics 82 (1) (1989) 1–15.

[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.

[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.

[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 (11) (2001) 1203–1220.

[14] J. Bonet, S. Kulasegaram, Alternative total lagrangian formulations for corrected smooth particle hydrodynamics (CSPH) methods in large strain dynamic problems, Revue EuropÃľenne des ÃĽlÃľments 11 (7-8) (2002) 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.

[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.

[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. Monaghan, An introduction to SPH, Computer Physics Communications 48 (1) (1988) 89–96.

[20] J. Bonet, T.-S. Lok, Variational and momentum preservation aspects of smooth particle hydrodynamic formulations, Computer Methods in Applied Mechanics and Engineering 180 (1âĂŞ2) (1999) 97–115.

[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.

[22] J. Bonet, R. D. Wood, Nonlinear Continuum Mechanics for Finite Element Analysis, Cambridge University Press, Cambridge, 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, Sandia National Laboratories, Albuquerque, New Mexico, USA, 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.

[26] A. Stukowski, Visualization and analysis of atomistic simulation data with OVITO – the open visualization tool, Modelling and Simulation in Materials Science and Engineering 18 (1) (2010) 015012.

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