The implementation of Smooth Particle Hydrodynamics in LAMMPS.

A guide to the SPH-USER package.

Georg C. Ganzenmüller* and Martin O. Steinhauser
Fraunhofer Ernst-Mach Institut für Hochgeschwindigkeitsdynamik
Freiburg, Germany

Paul Van Liedekerke
Faculty of Bio-Engineering, MEBIOS Division
Katholieke Universtiteit Leuven
Leuven, Belgium

July 17, 2011

*georg.ganzenmueller@emi.fraunhofer.de

Contents

1. Introduction

This document describes the implementation of the Smooth Particle Hydrodynamics (SPH) method within the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS).

LAMMPS is a particle simulation code, developed and maintained at Sandia National Laboratories, USA. While is primarily aimed at Molecular Dynamics simulations of atomistic systems, it provides a general, fully parallelized framework for particle simulations governed by Newton's equations of motion.

SPH is a continuum method, which does not require a predefined grid to evaluate the associated partial differential field equations of continuum mechanics. Instead, SPH discretises the mass distribution field into point masses which move with the material, according to Newtons equations of motion. The positions of the point masses serve as integration nodes for the field equations of continuum mechanics. The required variable fields are constructed on-the-fly using interpolation kernels, which are centred at the point masses. Due to its particle nature, SPH is directly compatible with the existing code architecture and data structures present in LAMMPS.

1.1. Quick Start Guide

For those who hate reading users' guides, please try the following:

1. Download LAMMPS from http://lammps.sandia.gov and untar the source.
2. In the LAMMPS `src/` directory do `make yes-sph` followed by `make <your platform>` (for example, `make serial`).
3. In the LAMMPS `examples/sph` directory, run the example input script (for example, `lmp_serial < dambreak.lmp`).
4. Visualise results using an appropriate software. We recommend Ovito [8] for this purpose.

2. Getting Started

We assume that you already have a working LAMMPS installation. For more on downloading and building LAMMPS, see `http://lammps.sandia.gov`. This document only provides information related to the SPH module within LAMMPS. For questions regarding the usage of LAMMPS, please see the LAMMPS documentation.

2.1. Building the SPH module within LAMMPS

In the LAMMPS distribution, the SPH is distributed as an add-on module, which means that it is not by default compiled with the rest of LAMMPS. To instruct LAMMPS to build the SPH module, go to the LAMMPS source subdirectory (`/src`) and type

```
make yes-sph
```

followed by

```
make <your platform>
```

to compile LAMMPS on your particular platform.

2.2. Running SPH simulations with LAMMPS

See Sec. 5 for a few examples.

3. SPH Theory

This section gives a concise introduction to SPH for fluids. For more detailed information, the reader is referred to the excellent books by Hoover [2] [1] and Liu [3]. SPH is a method to solve problems in Lagrangian continuum mechanics, where the governing partial differential equations describe the co-moving evolution of the density ρ, coordinates $\mathbf{r}$, velocity $\mathbf{v}$, and energy per unit mass e in terms of gradients of the velocity, pressure tensor[2] P, and the heat-flux vector $Q = \kappa\nabla T$, with thermal conductivity κ and temperature gradient ∇T.

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\rho\nabla\cdot\mathbf{v} \tag{1}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\frac{1}{\rho}\nabla\cdot P \tag{2}$$

$$\frac{\mathrm{d}e}{\mathrm{d}t} = -\frac{1}{\rho}P\colon\nabla\mathbf{v} - \frac{1}{\rho}\nabla\cdot Q \tag{3}$$

SPH interpolates the set of field variables $\{\rho, \mathbf{v}, e, P, Q\}$ by means of kernel interpolation. For any variable $f(\mathbf{r})$, a local average at each coordinate r_i is calculated according to

$$f(\mathbf{r_i}) = \sum_j m_j \frac{f_j}{\rho_j} W(\mathbf{r_i} - \mathbf{r}_j). \tag{4}$$

m_j and f_j are mass of particle j and value of the field $f(\mathbf{r})$ at position $\mathbf{r}_j$, respectively. ρ_j is the value of the mass density at $\mathbf{r}_j$. $W(\mathbf{r_i} - \mathbf{r}_j)$ is a weight (or kernel) function of compact support, which decays to zero within a range h comparable to a few typical inter-particle spacings. Here, only radially symmetric weight functions $W(\mathbf{r_i} - \mathbf{r}_j) \equiv W(r_{ij})$ are considered. Here, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and $r_{ij} = \|\mathbf{r}_{ij}\|$. The sum in Eqn. (4) formally extends over all particles, however, due to the compact support of W, only particles for which $\|\mathbf{r_i} - \mathbf{r}_j\| < h$ need to be considered. The process of local averaging turns the coupled set of partial differential equations (1 – 3) into N uncoupled ordinary differential equations, with N being the number of SPH particles used.

A particularly convenient feature of SPH is that neither m_j nor f_j is affected by the gradient operator ∇. Because the m_j and f_j are themselves particle properties, the gradient operator affects only the weight functions W_{ij}. Therefore the gradient of a vector field $f(\mathbf{r})$, evaluated at position $\mathbf{r}_i$, is obtained as follows:

$$\nabla f(\mathbf{r}_i) = \nabla \sum_j m_j \frac{f_j}{\rho_j} W_{ij} = \sum_j m_j \frac{f_j}{\rho_j} \nabla_j W_{ij} \tag{5}$$

Due to the radial symmetry of W, $\nabla_j W_{ij} = \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|}\frac{\mathrm{d}W_{ij}}{\mathrm{d}r_{ij}}$. This implies the antisymmetry property of the SPH gradient, with $\nabla_j W_{ij} = -\nabla_i W_{ji}$.

3.1. SPH approximation of the local density

The SPH expression for the local density is obtained by setting $f(\mathbf{r_i}) \equiv \rho(r_i) \equiv \rho_i$:

$$\rho_i = \sum_j m_j \frac{\rho_j}{\rho_j} W_{ij} \equiv \sum_j m_j 1 W_{ij}. \tag{6}$$

The above expression is referred to as the *partition of unity*. Note that the local density, calculated at each particle, is a smoothed quantity with contributions from all particles within the h-neighbourhood.

3.2. SPH approximation of the Navier-Stokes equation of motion

In order to obtain a computable expression for the equation of motion, i.e Eqn. (2), the gradient of the pressure tensor needs to be evaluated. We start by noting that the divergence of the quantity (P/ρ) can be rewritten using ordinary calculus as follows:

$$\nabla\cdot\frac{P}{\rho} \equiv -\frac{P}{\rho^2}\cdot\nabla\rho + \frac{1}{\rho}\nabla\cdot P \tag{7}$$

[1] This user guide draws heavily on the book by Hoover [2]

[2] Note the definition of the tensor double product, $A\colon B = \sum_{ij} A_{ij}B_{ij}$

This identity can be rearranged to provide a starting point for the SPH discretisation of the time evolution of the particle velocity,

$$-\frac{1}{\rho}\nabla\cdot P \equiv -\nabla\cdot\frac{P}{\rho}+\frac{P}{\rho^2}\cdot\nabla\rho. \tag{8}$$

Inserting the above line into Eqn. (2), we obtain

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\frac{P}{\rho^2}\cdot\nabla\rho-\nabla\cdot\frac{P}{\rho}. \tag{9}$$

The spatial derivatives, $\nabla\rho$ and $\nabla\cdot\frac{P}{\rho}$ can be discretised using the SPH expression for the gradient of a variable field, Eqn. (5):

$$\nabla\rho = \sum_j m_j\nabla_j W_{ij} \tag{10}$$

$$\nabla\cdot\frac{P}{\rho} = \sum_j m_j\frac{P_j}{\rho_j^2}\nabla_j W_{ij} \tag{11}$$

The equation of motion for particle i now reads

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\frac{P_i}{\rho_i^2}\cdot\sum_j m_j\nabla_j W_{ij}-\sum_j m_j\frac{P_j}{\rho_j^2}\nabla_j W_{ij}, \tag{12}$$

and is immediately written as an expression for pair-wise forces, suitable for implementation in an Molecular Dynamics code:

$$\mathbf{f}_i = m_i\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_j m_i m_j\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}\right)\nabla_j W_{ij}. \tag{13}$$

It is evident that this expression for the force is antisymmetric due to the antisymmetry property of the SPH gradient. It therefore follows that this SPH discretisation preserves total linear momentum.

3.3. SPH approximation of the Navier-Stokes continuity equation

The continuity equation, Eqn. (1), contains the gradient of the velocity field. As above, we begin the SPH discretisation by using the identity

$$\nabla(\mathbf{v}\rho) = \rho\nabla\mathbf{v}+\mathbf{v}\nabla\rho, \tag{14}$$

which enables us to rewrite the continuity equation as

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = \nabla(\mathbf{v}\rho)-\mathbf{v}\nabla\rho. \tag{15}$$

Applying the SPH discretisation of the gradient of a vector field, Eqn. (5), we obtain:

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \sum_j m_j\mathbf{v}_j\nabla_j W_{ij}-\mathbf{v}_i\sum_j m_j\nabla_j W_{ij} = -\sum_j m_j\mathbf{v}_{ij}\nabla_j W_{ij} \tag{16}$$

3.4. SPH approximation of the Navier-Stokes energy equation

In order to derive an SPH expression for the time-evolution of the energy per unit mass, one can, proceed in analogy to the above steps by evaluating the divergence of the RHS of Eqn. (3). Here, we only quote the final result:

$$m_i\frac{\mathrm{d}e_i}{\mathrm{d}t} = -\frac{1}{2}\sum_j m_i m_j\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}\right):\mathbf{v}_{ij}\nabla_j W_{ij}-\sum_j\frac{m_i m_j}{\rho_i\rho_j}\frac{(\kappa_i+\kappa_j)(T_i-T_j)}{r_{ij}^2}\mathbf{r}_{ij}\cdot\nabla_j W_{ij} \tag{17}$$

3.5. SPH artificial viscosity

Numerical integration of the compressible Navier-Stokes equations is generally unstable in the sense, that infinitesimally small pressure waves can steepen due to numerical artifacts and turn into shock waves. In order to suppress this source of instability, Monaghan introduced an extension of the von Newman-Richter artificial viscosity into SPH. An additional viscous component Π_{ij} is introduced into the SPH force expression, Eqn. (13),

$$\mathbf{f}_i = m_i \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_j m_i m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_j W_{ij}, \tag{18}$$

with

$$\Pi_{ij} = -\alpha h \frac{c_i + c_j}{\rho_i + \rho_j} \frac{\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}}{r_{ij}^2 + \epsilon h^2}. \tag{19}$$

Here, c_i and c_j are the speed of sound of particles i and j, α is a dimensionless factor controlling the dissipation strength, and $\epsilon \simeq 0.01$ avoids singularities in the case that particles are very close to each other. For correct energy conservation, the artificial viscosity must be included in the time-evolution of the energy:

$$m_i \frac{\mathrm{d}e_i}{\mathrm{d}t} = -\frac{1}{2} \sum_j m_i m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) : \mathbf{v}_{ij} \nabla_j W_{ij} - \sum_j \frac{m_i m_j}{\rho_i \rho_j} \frac{(\kappa_i + \kappa_j)(T_i - T_j)}{r_{ij}^2} \mathbf{r}_{ij} \cdot \nabla_j W_{ij} \tag{20}$$

Note that the artificial viscosity can be understood [4] in terms of an effective kinematic viscosity ν:

$$\nu = \frac{\alpha h c}{8} \tag{21}$$

3.6. SPH laminar flow viscosity

While the artificial viscosity description usually gives good results for turbulent flows, the spatial velocity profiles may be inaccurate for situations at low Reynolds numbers. To estimate the SPH viscous diffusion term, Morris et. al (1997) [6] resorted to an expression for derivatives similarly as used in computations for heat conduction. The viscous term in the Navier-Stokes equations is now estimated as:

$$\left(\frac{1}{\rho} \nabla \cdot \mu \nabla \mathbf{v} \right)_i = \sum_j \frac{m_j (\mu_i + \mu_j) \mathbf{r}_{ij} \cdot \nabla_j W_{ij}}{\rho_i \rho_j (r_{ij}^2 + \epsilon h^2)} \mathbf{v}_{ij}, \tag{22}$$

with the dynamic viscosity $\mu = \rho \nu$. The final momentum equation reads:

$$\mathbf{f}_i = -\sum_j m_i m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla_j W_{ij} + \sum_j \frac{m_i m_j (\mu_i + \mu_j) \mathbf{v}_{ij}}{\rho_i \rho_j} \left(\frac{1}{r_{ij}} \frac{\partial W_{ij}}{\partial r_i} \right), \tag{23}$$

For correct energy conservation, the viscous entropy production can be included in the time-evolution of the energy as follows:

$$\begin{aligned} m_i \frac{\mathrm{d}e_i}{\mathrm{d}t} = -\frac{1}{2} \sum_j m_i m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) : \mathbf{v}_{ij} \nabla_j W_{ij} - \frac{1}{2} \sum_j \frac{m_i m_j (\mu_i + \mu_j)}{\rho_i \rho_j} \left(\frac{1}{r_{ij}} \frac{\partial W_{ij}}{\partial r_i} \right) v_{ij}^2 \\ - \sum_j \frac{m_i m_j}{\rho_i \rho_j} \frac{(\kappa_i + \kappa_j)(T_i - T_j)}{r_{ij}^2} \mathbf{r}_{ij} \cdot \nabla_j W_{ij} \end{aligned} \tag{24}$$

Boundaries for laminar flows can be performed simply by prescribed fixed particles which participate in the SPH summations, but note that the despite the zero velocity of these particles, the SPH interpolated velocity at that point can be non-zero. This results in errors due to boundary conditions. One should be aware that the timestep in laminar flows can be restricted by the

viscous diffusion, rather than by the CFL criterion[3]. A stable simulation should therefore fulfil:

$$\delta t < 0.125 \frac{h^2}{\nu}. \tag{25}$$

[3]The Courant-Friedrichs-Lewy (CFL) criterion is a necessary but not necessarily sufficient condition for the convergence of the finite-difference approximation of a given numerical problem. In the context of SPH it is typically formulated as $\delta t \leq 0.3c/h$, where c is the speed of sound.

4. Implementation of SPH in LAMMPS

4.1. Data structure

LAMMPS provides data structures for forces, positions and velocities. SPH requires at least four new per-particle variables: local density ρ, internal energy[4] $E = me$, and their respective time derivatives $\dot{\rho}$ and $\dot{E}$. In order to add support for these quantities, a new data structure was created which can be accessed using the following command:

```
atom_style meso
```

This `atom_style` also defines a per-particle heat capacity, such that a per-particle temperature $T_i = E_i/C_{v,i}$ can be calculated. With both ρ and T available, complete equations of state[5] are supported. Additionally, `atom_style meso` defines an *extrapolated velocity*, which is an estimate of a velocity consistent with the positions at the time when forces are evaluated.

4.2. Time stepping

LAMMPS uses a Velocity-Verlet scheme to perform time integration:

1a) $\mathbf{v}_i(t + \frac{1}{2}\delta t) = \mathbf{v}_i(t) + \frac{\delta t}{2m_i}\mathbf{f}_i(t)$

1b) $\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t) + \delta t\mathbf{v}_i(t + \frac{1}{2}\delta t)$

2) – calculate new forces $\mathbf{f}_i(t + \delta t)$ –

3) $\mathbf{v}_i(t + \delta t) = \mathbf{v}_i(t + \frac{1}{2}\delta t) + \frac{\delta t}{2m_i}\mathbf{f}_i(t + \delta t)$

This integration scheme cannot directly be used with SPH because the velocities lag behind the positions by $\frac{1}{2}\delta t$ when the forces are computed. This leads to poor conservation of total mass and energy because the SPH expressions, Eqn. (16) and Eqn. (17) depend explicitly on the velocity. This situation can be improved by computing an extrapolated velocity,

$$\tilde{\mathbf{v}}_i(t + \delta t) = \mathbf{v}_i(t) + \frac{\delta t}{m_i}\mathbf{f}_i(t), \tag{26}$$

prior to the force computation, and basing all SPH expressions on $\tilde{\mathbf{v}}$. This extrapolation, however, is only accurate to $\mathcal{O}(\delta t)$. With the added time-evolution of the local density and internal energy, the complete integration scheme reads:

1a) $\mathbf{v}_i(t + \frac{1}{2}\delta t) = \mathbf{v}_i(t) + \frac{\delta t}{2m_i}\mathbf{f}_i(t)$

1b) $\tilde{\mathbf{v}}_i(t + \delta t) = \mathbf{v}_i(t) + \frac{\delta t}{m_i}\mathbf{f}_i(t)$

1c) $\rho_i(t + \frac{1}{2}\delta t) = \rho_i(t) + \frac{\delta t}{2}\dot{\rho}_i(t)$

1d) $E_i(t + \frac{1}{2}\delta t) = E_i(t) + \frac{\delta t}{2}\dot{E}_i(t)$

1e) $\mathbf{r}_i(t + \delta t) = \mathbf{r}_i(t) + \delta t\mathbf{v}_i(t + \frac{1}{2}\delta t)$

2) – calculate $\mathbf{f}_i(t + \delta t)$, $\dot{\rho}_i(t + \delta t)$, $\dot{E}_i(t + \delta t)$ –

3a) $\rho_i(t + \delta t) = \rho_i(t + \frac{1}{2}\delta t) + \frac{\delta t}{2}\dot{\rho}_i(t + \delta t)$

3b) $E_i(t + \delta t) = E_i(t + \frac{1}{2}\delta t) + \frac{\delta t}{2}\dot{E}_i(t + \delta t)$

3c) $\mathbf{v}_i(t + \delta t) = \mathbf{v}_i(t + \frac{1}{2}\delta t) + \frac{\delta t}{2m_i}\mathbf{f}_i(t + \delta t)$

The splitting of the time-evolution of ρ and E into two separate updates is in analogy with the time integration of $\mathbf{v}$. Simple Euler integration for ρ and E would lead to poor energy conservation. The corresponding command to perform the above time-integration is:

```
fix fix_ID group_ID meso
```

[4] We store the internal energy per particle, not the internal energy per unit mass per particle.
[5] A complete equation of state is not only a function of density, but also of temperature.

4.3. Local density calculation

The local density ρ can be (re-)initialised in two ways:

- By density summation: The SPH density is calculated from scratch using Eqn. (6) by invoking the `pair_style sph/rhosum n` command:

  ```
  pair_style sph/rhosum
  pair_coeff I J h
  ```

  Here, `I` and `J` are the types of SPH particles for which ρ is to be calculated. `n` is the time-step period of summation, i.e., summation is performed every `n` time-steps. During those time-steps when this `pair_style` is not invoked, the usual density continuity equation is used to update ρ. If `n` is 0, ρ is only computed once at the beginning of a run. `h` is the range of the kernel function, which is taken as the following polynomial:

  $$W(r < h) = \frac{1}{s}\left[1 - \left(\frac{1}{h}\right)^2\right]^4 . \tag{27}$$

  Here, s is a normalisation constant which depends on the number of spatial dimensions. This particular form of the kernel function is very appealing from a computational perspective, as it does not require the evaluation of a square root.

- By assigning ρ directly using a `set` command. The local density can be assigned prior to a run using

  ```
  set style ID meso_rho d
  ```

  `style`, `ID` are documented in the LAMMPS users' guide. `d` is the value of ρ.

4.4. Equation of State

The equation of state (EOS) determines pressure as a function of local density ρ and temperature. The following equations of state (EOS) are implemented as `pair_style` commands:

4.4.1. Tait's equation of state with artificial viscosity

the Tait equation of state,

$$P(\rho) = \frac{c_0^2 \rho_0}{7}\left[\left(\frac{\rho}{\rho_0}\right)^7 - 1\right] \tag{28}$$

is an incomplete EOS devised to model water at ambient conditions. c_0 and ρ_0 are the sound speed and density at zero applied stress. It can be selected using

```
pair_style sph/taitwater
pair_coeff I J rho_0 c_0 alpha h
```

Here, `I` and `J` are the types of SPH particles which interact according to this EOS. `rho_0` is ρ_0, `c_0` is c_0, `alpha` sets the strength of the artificial viscosity according to Eqn. (19), and `h` is the range of the Lucy kernel function

$$W(r < h) = \frac{1}{s}\left[1 + 3\frac{r}{h}\right]\left[1 - \frac{r}{h}\right]^3 \tag{29}$$

Note that, because `rho_0` and `c_0` are defined on a per-type basis, you need to specify the `pair_coeff I I` and the `pair_coeff J J` lines before an `pair_coeff I J` line.

4.4.2. Tait's equation of state with laminar viscosity

The Tait EOS can also be combined with Morris' expression for laminar viscosity, Eqn. 22 instead of artificial viscosity. The corresponding syntax is:

```
pair_style sph/taitwater/morris
pair_coeff I J rho_0 c_0 alpha h
```

Here, α is the dynamic viscosity with units Pa s.

4.4.3. Ideal gas equation of state

The ideal gas equation of state reads

$$P(\rho, e) = (\gamma - 1)\,\rho e \tag{30}$$

Here, $\gamma = C_p/C_V$ is the heat capacity ratio. In this implementation, $\gamma = 1.4$, corresponding to dry air. This EOS is selected using the commands

```
pair_style sph/idealgas
pair_coeff I J alpha h
```

Here, `I` and `J` are the types of SPH particles which interact according to this EOS. `alpha` sets the strength of the artificial viscosity according to Eqn. (19), and `h` is the range of the Lucy kernel function, Eqn. (29).

4.4.4. Lennard-Jones equation of state

This EOS is the continuum mechanics equivalent to the Lennard-Jones pair potential:

$$u(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]. \tag{31}$$

The EOS is implemented as a polynomial using the parametrisation of Ree [7], with the Lennard-Jones parameters ϵ and σ set to unity. It is selected using the commands

```
pair_style sph/lj
pair_coeff I J alpha h
```

Here, `I` and `J` are the types of SPH particles which interact according to this EOS. `alpha` sets the strength of the artificial viscosity according to Eqn. (19), and `h` is the range of the Lucy kernel function, Eqn. (29).

4.5. Heat conduction

Thermal conductivity between SPH particles is enabled using the following command:

```
pair_style sph/heatconduction
pair_coeff I J D h
```

Here, `I` and `J` are the types of SPH particles which interact according to this EOS. `D` $= \kappa m/(C_V \rho)$ is the heat diffusion coefficient with units length^2/time. `h` is the range of the Lucy kernel function, Eqn. (29).

4.6. Boundary conditions

In macroscopic simulations, it is often desirable to have boundary conditions which constrain a fluid to a certain region in space. Such boundary conditions are required to be stationary. A possible way of generating hard boundaries is to employ one of the various `fix wall/*` commands available in the main LAMMPS distribution. However, the use of these walls results in poor energy conservation. It is comparatively better to use stationary SPH particles as boundary conditions, of which only the internal energy E and the local density ρ is integrated. A suitable integration fix is provided by

```
fix fix_ID group_ID meso/stationary.
```

4.7. Accessing SPH variables for initialisation and output

4.7.1. Initialisation

Internal energy E, heat capacity C_V, and local density ρ can be set using the following commands:

- `set style ID meso_e d`
- `set style ID meso_cv d`
- `set style ID meso_rho d`

`style`, `ID` are documented in the LAMMPS users' guide. `d` is the value of E, C_V, or ρ respectively. Alternatively, these variables can be read from a LAMMPS data file. The required line format for `atom_style meso` in the `Atoms` section of the data file is:

`atomID atom-type` ρ E C_V x y z

If used in conjunction with `atom_style hybrid`, the required information is ρ E C_V.

4.7.2. Output

The per-particle SPH variables internal energy E, local density ρ, and local temperature $T = E/C_V$ can be accessed using the following `compute` commands:

- `compute compute_ID group_ID meso_e/atom`
- `compute compute_ID group_ID meso_rho/atom`
- `compute compute_ID group_ID meso_t/atom`

These computes return a vector of length N, with N being the number of particles present in the system. These vectors can then be output via the usual LAMMPS mechanisms, e.g. via the `dump custom` command.

5. Validation tests

5.1. Heat conduction

This example considers a 2d bar of SPH particles, with dimensions 100 cm × 10 cm and an inter-particle spacing of 1 cm. The left half is assigned an internal energy of 1 J per particle, the right half 2 J per particle. Heat flows from right to left with a heat diffusion coefficient of 1.0×10^{-4} $m^2 s^{-1}$, via the `sph/heatconduction` pair style. There is neither potential nor kinetic energy present in this simulation, so that energy conservation can be monitored by tracking the total internal energy, computed in variable `ie`. The energy profile is compared to an analytic solution after 4.0 s. This example is available with the distribution of the `SPH-USER` package.

5.1.1. Input script

```
dimension       2
units           si
atom_style      meso
boundary        f p p

# create the system
lattice         sq 0.01
region          box block 0 100 0 10 0 0.1
create_box      1 box
create_atoms    1 box
mass            1 1.0e-5

# define left & right regions, assign internal energies
region          left  block EDGE 49.9  EDGE EDGE EDGE EDGE
region          right block 50 EDGE EDGE EDGE EDGE EDGE
set             region left  meso_e 1.0 # internal energies
set             region right meso_e 2.0

# Note: local density rho should correspond to mass density
# of the system. Otherwise poor results are obtained.
set             group all meso_rho 0.1 # 0.1 = 1.e-5/(0.01)^2

pair_style      sph/heatconduction
#               I | J | diffusion coeff. | cutoff
pair_coeff      1   1   1.0e-4             2.0e-2

# compute internal energy per particle & sum up
compute         ie_atom all meso_e/atom
compute         ie all reduce sum c_ie_atom

thermo          10
thermo_style    custom step temp c_ie
timestep        0.25e-1
neighbor        0.2e-2 bin

# time integration: particles do not move in this setup
fix             integrate_fix all meso/stationary

dump            dump_fix all custom 10 dump.heat id type x y z c_ie_atom
dump_modify     dump_fix first yes
run             160
undump          dump_fix
```

5.1.2. Results

Figure 1 shows the final simulation snapshot, with some of the heat redistributed from the right to the left.

The analytic solution to this problem is obtained from the 1d diffusion equation:

$$E(x,t) = E_l^0 + \frac{E_r^0 - E_l^0}{2} \text{ erf}\left(\frac{x - x_c}{\sqrt{4\alpha t}}\right) \tag{32}$$

Here, E_l^0 and E_r^0 are the initial energies on the left and right sides, respectively. x_c is the contact position between cold and hot regions, α is the diffusion coefficient, and t is time. As shown in Fig. 2, there is good agreement between the SPH-discretised simulation results and the analytic solution.

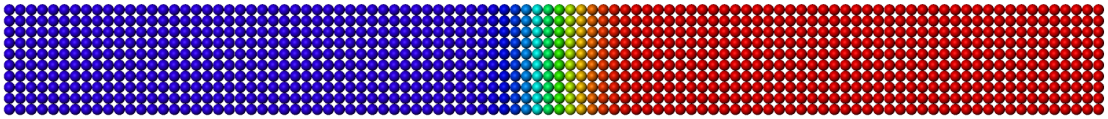

Figure 1: Simulation snapshot for the heat conduction example at $t = 4.0$ s. Colour-coding shows the internal energy per particle.

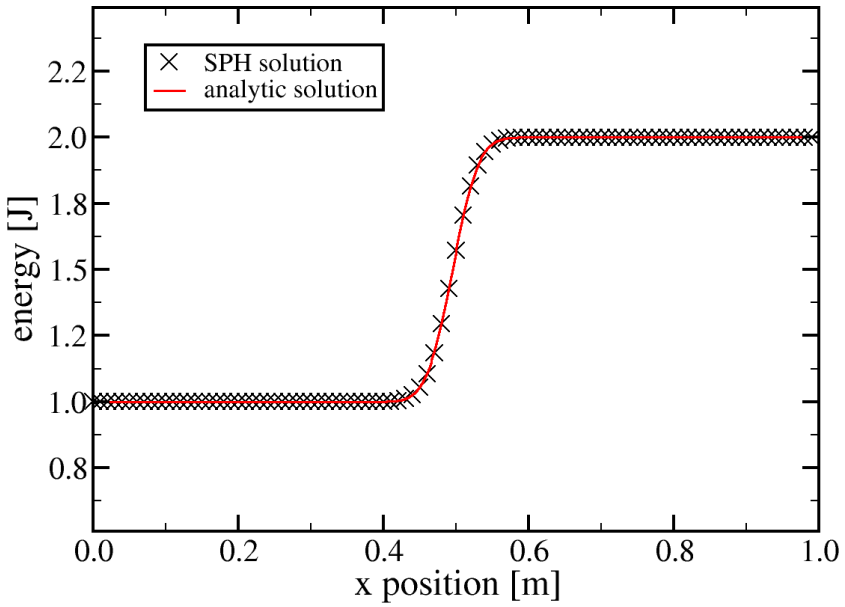

Figure 2: Comparison of the SPH results for the internal energy per particle after $t = 4.0$ s with the analytic solution.

5.2. Shock wave structure

This example considers the solution of a quasi-1d shock wave problem in 2 or 3 spatial dimensions. It is adapted from Monaghan's investigation of the performance of SPH for a 1d shock wave problem [5]. The SPH results are compared with an exact numerical solution.

In reduced units, the initial conditions for the shock problem are defined by two regions of an ideal gas, a high density region on the left with $p = \rho = 1$ in contact with a low density on the right with $p^* = \rho = 0.25$. As the high density gas expands into the low density region, a shock wave travels to the right while a rarefaction wave moves to the left. In 2d (3d), SPH particles are arranged on a square (simple cubic) lattice with spacing 1, extending from $x = -100..150$, $y = -4..4$ (and $z = -4..4$ in 3d). Particles with a negative x-coordinate are high-density particles with $m = 1$, and particles with a positive x-coordinate are low-density particles with $m = 0.25$. We use density summation every timestep and a value for the artificial viscosity of $\alpha = 0.75$. The initial setup, is shown in Fig. 3.

Figure 3: Initial setup for the shock wave problem. Colour represents mass density with red corresponding to $\rho = 1$ and blue to $\rho = 0.25$.

5.2.1. Results

SPH results are compared to an exact solution based on a numerical solution of the corresponding Riemann problem. As shown in Fig. 4, the overall agreement for the mass density is quite satisfactory, with the SPH results being smoothed out over a distance $\simeq h$. Total energy, i.e. the sum of kinetic and internal energy is conserved with an accuracy of about one part per million.

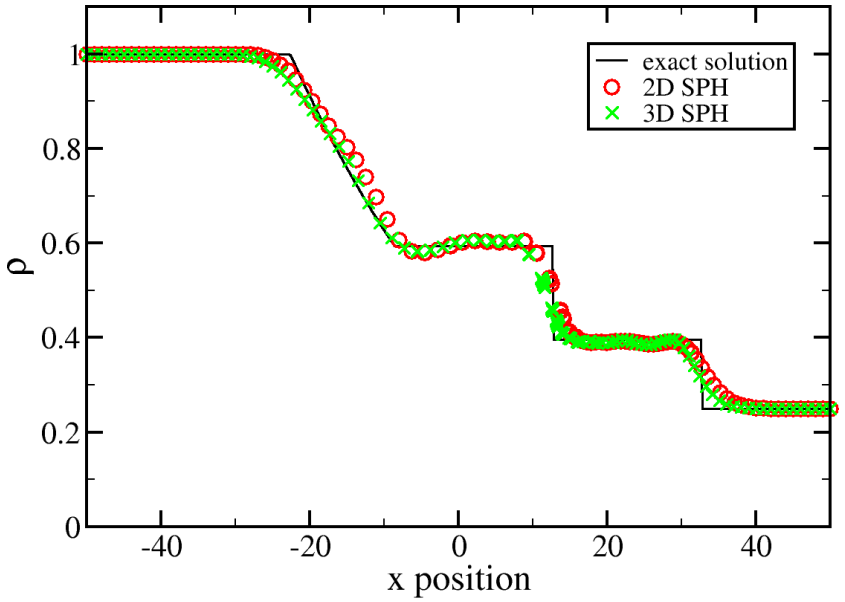

Figure 4: SPH and exact solution setup for the shock wave problem at $t = 20$.

5.2.2. Input script

This is the input script for the solution of the quasi 1d shock wave problem in 3 spatial dimension:

```
atom_style         meso
boundary           s p p

region             box block -100 150 -4 4 -4 4 units box
create_box         2 box
lattice            sc 1.0
create_atoms       1 box

region             left block EDGE 0.0 EDGE EDGE EDGE EDGE units box
region             right block 1 EDGE EDGE EDGE EDGE EDGE units box
set                region right type 2

mass               1 1
mass               2 0.25
set                type 1 meso_e 2.5 # internal energy corresponding to p=1, rho=1
set                type 2 meso_e 0.625 # internal energy corresponding to p=0.25, rho=0.25
set                type 1 meso_rho 1.0
set                type 2 meso_rho 0.25

pair_style         hybrid/overlay sph/rhosum 1 sph/idealgas
pair_coeff         * * sph/rhosum 4.0
pair_coeff         * * sph/idealgas 0.75 4.0

compute            rhoatom all meso_rho/atom
compute            ieatom all meso_e/atom
compute            emeso all reduce sum c_ieatom # total internal energy
compute            ke all ke
variable           etot equal c_ke+c_emeso # total energy

# dump positions and local density
dump               dump_id all custom 100 dump.3d id type x z y c_rhoatom
dump_modify        dump_id first yes

neighbor           0.5 bin
neigh_modify       every 5 delay 0 check yes
thermo             10
thermo_style       custom step c_ke c_emeso v_etot
thermo_modify      norm no

fix                integration_fix all meso
fix                1 all setforce NULL 0.0 0.0 # treat as a quasi 1d problem
timestep           0.05
log                log.3d
run                400 # run for t=20
```

5.3. Collapse of a water column

This example shows a prototypical test for SPH: the collapse of a water column in a rectangular container with added obstacles [1].

Boundary conditions are realised as SPH particles, which are time-integrated only in the local density and internal energy, but remain stationary. These boundary conditions are shown in grey in Fig. 5. Water is modelled using Tait's EOS with $c = 10$ m/s, $\rho_0 = 1000$ kg/m^3, and a value for the artificial viscosity of $\alpha = 10$. In order to deal efficiently with the time integration, a variable timestep is used: δt is chosen such that the fastest particle may move no more than a distance of 0.0005 m, or 5/300 h. Additionally, a CFL criterion with $\delta t < 0.1h/c$ is employed.

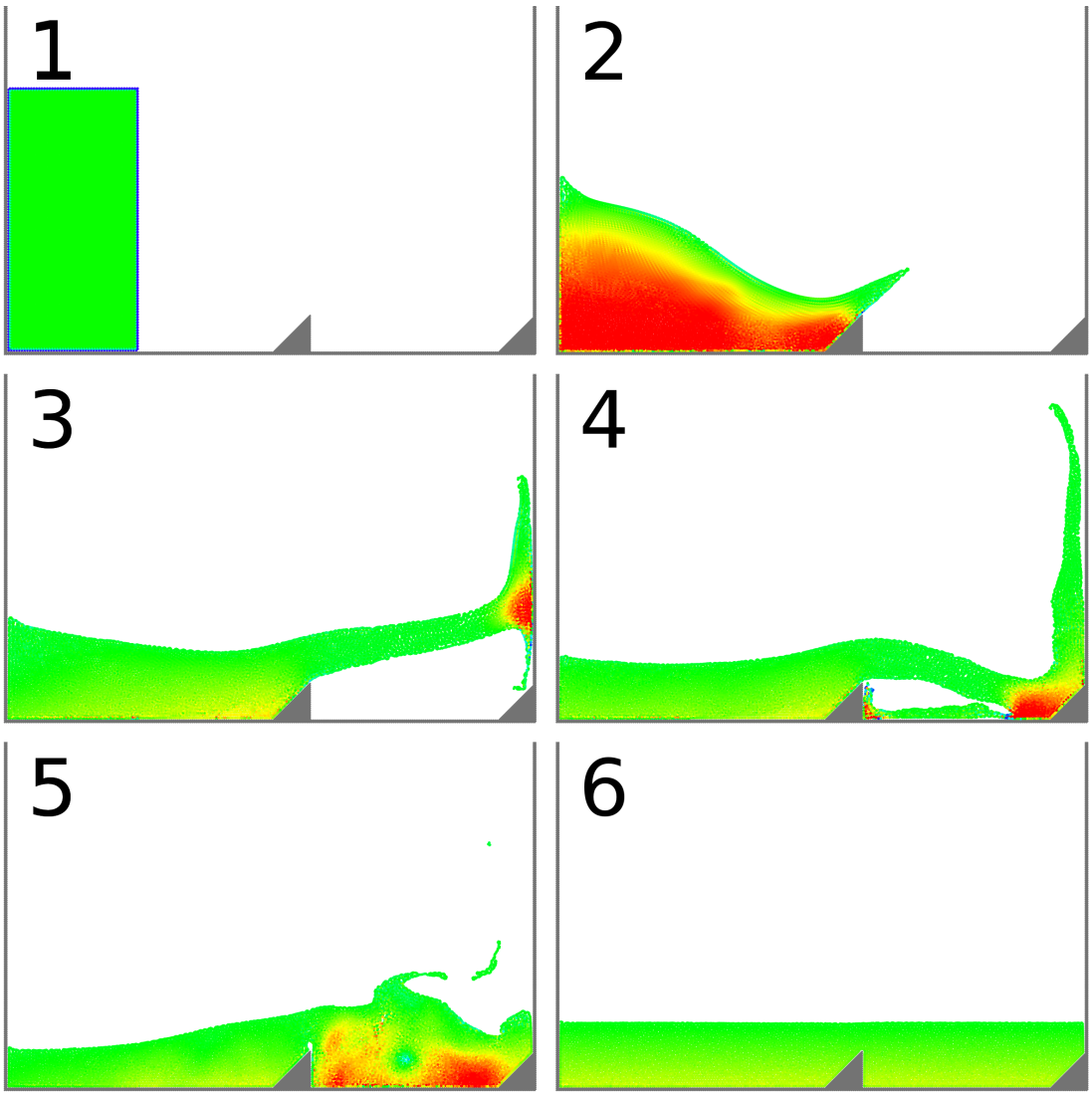

Figure 5: Simulation snapshots: colour represents mass density. The initial configuration (1) shows a water column with width, height = 1 m × 4 m embedded in a container of stationary particles (grey). Gravity imposes a downward force. As the water column collapses, a wave splashes over a wedge shaped obstacle and hits the right container wall (2-5). Snapshot (6) shows the equilibrium configuration after 9.4 s.

5.3.1. Input script

```
atom_style         meso
dimension          2
boundary           f f p
read_data          data.initial # read particle geometry from data file

variable           h equal 0.03 # SPH smoothing length
variable           c equal 10.0 # soundspeed for Tait's EOS
variable           dt equal 0.1*${h}/${c} # CFL criterion for upper limit of timestep
variable           nrun equal 15.0/${dt} # number of timesteps to run

group              bc    type 2 # assign group name "bc" to boundary particles (type 2)
group              water type 1 # assign group name "water" to water particles (type 1)

# use hybrid pairstyle which does density summation with cutoff ${h} every timestep (1)
pair_style         hybrid/overlay sph/rhosum 1 sph/taitwater
# use rho_0=1000, soundspeed ${c}, art. viscosity=1.0, smoothing length ${h}
pair_coeff         * * sph/taitwater 1000.0 ${c} 1.0 ${h}
pair_coeff         1 1 sph/rhosum ${h} # only do density summation for water

# add gravity. This fix also computes potential energy of mass in gravity field.
fix                gfix water gravity -9.81 vector 0 1 0

# computes local density & internal energy, sum total energy
compute            rho_peratom all meso_rho/atom
compute            e_peratom all meso_e/atom
compute            esph all reduce sum c_e_peratom
compute            ke all ke
variable           etot equal c_esph+c_ke+f_gfix

# use a variable timestep, such that any particle may travel only
# a maximum of 0.0005 distance units per timestep
fix                dtfix all dt/reset 1 NULL ${dt} 0.0005 units box

# time-integrate position, velocities, internal energy and density of water particles
fix                integrate_water_fix water meso

# time-integrate only internal energy and density of boundary particles
fix                integrate_bc_fix bc meso/stationary
dump               dump_id all custom 100 dump.lammpstrj id type xs ys zs\
                   c_rho_peratom c_e_peratom fx fy
dump_modify        dump_id first yes
thermo             10
thermo_style       custom step ke c_esph v_etot f_gfix press f_dtfix[1] f_dtfix
thermo_modify      norm no

neigh_modify       every 5 delay 0 check no
variable           skin equal 0.3*${h}
neighbor           ${skin} bin # set Verlet list skin distance
run                ${nrun}
```

5.3.2. Results

The importance of a variable timestep is demonstrated in Fig. 6. If a fixed timestep were used instead, it would need to be set to the smallest value attained in that figure in order to achieve the same degree of energy conservation. In contrast, a variable timestep only reduces δt when it is needed, e.g. during the highly turbulent collapse of the column, and not during the comparatively well-ordered flow afterwards. Fig. 7 shows that small timesteps correspond to the initial simulation regime, when kinetic energy is converted into internal energy due to viscous dissipation.

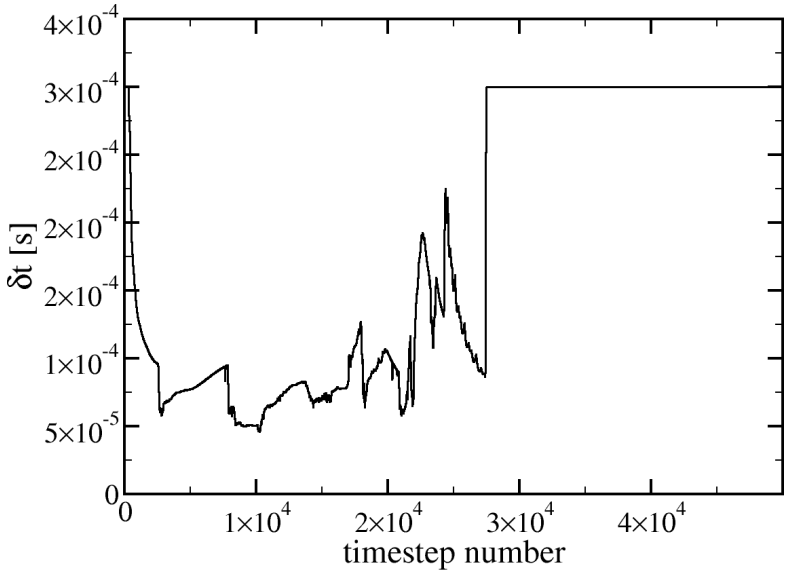

Figure 6: Variation of timestep size due to turbulent motion.

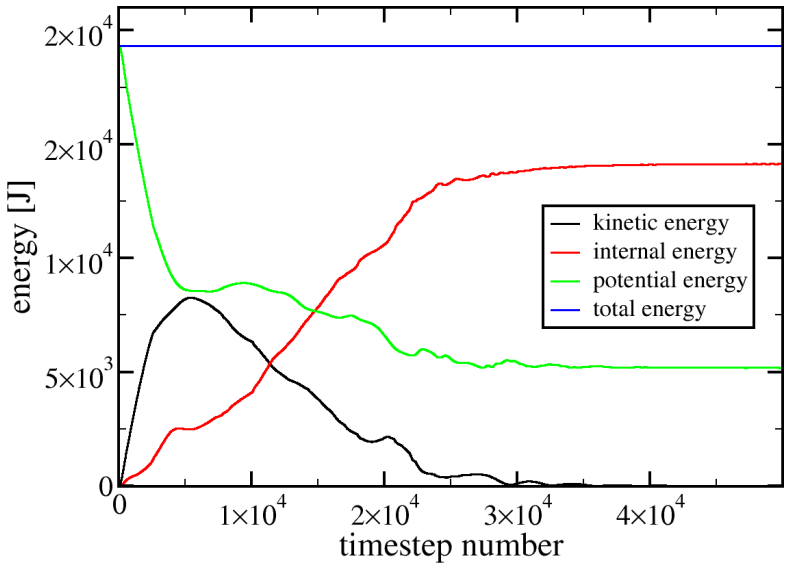

Figure 7: Distribution of total energy into kinetic, internal, and potential energy in the gravitational field. Total energy is conserved to 22 parts per million.

5.4. Shear cavity flow

The shear cavity flow is a standard test for a laminar flow profile. Here, we consider a 2D square lattice of fluid particles with the top edge moving at a constant speed of $10^{-3}m/s$. The other three edges are kept stationary. The driven fluid inside is represented by Tait's EOS with Morris' laminar flow viscosity. We use a kinematic viscosity of $\nu = 10^{-6}m^2/s$. This simulation produces a steady-state flow with a laminar vortex (see Fig. 8) after a few thousand cycles of equilibration. The velocity profile along the vertical centerline of the cavity agrees quite well with a Finite Difference solution (Fig. 9).

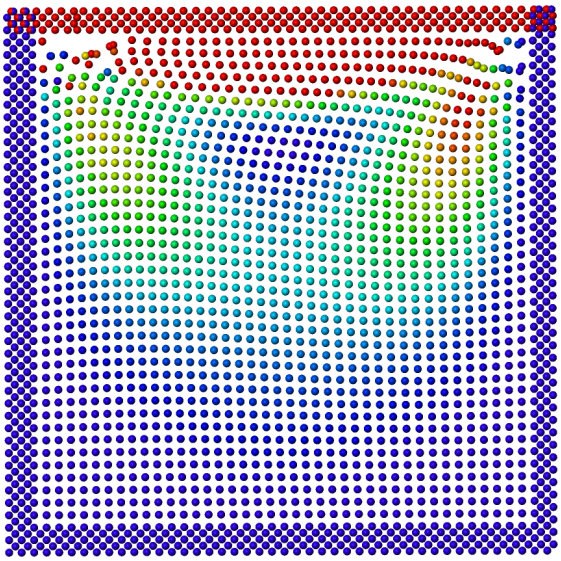

Figure 8: Simulations snapshot of the shear driven fluid filled cavity (upper boundary is moving to the right) in steady state. Particles are colored according to their kinetic energy.

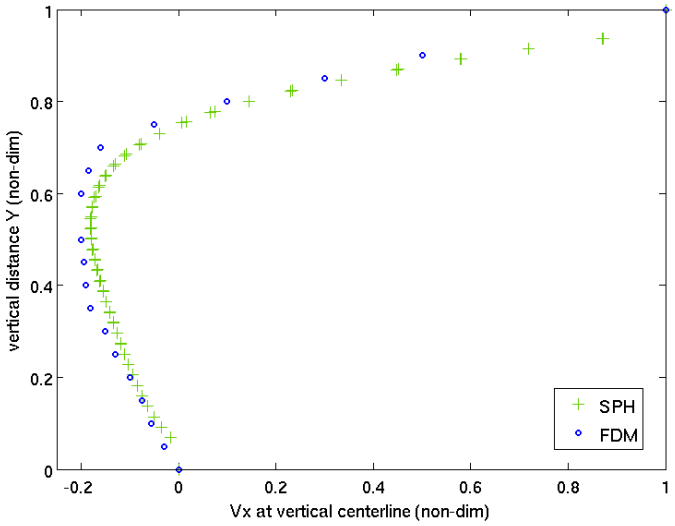

Figure 9: Non-dimensional horizontal particle velocities along the vertical centerline of the cavity. For comparison, a finite difference solution is also shown.

5.4.1. Input script

```
dimension          2
units              si
atom_style         meso

# create simulation box
region             box block -0.050e-3 1.044e-3 -0.05e-3 1.044e-3 -1.0e-6 1.0e-6 units box
create_box         3 box

# create fluid particles
region             fluid block 0.0001e-3 0.999e-3 0.0001e-3 0.999e-3 EDGE EDGE side in units
lattice            sq 0.025e-3
create_atoms       1 region fluid

# create bottom, left, and right wall
region             walls block 0.0001e-3 0.999e-3 0.0001e-3 EDGE EDGE EDGE side out units bo
lattice            sq2 0.025e-3
create_atoms       2 region walls

# create a driver strip of particles, which exerts shear forces on the fluid
region             driver block EDGE EDGE 0.999e-3 EDGE EDGE EDGE side in units box
create_atoms       3 region driver

group              fluid type 1
group              walls type 2
group              driver type 3
group              integrate_full union fluid driver

mass               3 2.0e-7
mass               2 2.0e-7
mass               1 4.0e-7
set                group all meso_rho 1000.0

# use Tait's EOS in combination with Morris' laminar viscosity.
# We set rho_0 = 1000 kg/m^3, c = 0.1 m/s, h = 6.5e-5 m.
# The dynamic viscosity is set to 1.0e-3 Pa s, corresponding to a kinematic viscosity of 1.0
pair_style         hybrid sph/taitwater/morris
pair_coeff         * *    sph/taitwater/morris 1000 0.1 1.0e-3 6.5e-5
pair_coeff         2 3    none # exclude interaction between walls and shear driver

compute            rho_peratom all meso_rho/atom
compute            e_peratom all meso_e/atom
compute            ke_peratom all ke/atom
compute            esph all reduce sum c_e_peratom
compute            ke all ke
variable           etot equal c_ke+c_esph

# assign a constant velocity to shear driver
velocity           driver set 0.001 0.0 0.0 units box
fix                freeze_fix driver setforce 0.0 0.0 0.0

# do full time integration for shear driver and fluid, but keep walls stationary
fix                integrate_fix_full integrate_full meso
fix                integrate_fix_stationary walls meso/stationary

dump               dump_id all custom 100 dump.lammpstrj id type xs ys zs vx vy c_rho_perato
```

```
56  dump_modify         dump_id first yes
57  thermo              100
58  thermo_style        custom step c_esph v_etot
59  thermo_modify       norm no
60
61  neighbor            3.0e-6 bin
62  timestep            5.0e-5
63  run                 4000
```

References

[1] M. Gomez-Gesteira, B. D. Rogers, R. A. Dalrymple, and A. J. C. Crespo. State-of-the-art of classical SPH for free-surface flows. *Journal of Hydraulic Research*, 48(extra):6—27, 2010.

[2] William G. Hoover. *Smooth Particle Applied Mechanics : The State of the Art*. World Scientific, Singapore, 2006.

[3] G.-R. Liu and M. B. Liu. *Smoothed Particle Hydrodynamics: a meshfree particle method*. World Scientific, 2003.

[4] Gonzalez L. M., Sanchez J. M., Macia F., and Souto-Iglesias A. Analysis of WCSPH laminar viscosity models. 4^{th} *international SPHERIC workshop*, Nantes, France, 2009.

[5] J.J Monaghan and R.A Gingold. Shock simulation by the particle method SPH. *Journal of Computational Physics*, 52(2):374–389, 1983.

[6] J.P. Morris, P.J. Fox, and Y. Zhu. Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics*, 136:214–226, 1997.

[7] Francis H. Ree. Analytic representation of thermodynamic data for the Lennard-Jones fluid. *The Journal of Chemical Physics*, 73(10):5401, 1980.

[8] 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):015012, 2010.