GPUSPH Theory Guide

version 5.0 — October 2016

Contents

1 Introduction

In the present document the SPH formulations realized in GPUSPH will be described. While this section is as complete as necessary we recommend to read the cited literature in order to gain a full understanding of the respective methods.
The basic premise is that GPUSPH approximates solutions to the Navier-Stokes equations which are given by

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \nabla \cdot \left[\nu(\boldsymbol{\nabla}\boldsymbol{v} + \boldsymbol{\nabla}\boldsymbol{v}^T)\right] + \boldsymbol{g}, \tag{1}$$

where $\boldsymbol{v}$ is the velocity, ρ the density, p the pressure, ν the kinematic viscosity and $\boldsymbol{g}$ an external force. These equations are coupled with the continuity equation

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\rho \nabla \cdot \boldsymbol{v}. \tag{2}$$

In order to close the system of equations a weakly-compressible formulation is chosen which uses the Cole Equation of State given by

$$p = \frac{c_0 \rho_0}{\xi}\left[\left(\frac{\rho}{\rho_0}\right)^{\xi} - 1\right], \tag{3}$$

where c_0 is the speed of sound, ρ_0 the reference density and ξ is the polytropic index, equal to 7 in the case of water.

2 The SPH approximation

At the heart of the SPH method is an interpolation that defines a physical quantity at a certain position. This interpolation consists in two steps: a kernel approximation followed by a particle approximation.

2.1 Kernel approximation

Let f be a field then

$$f(\boldsymbol{r}) = \int f(\boldsymbol{r'})\delta(\|\boldsymbol{r} - \boldsymbol{r'}|)\mathrm{d}\boldsymbol{r'}, \tag{4}$$

where δ is the Dirac delta distribution. The continuous SPH approximation can be written as

$$< f > (\boldsymbol{r}) = \int_\Omega f(\boldsymbol{r'})W(|\boldsymbol{r} - \boldsymbol{r'}|, h)\mathrm{d}\boldsymbol{r'}, \tag{5}$$

where the Dirac delta function was replaced by a weight or kernel function W, Ω is the computational domain and h is the smoothing length. For this approximation to be consistent, two conditions are required:

- $\lim_{h\to 0} W(.,h) = \delta(.)$, *i.e.* the kernel function converges to the delta function,
- $\int W(\boldsymbol{r}-\boldsymbol{r}',h)\mathrm{d}\boldsymbol{r}' = 1$, *i.e.* the kernel function has unitary integral.

Additional constraints are imposed to the SPH kernels, in order to simplify the physical modelling:

- $W(\boldsymbol{r}-\boldsymbol{r}',h) = W(\boldsymbol{r}'-\boldsymbol{r},h)$, *i.e.* the kernel is symmetric,
- the kernel has compact support.

From now one we will assume that the kernel fulfill this additional properties.

2.2 Particle approximation

The kernel approximation of Eq. (5) does not yet contain any spatial discretization. The spatial discretization is based on a the subdivision of the simulation domain using particles with each particle representing a small volume of the simulation domain. Each particle carry his one physical quantities (mass, density, velocity, ...). SPH being a purely Lagrangian method there is no need for any connectivity between particles.
For a given particle a, $\boldsymbol{r}_a$ will denote the particle position and f_a the value of f at the particle.
Now that the space has been discretized we can apply the particle approximation to the kernel one : approximating Eq. (5) by the following sum

$$[f](\boldsymbol{r}_a) = \sum_{b\in\mathcal{F}} \frac{m_b}{\rho_b} f_b W_{ab}, \tag{6}$$

where $\mathcal{F}$ is the set of all particles, m the mass and $W_{ab} = W(|\boldsymbol{r}_a - \boldsymbol{r}_b|/h, h)$. It should be noted that $[f] \neq f$.

3 SPH kernels

Currently GPUSPH features four different kernels.

- `CUBICSPLINE`: the cubic spline kernel

- `QUADRATIC`: the quadratic splain kernel
- `WENDLAND`: the quintic Wendland kernel
- `GAUSSIAN`: the Gaussian kernel

The formulae for the different kernels and their derivatives are given below. The variable q is defined by:

$$q = \frac{|\boldsymbol{r} - \boldsymbol{r}'|}{h} \tag{7}$$

The B-splines kernels are widely used in the SPH literature. The quadratic spline is defined, in three-dimensions, by:

$$W(q,h) = \frac{\alpha_2}{h^3} f_2(q) \tag{8}$$

with:

$$f_2(q) = \begin{cases} \frac{1}{4}q^2 - q + 1 & 0 \leq q \leq 2 \\ 0 & 2 < q \end{cases} \tag{9}$$

where $\alpha_2 = \frac{15}{16\pi}$ and its derivative by

$$f_2'(q) = \begin{cases} \frac{1}{2}q - 1 & 0 \leq q \leq 2 \\ 0 & 2 < q \end{cases} \tag{10}$$

The cubic spline is defined, in three-dimensions, by:

$$W(q,h) = \frac{\alpha_3}{h^3} f_3(q) \tag{11}$$

with:

$$f_3(q) = \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3 & 0 \leq q \leq 1 \\ \frac{1}{4}(2-q)^3 & 1 \leq q \leq 2 \\ 0 & 2 < q \end{cases} \tag{12}$$

where $\alpha_3 = 1/\pi$ and its derivative by

$$f_3'(q) = \begin{cases} -3q + \frac{9}{4}q^2 & 0 \leq q \leq 1 \\ -\frac{3}{4}(2-q)^2 & 1 \leq q \leq 2 \\ 0 & 2 < q \end{cases} \tag{13}$$

The Wendland kernel is given by

$$W(q,h) = \begin{cases} \frac{\alpha_W}{h^3}\left(1-\frac{q}{2}\right)^4(1+2q) & 0 \le q \le 2 \\ 0 & if 2 < q \end{cases} \tag{14}$$

where $\alpha_W = 21/(16\pi)$ and its derivative by

$$|\boldsymbol{\nabla} W(q,h)| = \begin{cases} \frac{-5\alpha_W}{h^5} q\left(1-\frac{q}{2}\right)^3 & 0 \le q \le 2 \\ 0 & 2 < q \end{cases} \tag{15}$$

The truncated Gaussian kernel is defined by

$$W(q,h) = \begin{cases} \frac{\alpha_G}{h^3}\left(e^{-q^2} - e^{-\left(\frac{3}{h}\right)^2}\right) & 0 \le q \le 3 \\ 0 & 3 < q \end{cases} \tag{16}$$

where $\alpha_G = \frac{1}{\pi\left(\sqrt{\pi}-36e^{-9}\right)}$.

$$|\boldsymbol{\nabla} W(q,h)| = \begin{cases} -2q\frac{\alpha_G}{h^3}\left(e^{-q^2} - e^{-\left(\frac{3}{h}\right)^2}\right) & 0 \le q \le 3 \\ 0 & 3 < q \end{cases} \tag{17}$$

3.1 First-order derivatives

In order to approximate derivatives the continuous SPH interpolation Eq. (5) is used as follows

$$< \boldsymbol{\nabla} f > (\boldsymbol{r}) = \int_\Omega \boldsymbol{\nabla} f(\boldsymbol{r'}) W(|\boldsymbol{r} - \boldsymbol{r'}|, h) \mathrm{d}\boldsymbol{r'} \tag{18}$$

Applying integration by parts yields

$$< \boldsymbol{\nabla} f > (\boldsymbol{r}) = -\int_\Omega f(\boldsymbol{r'}) \boldsymbol{\nabla}_{\boldsymbol{r'}} W(|\boldsymbol{r} - \boldsymbol{r'}|, h) \mathrm{d}\boldsymbol{r'} + \int_\Omega \boldsymbol{\nabla}(f(\boldsymbol{r'}) W(|\boldsymbol{r} - \boldsymbol{r'}|, h)) \mathrm{d}\boldsymbol{r'} \tag{19}$$

As W is symmetric the kernel is antisymmetric and thus

$$\boldsymbol{\nabla}_{\boldsymbol{r'}} W(|\boldsymbol{r} - \boldsymbol{r'}|, h) = -\boldsymbol{\nabla}_{\boldsymbol{r}} W(|\boldsymbol{r} - \boldsymbol{r'}|, h) \tag{20}$$

Furthermore, using Stokes theorem the last term in Eq. (19) can be rewritten as integral over the boundary of the computational domain, denoted by $\partial\Omega$ which yields

$$< \boldsymbol{\nabla} f > (\boldsymbol{r}) = \int_\Omega f(\boldsymbol{r'}) \boldsymbol{\nabla}_{\boldsymbol{r}} W(|\boldsymbol{r} - \boldsymbol{r'}|, h) \mathrm{d}\boldsymbol{r'} - \int_{\partial\Omega} (f(\boldsymbol{r'}) W(|\boldsymbol{r} - \boldsymbol{r'}|, h)) \boldsymbol{n}_{\boldsymbol{r'}} \mathrm{d}\boldsymbol{\Gamma}(\boldsymbol{r'}) \tag{21}$$

where $\boldsymbol{n}_{\boldsymbol{r}'}$ is the inward pointing normal. As a convention for the remainder of this document all normals will point inward the domain. If the domain Ω is unbounded then the last term of Eq. (21) will vanish due to the compact support of the kernel W. Thus, in a continuous sense the first derivative can be approximated as

$$< \boldsymbol{\nabla} f > (\boldsymbol{r}) = \int_{\Omega} f(\boldsymbol{r}')\boldsymbol{\nabla}_{\boldsymbol{r}} W(|\boldsymbol{r} - \boldsymbol{r}'|, h)\mathrm{d}\boldsymbol{r}' \tag{22}$$

The discretization in space again replaces the integral by a sum over all particles and so

$$[\boldsymbol{\nabla} f](\boldsymbol{r}_a) = \sum_{b \in \mathcal{F}} V_b f_b \boldsymbol{\nabla}_a W_{ab} \tag{23}$$

where $V_b = m_b/\rho_b$ is the volume of a particle b. It is common in SPH practice to write the gradient operator slightly different as it is used to compute the pressure gradient and thus Newtons third law of equal but opposite forces should be obeyed. This can be achieved by defining the SPH gradient as

$$\mathbf{G}_a(f) = [\boldsymbol{\nabla} f](\boldsymbol{r}_a) + [\boldsymbol{\nabla} f(\boldsymbol{r}_a)](\boldsymbol{r}_a) \tag{24}$$

where the latter derivative is that of a constant, thus equal to zero in the continuous framework. However, the SPH discretisation of the gradient as defined by equation (23) does not yield zero when applied to constants. The second term will thus have a non-zero contribution in this newly-defined SPH gradient.
The principle of energy conservation requires the gradient and divergence operators to be skew-adjoint, *i.e.*

$$< \mathbf{G}_a(f), \boldsymbol{F} >= - < f, D_a(\boldsymbol{F}) > \tag{25}$$

where $< ., . >$ is a scalar product on all particles. As a result of this constraint the divergence can be shown to be

$$D_a(\boldsymbol{F}) = [\nabla \cdot \boldsymbol{F}]_a - [\nabla \cdot \boldsymbol{F}_a]_a \tag{26}$$

This allows to write the gradient and divergence as

$$\mathbf{G}_a(f) = \sum_{b \in \mathcal{F}} V_b (f_b + f_a) \boldsymbol{\nabla}_a W_{ab} \tag{27}$$

and

$$D_a(\boldsymbol{F}) = \sum_{b \in \mathcal{F}} V_b (\boldsymbol{F}_b - \boldsymbol{F}_a) \cdot \boldsymbol{\nabla}_a W_{ab} \tag{28}$$

respectively.

3.2 Second-order derivatives

In theory second-order derivatives could simply be derived from first order ones, but that would cause second derivatives to appear which are numerically unstable. Thus, the preferred approach is to use a combination of a SPH first order derivative and a finite difference first order derivative.
The goal of this section is to derive a discretization for $\boldsymbol{\nabla}\cdot(f\boldsymbol{\nabla}\otimes\boldsymbol{F})$. To approximate the interior gradient, a Taylor Series of $\boldsymbol{F}$ can be used as follows

$$\boldsymbol{F}_b = \boldsymbol{F}_a - (\boldsymbol{\nabla}_a \otimes \boldsymbol{F})^T \cdot \boldsymbol{r}_{ab} + \mathcal{O}(\boldsymbol{r}_{ab}) \tag{29}$$

where $\boldsymbol{r}_{ab} = \boldsymbol{r}_a - \boldsymbol{r}_b$, a convention that will be used throughout this document also for quantities different from $\boldsymbol{r}$. The above can be rewritten to obtain an expression for the interior gradient

$$(\boldsymbol{\nabla}_a \otimes \boldsymbol{F})^T \cdot \frac{\boldsymbol{r}_{ab}}{|\boldsymbol{r}_{ab}|} \approx \frac{1}{|r_{ab}|}\boldsymbol{F}_{ab} \tag{30}$$

and similarly

$$(\boldsymbol{\nabla}_b \otimes \boldsymbol{F})^T \cdot \frac{\boldsymbol{r}_{ab}}{|\boldsymbol{r}_{ab}|} \approx \frac{1}{|r_{ab}|}\boldsymbol{F}_{ab} \tag{31}$$

As the kernel is radially symmetric

$$\boldsymbol{\nabla}_a W_{ab} = \frac{\boldsymbol{r}_{ab}}{|\boldsymbol{r}_{ab}|}|\boldsymbol{\nabla}_a W_{ab}| \tag{32}$$

The Divergence similar to the one given by Eq. (28) but with a + instead of the − yields

$$\boldsymbol{\nabla}\cdot(f\nabla\otimes\boldsymbol{F}) = \sum_{b\in\mathcal{F}} V_b(f_b\boldsymbol{\nabla}\otimes\boldsymbol{F} + f_a\boldsymbol{\nabla}\otimes\boldsymbol{F})\frac{\boldsymbol{r}_{ab}}{|\boldsymbol{r}_{ab}|}|\boldsymbol{\nabla}W_{ab}| \tag{33}$$

which, with Eqs. (30), (31) and (32), can be used to define the Laplacian

$$\mathbf{L}_a(f,\boldsymbol{F}) = \boldsymbol{\nabla}\cdot(f\boldsymbol{\nabla}\otimes\boldsymbol{F}) = \sum_{b\in\mathcal{F}} V_b(f_b+f_a)\boldsymbol{F}_{ab}\frac{1}{|\boldsymbol{r}_{ab}|}|\boldsymbol{\nabla}W_{ab}| \tag{34}$$

3.3 Discretization of the Navier-Stokes equations

With the definition of the three operators $\mathbf{G}$, D and $\mathbf{L}$ as given by Eqs. (27), (28) and (34), respectively, the Navier-Stokes equations (1) and (2) can be discretized in space as

$$\frac{\mathrm{d}\boldsymbol{v}_a}{\mathrm{d}t} = -\frac{1}{\rho_a}\mathbf{G}_a(p) + \mathbf{L}_a(\nu,\boldsymbol{v}) + \boldsymbol{g} \tag{35}$$

$$\frac{\mathrm{d}\rho_a}{\mathrm{d}t} = -\rho_a D_a(\boldsymbol{v}) \tag{36}$$

The final step to a full discretization is the implementation of a time integration scheme. GPUSPH currently features a predictor-corrector scheme that is given by

$$\begin{aligned} \boldsymbol{u}^{n+1/2} &= \boldsymbol{u}^n + \frac{\Delta t}{2} NS(\boldsymbol{u}^n) \\ \boldsymbol{u}^n &= \boldsymbol{u}^n + \Delta t NS(\boldsymbol{u}^{n+1/2}) \end{aligned} \tag{37}$$

where $\boldsymbol{u}$ is the state vector and NS are the right-hand sides of the discretized Navier-Stokes equations. Furthermore, the superscripts refer to the time instances. Written out in detail the full discretization reads

$$\begin{aligned} \boldsymbol{v}_a^{n+1/2} &= \boldsymbol{v}_a^n + \frac{\Delta t}{2}\left(-\frac{1}{\rho_a^n}\mathbf{G}_a(p^n) + \mathbf{L}_a(\nu^n, \boldsymbol{v}^n) + \boldsymbol{g}\right) \\ \boldsymbol{r}_a^{n+1/2} &= \boldsymbol{r}_a^n + \frac{\Delta t}{2}\boldsymbol{v}_a^n \\ \rho_a^{n+1/2} &= \rho_a^n - \frac{\Delta t}{2}\rho_a^n D_a(\boldsymbol{v}^n) \\ p_a^{n+1/2} &= \frac{c_0\rho_0}{\xi}\left[\left(\frac{\rho_a^{n+1/2}}{\rho_0}\right)^{\xi} - 1\right] \\ \boldsymbol{v}_a^{n+1} &= \boldsymbol{v}_a^n + \Delta t\left(-\frac{1}{\rho_a^{n+1/2}}\mathbf{G}_a(p^{n+1/2}) + \mathbf{L}_a(\nu^{n+1/2}, \boldsymbol{v}^{n+1/2}) + \boldsymbol{g}\right) \\ \boldsymbol{r}_a^{n+1} &= \boldsymbol{r}_a^n + \Delta t\, \boldsymbol{v}_a^{n+1/2} \\ \rho_a^{n+1} &= \rho_a^n - \Delta t\, \rho_a^{n+1/2} D_a(\boldsymbol{v}^{n+1/2}) \\ p_a^{n+1} &= \frac{c_0\rho_0}{\xi}\left[\left(\frac{\rho_a^{n+1}}{\rho_0}\right)^{\xi} - 1\right] \end{aligned} \tag{38}$$

3.3.1 Alternative continuity equation

The density can alternatively be computed by

$$[\rho]_a = \sum_{b\in\mathcal{F}} V_b \rho_b W_{ab} \tag{39}$$

which simplifies to

$$[\rho]_a = \sum_{b\in\mathcal{F}} m_b W_{ab} \tag{40}$$

To be consisten with the time-stepping and to preserve initial density values the following time dependent version is used

$$\rho_a^{n+1} = \rho_a^n - \left(\sum_{b\in\mathcal{F}} m_b W_{ab}\right)^n + \left(\sum_{b\in\mathcal{F}} m_b W_{ab}\right)^{n+1} \tag{41}$$

where the sums inside the brackets are with respect to the positions of the particles at the respective time step indicated by the superscript. It can be shown that this is equivalent to the continuity equation based on the divergence (see Vila (1999)). However, as the formulation depends only on the particle position it is numerically more stable.

4 Wall boundary conditions

Several different boundary conditions are available for the SPH method and a few selected ones are implemented in GPUSPH. Currently the following options are implemented

- `LJ_BOUNDARY`: Lennard–Jones boundary conditions
- `MK_BOUNDARY`: Monaghan–Kajtar boundary conditions
- `SA_BOUNDARY`: semi-analytical boundary conditions
- `DYN_BOUNDARY`: dynamic boundary conditions

In the following these boundary conditions and their formulations will be described in detail.

4.1 Lennard–Jones boundaries

With this type of boundaries, the walls are discretised with particles and repulsive forces are imposed between boundary particles and fluid particles (that move according to the SPH equations). The repulsive force employed derives from the Lennard-Jones potential (see Monaghan (2005)).
This method is computationally cheap, but may lead to spurious behaviours of the particles close to the walls. Indeed, none of the consistency issues related to boundaries are addressed by this method and although the impermeability of the walls is ensured, the SPH equations are inaccurately solved close to the boundaries. One effect is that the fluid does not remain still near the walls in a hydrostatic case.

4.2 Dynamic boundaries

Section under construction.

4.3 Semi-analytical boundaries

The semi-analytical wall boundary conditions developed by Ferrand et al. (2012) have shown promising results in the simulation of flows with complex boundaries using the Smoothed Particle Hydrodynamics (SPH) method. Recent efforts have pushed these boundary towards practical applications (see Mayrhofer et al. (2014); Leroy et al. (2014)). While the accuracy of these boundary conditions is outstanding, one of their downsides is their comparably high computational cost. The domain Ω

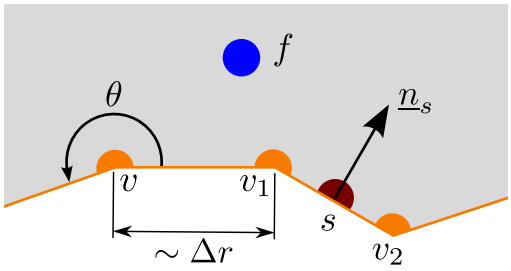

Figure 1: Different particle types

which is the fluid domain is discretized using three different sets of particles:

1. $f \in \mathcal{F}$: the fluid particles,
2. $p \in \mathcal{P}$: the vertex particles,
3. $s \in \mathcal{S}$: the boundary elements.

The boundary segments are triangles, in the 3-D case, that are located at the boundary $\partial\Omega$ of the domain. At each vertex of such a boundary elements is a vertex particle that has a mass that is related to the boundary shape as shown in Fig. 1. These particles, in a finite volume sense, represent the near-wall cells and are moving only if the solid wall is. The fluid particles on the other hand are typical SPH particles that move in a Lagrangian fashion and occupy Ω. The union of fluid and vertex particles will be denoted with $\mathcal{P}$.

The SPH interpolation in the context of the semi-analytical boundary conditions is given by

$$[f]_a = \frac{1}{\gamma_a}\sum_{b\in\mathcal{P}} V_b f_b W_{ab} \tag{42}$$

where γ_a is the kernel renormalization parameter defined as

$$\gamma_a = \int_\Omega W(|\boldsymbol{r} - \boldsymbol{r_b}|) \tag{43}$$

which is computed following the idea by Violeau et al. (2014). However, instead of analytically evaluating the integral over the boundary element, a 5th-order quadrature rule is used. Note that currently this is implemented only for the Wendland kernel and so the semi-analytical boundary conditions can only be used using this kernel.
The differential operators gradient and divergence are given as

$$\mathbf{G}_a(p) = \frac{1}{\gamma_a}\sum_{b\in\mathcal{P}} V_b(p_a + p_b)\boldsymbol{\nabla} W_{ab} - \frac{1}{\gamma_a}\sum_{s\in\mathcal{S}}(p_a + p_s)\boldsymbol{\nabla}\gamma_{as} \tag{44}$$

$$D_a(\boldsymbol{v}) = \frac{1}{\gamma_a}\sum_{b\in\mathcal{P}} V_b(\boldsymbol{v}_b - \boldsymbol{v}_a)\cdot\boldsymbol{\nabla} W_{ab} - \frac{1}{\gamma_a}\sum_{s\in\mathcal{S}}(\boldsymbol{v}_s - \boldsymbol{v}_a)\cdot\boldsymbol{\nabla}\gamma_{as} \tag{45}$$

respectively. $\boldsymbol{\nabla}\gamma_{as}$ is the surface integral of the kernel on a triangle s which is solved using the same 5th-order quadrature rule. The derivation of these operators can be found in Ferrand et al. (2012). The basic idea is to not drop the surface integral term in Eq. (21) but instead replace it with a discrete approximation, *i.e.* the sum over the segments. The Laplacian operator, required to discretize the viscous term of the momentum equation, is given by

$$\mathbf{L}_a(\nu, \boldsymbol{v}) = \frac{1}{\gamma_a}\sum_{b\in\mathcal{P}} m_b\left(\frac{\nu_a}{\rho_b} + \frac{\nu_b}{\rho_a}\right)\frac{\boldsymbol{v}_{ab}}{|\boldsymbol{r}_{ab}|^2}\boldsymbol{r}_{ab}\cdot\boldsymbol{\nabla} W_{ab} - \frac{2\nu_a\boldsymbol{v}_a}{\gamma_a}\sum_{s\in\mathcal{S}}\frac{|\boldsymbol{\nabla}\gamma_{as}|}{\delta r_{as}} \tag{46}$$

where the laminar shear stress was used in the boundary term in Ferrand et al. (2012), $\boldsymbol{r}_{ab} = \boldsymbol{r}_a - \boldsymbol{r}_b$ and $\delta r_{as} = \max(\boldsymbol{r}_{as}\cdot\boldsymbol{n}_s, \Delta r)$. Finally the Dirichlet boundary condition $\boldsymbol{v} = 0$ is applied at no-slip boundaries by imposing $\boldsymbol{v}_a = 0\ \forall a \in \mathcal{V}\cup\mathcal{S}$. The Neumann boundary condition $\partial\rho/\partial\boldsymbol{n} = 0$ is imposed using the approximation

$$\rho_a = \frac{1}{\alpha_a}\sum_{b\in\mathcal{F}} m_b W_{ab} \tag{47}$$

with

$$\alpha_a = \sum_{b\in\mathcal{F}} V_b W_{ab} \tag{48}$$

for all vertex particles and boundary elements (for details see Mayrhofer et al. (2013b)).
The alternative continuity equation can also be extended in order to take boundaries into account via

$$\rho_a^{n+1} = \frac{1}{\gamma_a^{n+1}} \left[\gamma_a^n \rho_a^n - \left(\sum_{b \in \mathcal{F}} m_b W_{ab} \right)^n + \left(\sum_{b \in \mathcal{F}} m_b W_{ab} \right)^{n+1} \right] \tag{49}$$

5 Corrections

5.1 Velocity smoothing: XSPH

Under construction

5.2 Sheppard filtering on density

Considering that

$$\sum_{b \in \mathcal{F}} \frac{m_b}{\rho_b} W_{ab} \neq 1$$

the standard SPH summation cannot achieve zeroth order consistency. We can restore zeroth order consistency by using

$$\widetilde{W_{ab}} = \frac{W_{ab}}{\sum\limits_{b \in \mathcal{F}} \frac{m_b}{\rho_b} W_{ab}}$$

instead of W.
Applied to density summation this lead to the Sheppard correction:

$$\rho_a^* = \frac{\sum\limits_{b \in \mathcal{F}} m_b W_{ab}}{\sum\limits_{b \in \mathcal{F}} \frac{m_b}{\rho_b} W_{ab}} \tag{50}$$

In GPUSPH this correction can be periodically enabling Sheppard filtering. **Note**: Sheppard correction leads to a volume increase. Thus this correction should not be used in for long term simulation. In the later it's preferable to use one of stabilizing method described in the next section.

5.3 MLS filtering on density

Under construction

6 Stabilizing methods

Due to the collocated nature of the SPH method it is unavoidable that spurious pressure modes are created. In order to prevent the numerical solution from becoming unstable several stabilizing methods can be employed.

6.1 Ferrari

The Ferrari correction is based on the work by Ferrari et al. (2009) and modifies the continuity equation with an additional term given by

$$\frac{\mathrm{d}\rho_a}{\mathrm{d}t} = -\rho_a D_a(\boldsymbol{v}) + \eta_F \sum_{b\in\mathcal{F}} V_b c_{a,b} \frac{\boldsymbol{r}_{ab}}{r_{ab}} \rho_{ab} \cdot \boldsymbol{\nabla} W_{ab} \tag{51}$$

where

$$c_{a,b} = \max(c_a, c_b) \tag{52}$$

and

$$c_a = c_0 \sqrt{\left(\frac{\rho_a}{\rho_0}\right)^{\xi-1}} \tag{53}$$

Note that in the case of the semi-analytical boundaries the formulation is the same and not boundary term needs to be added.
In certain flows (*e.g.* direct numerical simulation of turbulent flow, see Mayrhofer et al. (2013a)) the induced diffusivity can be too high. A more appropriate damping can be introduced by using a damping coefficient η_F different from one. Mayrhofer et al. (2013b) have shown that it should be chosen according to

$$\eta_F = \frac{L}{10^3 \Delta r} \tag{54}$$

where L is a typical length-scale of the flow. In GPUSPH L can be set directly using the `ferrariLengthScale` variable.

6.2 Rhie and Chow

Similar to the Ferrari correction is the correction in the spirit of Rhie and Chow. Again an additional term is added to the continuity equation. However, the density at time $n+1$ is computed first, denoted by $\widetilde{\rho}^{n+1}$, by solving the continuity equation and then a correction is applied to obatin ρ^{n+1}. This correction is given by

$$\rho_a^{n+1} = \eta_{RC} \widetilde{\rho}^{n+1} \left[\mathbf{L}_a \left(\frac{\Delta t}{\widetilde{\rho}}, \widetilde{\rho} \right) - \mathbf{L}_a(\Delta t, \boldsymbol{g} \cdot \boldsymbol{r}) \right] \tag{55}$$

C_μ	σ_k	σ_ϵ	C_{ϵ_1}	C_{ϵ_2}	κ
0.09	1.0	1.3	1.44	1.92	0.41

Table 1: Constants of the $k-\epsilon$ model

where η_{RC} governs the strength of this correction, but is usually equal to one.

7 Turbulence modelling

7.1 The $k-\epsilon$ model

The $k-\epsilon$ model is a Reynolds-Averaged Navier Stokes (RANS) model that uses two additional differential equations to close the equations Pope (2001). The RANS equations modify the momentum equations as follows:

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \boldsymbol{\nabla}\cdot((\nu+\nu_T)\boldsymbol{\nabla}\otimes\boldsymbol{v}) + \boldsymbol{g} \tag{56}$$

where ν_T is the turbulent viscosity. This in turn is given by

$$\nu_T = C_\mu \frac{k^2}{\epsilon} \tag{57}$$

where k is the turbulent kinetic energy and ϵ the turbulent dissipation. The constants, such as C_μ are summarized in Table 1. Both k and ϵ are given by two differential equations:

$$\frac{\mathrm{d}k}{\mathrm{d}t} = P - \epsilon + \nabla\cdot\left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\boldsymbol{\nabla} k\right] \tag{58}$$

and

$$\frac{\mathrm{d}\epsilon}{\mathrm{d}t} = \frac{\epsilon}{k}\left[C_{\epsilon_1}P - C_{\epsilon_2}\epsilon\right] + \nabla\cdot\left[\left(\nu + \frac{\nu_T}{\sigma_\epsilon}\right)\boldsymbol{\nabla} k\right] \tag{59}$$

where P is the production, given by

$$P = \min(\sqrt{C_\mu}kS, \nu_T S^2), \tag{60}$$

with S is the scalar mean rate-of-strain $S = \sqrt{2\boldsymbol{S}:\boldsymbol{S}}$. Finally, the boundary conditions for k and ϵ at solid walls need to be prescribed. For k a von Neumann boundary condition is imposed

$$\frac{\partial k}{\partial n} = 0. \tag{61}$$

ϵ close to the wall can be approximated via the theoretical relation

$$\epsilon = \frac{u_k^3}{\kappa z} \tag{62}$$

where $u_k = C_\mu^{1/4}\sqrt{k}$ and z is the normal distance to the wall. Clearly, this formula is singular at the wall and ϵ varies strongly when close to the wall. The flow is rather sensitive to this value and thus proper boundary conditions are essential. In the following section about the implementation of the $k-\epsilon$ model with the semi-analytical boundary model this can be taken into account.

7.1.1 Semi-analytical boundaries

The discretization presented in the present section is based on the work by Leroy et al. (2014) and for the exact derivation the interested reader is referred to this paper.
To discretize the equations for k and ϵ the production term requires the computations of the strain rate, which can be achieved by using the gradient operator. However, not the one of Eq. (44) is used, but instead the zero-order accurate version given by

$$\mathbf{G}_a^-(\boldsymbol{v}) = -\frac{1}{\gamma_a}\sum_{b\in\mathcal{P}} V_b \boldsymbol{v}_{ab} \otimes \boldsymbol{\nabla} W_{ab} + \frac{1}{\gamma_a}\sum_{s\in\mathcal{S}} \boldsymbol{v}_{ab} \otimes \boldsymbol{\nabla}\gamma_{as} \tag{63}$$

One important change compared to the laminar simulation is that the vertex particles now also carry a velocity which is evolved according to the viscous term, *i.e.*

$$\frac{\mathrm{d}\boldsymbol{v}_v}{\mathrm{d}t} = \mathbf{L}(\nu+\nu_T, \boldsymbol{v}) \tag{64}$$

The normal component of this velocity is set to zero to avoid particles penetrating the wall. The laplacian of the viscous term presented in Eq. (46) assumes a laminar flow profile and thus needs to be modified for turbulent flows in order to properly take the log law into account. This is achieved by setting the Laplacian to

$$\mathbf{L}_a(\nu, \boldsymbol{v}) = \frac{1}{\gamma_a}\sum_{b\in\mathcal{P}} m_b \left(\frac{\nu_a}{\rho_b} + \frac{\nu_b}{\rho_a}\right) \frac{\boldsymbol{v}_{ab}}{|\boldsymbol{r}_{ab}|^2} \boldsymbol{r}_{ab}\cdot\boldsymbol{\nabla} W_{ab} - \frac{2}{\gamma_a}\sum_{s\in\mathcal{S}} u_{*,as}^2 \boldsymbol{t}_{as} |\boldsymbol{\nabla}\gamma_{as}| \tag{65}$$

where $u_{*,as}$ is the friction velocity at the wall seen by particle a computed iteratively from the implicit equation

$$\frac{\boldsymbol{v}_{as}\cdot\boldsymbol{t}_{as}}{u_{*,as}} = \frac{1}{\kappa}\ln\left(\frac{\delta r_{as} u_{*,as}}{\nu}\right) + 5.2 \tag{66}$$

where

$$\boldsymbol{t}_{as} = \frac{\boldsymbol{v}_{as} - (\boldsymbol{v}_{as} \cdot \boldsymbol{n}_s)\boldsymbol{n}_s}{|\boldsymbol{v}_{as} - (\boldsymbol{v}_{as} \cdot \boldsymbol{n}_s)\boldsymbol{n}_s|} \tag{67}$$

The other term in the governing equations for k and ϵ is the Laplacian which are discretized as follows:

$$\mathbf{L}_a(\nu + \frac{\nu_T}{\sigma_k}, k) = \frac{1}{\gamma_a} \sum_{b \in \mathcal{P}} V_b \left(2\nu + \frac{\nu_{T,a} + \nu_{T,b}}{\sigma_k}\right) \frac{k_{ab}}{r_{ab}^2} \boldsymbol{r}_{ab} \cdot \boldsymbol{\nabla} W_{ab} \tag{68}$$

and

$$\mathbf{L}_a(\nu + \frac{\nu_T}{\sigma_\epsilon}, \epsilon) = \frac{1}{\gamma_a} \sum_{b \in \mathcal{P}} V_b \left(2\nu + \frac{\nu_{T,a} + \nu_{T,b}}{\sigma_\epsilon}\right) \frac{\epsilon_{ab}}{r_{ab}^2} \boldsymbol{r}_{ab} \cdot \boldsymbol{\nabla} W_{ab} + \frac{4C_\mu}{\sigma_\epsilon \gamma_a} \sum_{s \in \mathcal{S}} \frac{k_a^2}{\delta r_{as}} |\boldsymbol{\nabla} \gamma_{as}| \tag{69}$$

The compatible values on the boundary are given by

$$k_v = \frac{1}{\alpha_v} \sum_{b \in \mathcal{F}} V_b k_b W_{vb} \tag{70}$$

and

$$\epsilon_v = \frac{1}{\alpha_v} \sum_{b \in \mathcal{F}} V_b \left(\epsilon_b + \frac{4C_\mu^{3/4} k_b^{3/2}}{\kappa \delta r_{bv}}\right) W_{vb} \tag{71}$$

where α_v is given according to Eq. (48).

8 Open boundaries

GPUSPH currently implements two different types of open boundaries. One of them is for the dynamic boundary conditions and uses a buffer zone approach. The other can only be used in conjunction with the semi-analytical boundary conditions and uses flat open boundaries, which allows for in- and outflow to be at the same wall (*e.g.* waves).

8.1 Semi-analytical open boundaries

The open boundaries are based around the idea of vertex particles with varying mass. The mass of these vertex particles can change due to three different reasons:

- Flux through the boundary
- Fluid particle crossing boundary

- Creation of new fluid particle

An open boundary has either a prescribed velocity or a prescribed pressure and in the following these two options will be denoted with velocity and pressure boundary, respectively. The quantity that is not prescribed needs to be computed using Riemann invariants which is detailed in the following.

8.1.1 Riemann invariants for velocity boundaries

The imposed normal velocity at the open boundary is denoted with u_{ext}. The normal velocity inside the flow is extrapolated to the wall by

$$u_{int,a} = \frac{1}{\alpha_a} \sum_{b \in \mathcal{F}} V_b \boldsymbol{v}_b \cdot \boldsymbol{n}_a W_{ab} \tag{72}$$

Similary, the pressure is extrapolated according to

$$p_{int,a} = \frac{1}{\alpha_a} \sum_{b \in \mathcal{F}} V_b p_b W_{ab} \tag{73}$$

ρ_{int} can easily be computed using the inverse Equation of State (EOS^{-1}). The aim of this section will be to compute $p_{ext,a}$ from the extrapolated and imposed quantities. The main idea is that there is an internal and an external state with the interface at the open boundary. Due to the discontinuity of the internal and external fields and the assumption of a 1-D problem the Riemann problem is recovered and a solution is known that can be divided into three different states.
Let

$$\psi(\rho) = \frac{2c_0}{\xi - 1} \left(\frac{\rho}{\rho_0} \right)^{\frac{\xi-1}{2}} \text{ if } \xi > 1 \quad \text{or} \quad \psi(\rho) = c_0 \ln \left(\frac{\rho}{\rho_0} \right) \text{ if } \xi = 1 \tag{74}$$

and ψ^{-1} the respective inverse function. The three states yield the following densities

- Expansion wave:

$$\rho_{ext,e} = \psi^{-1}(\psi(\rho_{int}) + u_{ext} - u_{int}) \tag{75}$$

- Shock wave:

$$\rho_{ext,s} = EOS^{-1}(p_{int} + \rho_{int} u_{int}(u_{int} - u_{ext})) \tag{76}$$

- Contact discontinuity

$$\rho_{ext,c} = \rho_{int} \tag{77}$$

To decide which state needs to be considered the speed of sound as function of density needs to be defined as

$$c(\rho) = c_0 \left(\frac{\rho}{\rho_0}\right)^{\frac{\xi-1}{2}} \tag{78}$$

to be able to compute the celerities λ. They are given as

$$\begin{aligned} \lambda &= u_{int} + c(\rho_{int}) && (79)\\ \lambda_e &= u_{ext} + c(\rho_{ext,e}) && (80)\\ \lambda_s &= u_{ext} + c(\rho_{ext,s}) && (81) \end{aligned}$$

Based on these celerities the states occur according to

- $\lambda_e \leq \lambda \Rightarrow$ expansion wave
- $\lambda_s > \lambda \Rightarrow$ shock wave
- $\lambda_e > \lambda \geq \lambda_s \Rightarrow$ contact discontinuity

Depending on the computed state the pressure p_{ext} is set according to the corresponding density in Eqs. (75), (76) or (77).

8.1.2 Riemann invariants for pressure boundaries

Compared to the velocity boundary, this time u_{ext} needs to be computed and p_{ext} is imposed. Similar to the previous section three different fluxes, u_{ext}, can be computed according to the different states

- Expansion wave:
$$u_{ext,e} = u_{int} + \psi(\rho_{ext}) - \psi(\rho_{int}) \tag{82}$$
- Shock wave:
$$u_{ext,s} = u_{int} + \frac{p_{int} - p_{ext}}{\rho_{int} \max(u_{int}, 10^{-5} c_0)} \tag{83}$$
- Contact discontinuity:
$$u_{ext,c} = u_{int} \tag{84}$$

The celerities $\lambda_{\{e,s\}} = u_{ext,\{e,s\}} + c(\rho_{ext})$ can then be used equivalently as above to determine the appropriate state and thus to set u_{ext}.

8.1.3 Mass update

Assuming that both velocity u_{ext} and pressure p_{ext} are known at both segments and vertices of an open boundary the mass update of a vertex particle can be performed. Each segment has three vertices associated with it and similarly, each vertex has a defined set of segments that it is associated with it. The latter set will be denoted with $\mathcal{S}_v$ for a specific vertex v. The principal mass change comes from the flux through segments and reads

$$\widetilde{m}_v = m_v^n + \dot{m}_v \tag{85}$$

where

$$\dot{m}_v = \Delta t \sum_{s \in \mathcal{S}_v} \rho_{ext,s} u_{ext,s} A_s \beta_v(\boldsymbol{r}_s) \tag{86}$$

where A_s represents the area of the segment s and $\beta_v(\boldsymbol{r}_s)$ is the mass repartition factor that will be described below.
Next some mass clippings occur in the sequence listed below

- If no fluid particle is in the support of v and $\dot{m}_v < 0$ then $\widetilde{m}_v = 0$.
- Ensure that $|\widetilde{m}_v| < 2m_{ref}$, where $m_{ref} = \rho_0 \Delta r^3$ is the reference mass.
- If $\dot{m}_v < 0$ ensure that $|\widetilde{m}_v| < m_v^0$, where m_v^0 is the initial mass of the vertex v.

After these clippings have been made, a new fluid particle is created if $\widetilde{m}_v > \frac{1}{2} m_{ref}$. This new fluid particle has exactly the same properties as the vertex particle and a mass equal to the reference mass. Note that this can only happen in the corrector step of the time-stepping scheme. Further conditions are that $u_{ext,v}$ and $p_{ext,v}$ are both greater than zero. If a fluid particle is created than $\widetilde{m}_v$ has m_{ref} subtracted.
If a fluid particle a has crossed a segment $s \in \mathcal{S}_v$ then the mass of the fluid particle is redistributed to the vertex particles associated to that segment. If v is one such segment then

$$\widetilde{m}_v = \widetilde{m}_v + \beta_v(\boldsymbol{r}_a) m_{ref} \tag{87}$$

Finally the new mass of the vertex particle $m_v^{n+1} = \widetilde{m}_v$.

8.1.4 The corners

Vertex particles that are part of the open boundary but have associated segments that are not part of a boundary are labeled as corner vertices. These vertices have several special properties that are detailed below.
First, they do not change their mass and due to that also never create new fluid particles. If a corner vertex is part of a velocity boundary, then both its velocity

and pressure is set to that of the solid wall. If instead the corner vertex is part of a pressure boundary, then the velocity is set to that of the solid wall and the imposed pressure of the open boundary is used.

8.1.5 Modified continuity equation

The open boundaries are restricted to use with the alternative continuity equation given by Eq. (49). In the following $\mathcal{S}^o$ and $\mathcal{V}^o$ denote the segments and vertices respectively that are associated with open boundaries.

$$\begin{aligned} \rho_a^{n+1} &= \frac{1}{\gamma_a^{n+1}} \Bigg\{ \gamma_a^n \rho_a^n + \sum_{b \in \mathcal{P}} m_b^n (W_{ab}^{n+1} - W_{ab}^n) + \\ &\quad + \sum_{v \in \mathcal{V}^o} m_v^n \left[W_{av}^n - w \left(\boldsymbol{r}_{av}^n + \boldsymbol{\delta r}_v^o(\Delta t) \right) \right] \\ &\quad + \frac{\rho_a^n}{2} \sum_{s \in \mathcal{S}^o} \left[\boldsymbol{\nabla} \gamma_{as} \left(\boldsymbol{r}_{as}^n + \boldsymbol{\delta r}_s^o(\Delta t) \right) + \boldsymbol{\nabla} \gamma_{as}(\boldsymbol{r}_{as}^n) \right] \cdot \boldsymbol{\delta r}_s^o(\Delta t) \Bigg\} \end{aligned} \tag{88}$$

where

$$\boldsymbol{\delta r}_a^o(\Delta t) = \Delta t (\boldsymbol{u}_a^n + \boldsymbol{v}_a^n) \tag{89}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the Eulerian and Lagrangian velocity, respectively.

8.1.6 Time integration

The full time-stepping scheme including for open boundaries reads

$$
\begin{aligned}
\boldsymbol{v}_a^{n+1/2} &= \boldsymbol{v}_a^n + \frac{\Delta t}{2}\left(-\frac{1}{\rho_a^n}\mathbf{G}_a(p^n) + \mathbf{L}_a(\nu^n, \boldsymbol{v}^n) + \boldsymbol{g}\right) \qquad (90)\\
\boldsymbol{r}_a^{n+1/2} &= \boldsymbol{r}_a^n + \frac{\Delta t}{2}\boldsymbol{v}_a^n\\
\gamma_a^{n+1/2} &= \gamma_a^n + \frac{1}{2}(\boldsymbol{r}_a^{n+1/2} - \boldsymbol{r}_a^n)\cdot(\boldsymbol{\nabla}\gamma_a^n + \boldsymbol{\nabla}\gamma_a^{n+1/2})\\
\rho_a^{n+1/2} &= \frac{1}{\gamma_a^{n+1/2}}\Bigg\{\gamma_a^n\rho_a^n + \sum_{b\in\mathcal{P}} m_b^n(W_{ab}^{n+1/2} - W_{ab}^n) +\\
&\quad + \sum_{v\in\mathcal{V}^o} m_v^n\left[W_{av}^n - w\left(\boldsymbol{r}_{av}^n + \boldsymbol{\delta r}_v^o(\Delta t/2)\right)\right]\\
&\quad + \frac{\rho_a^n}{2}\sum_{s\in\mathcal{S}^o}\left[\boldsymbol{\nabla}\gamma_{as}\left(\boldsymbol{r}_{as}^n + \boldsymbol{\delta r}_s^o(\Delta t/2)\right) + \boldsymbol{\nabla}\gamma_{as}(\boldsymbol{r}_{as}^n)\right]\cdot\boldsymbol{\delta r}_s^o(\Delta t/2)\Bigg\}\\
p_a^{n+1/2} &= \frac{c_0\rho_0}{\xi}\left[\left(\frac{\rho_a^{n+1/2}}{\rho_0}\right)^{\xi} - 1\right]\\
&\quad \text{Boundary conditions \& mass update}\\
\boldsymbol{v}_a^{n+1} &= \boldsymbol{v}_a^n + \Delta t\left(-\frac{1}{\rho_a^{n+1/2}}\mathbf{G}_a(p^{n+1/2}) + \mathbf{L}_a(\nu^{n+1/2}, \boldsymbol{v}^{n+1/2}) + \boldsymbol{g}\right)\\
\boldsymbol{r}_a^{n+1} &= \boldsymbol{r}_a^n + \Delta t\,\boldsymbol{v}_a^{n+1/2}\\
\gamma_a^{n+1} &= \gamma_a^n + \frac{1}{2}(\boldsymbol{r}^{n+1} - \boldsymbol{r}_a^n)\cdot(\boldsymbol{\nabla}\gamma_a^n + \boldsymbol{\nabla}\gamma_a^{n+1})\\
\rho_a^{n+1} &= \frac{1}{\gamma_a^{n+1}}\Bigg\{\gamma_a^n\rho_a^n + \sum_{b\in\mathcal{P}} m_b^n(W_{ab}^{n+1} - W_{ab}^n) +\\
&\quad + \sum_{v\in\mathcal{V}^o} m_v^n\left[W_{av}^n - w\left(\boldsymbol{r}_{av}^n + \boldsymbol{\delta r}_v^o(\Delta t)\right)\right]\\
&\quad + \frac{\rho_a^n}{2}\sum_{s\in\mathcal{S}^o}\left[\boldsymbol{\nabla}\gamma_{as}\left(\boldsymbol{r}_{as}^n + \boldsymbol{\delta r}_s^o(\Delta t)\right) + \boldsymbol{\nabla}\gamma_{as}(\boldsymbol{r}_{as}^n)\right]\cdot\boldsymbol{\delta r}_s^o(\Delta t)\Bigg\}\\
p_a^{n+1} &= \frac{c_0\rho_0}{\xi}\left[\left(\frac{\rho_a^{n+1}}{\rho_0}\right)^{\xi} - 1\right]\\
&\quad \text{Boundary conditions \& mass update}\\
&\quad \text{Create and delete particles if required}
\end{aligned}
$$

9 Solid bodies

GPUSPH fully integrates moving and floating bodies. A moving body as an user prescribed by user while the movement of a floating body is controlled by the forces acting on it.

9.1 Reference frame and orientation

Under construction.

9.2 Moving bodies

Under construction

9.3 Floating bodies

All the floating bodies dynamics is solved using Chrono engine. Basically GPUSPH computes force and torque on each body and feed them to Chrono engine that integrate the movement.
Details under construction.

10 Granular rheology

More details on this model can be found in Ghaitanellis (2017) and Ghaïtanellis et al. (2018).

10.1 Context

The main difficulty in modelling granular materials is that they cannot be classified as solids or liquids. Indeed they can behave like both of them under slightly different conditions. That is the reason why rheological approach is often adopted to model granular flows. In particular, soil is usually treated as a viscoplastic material. Thus it has a yield stress τ_y under which no deformation occurs. When the yield stress is exceeded, the material starts to flow. In the present model, a single behaviour law is used to model the two mechanical behaviours. The granular material is modelled as a liquid that exhibits a very large viscosity when the strain rate vanishes. The solid

state is thus approached by a highly viscous state. Except for viscosity which is not constant, granular phase is thus treated similarly to any other liquid phase.

10.2 Modelling assumptions

In this work, the granular material, being dry or saturated, is treated as a continuum. Under this assumption, the continuous medium represents a mixture of grains and air, or grains and water. Therefore, we need to define equivalent properties of the continuous material from the physical properties of the granular material components. The continuum properties should also depend on the grain concentration. In this model, the grain concentration will be assumed to be constant and homogeneous within the mixture domain. Therefore, in practise, ϕ is simply the porosity of the granular material at initial time. In addition, the fluid surrounding the grains will be assumed to be at rest with respect to the granular material skeleton. Thus in the present work, we will only consider cases in which the material is initially totally dry (denoted by $^{\mathrm{dry}}$) or fully saturated (denoted by $^{\mathrm{sat}}$), and remains that way for ever. For the dry case, the mass of air in the pore is neglected, while for the saturated case, the equivalent density also depends on water density ρ_{w}:

$$\begin{cases} \rho_{\mathrm{eq}}^{\mathrm{dry}} = (1-\phi)\ \rho_{\mathrm{g}} \\ \rho_{\mathrm{eq}}^{\mathrm{sat}} = (1-\phi)\ \rho_{\mathrm{g}} + \phi\, \rho_{\mathrm{w}} \end{cases} \tag{91}$$

We assume here that ρ_{w} and ρ_{g} are constant. Secondly, neglecting the flow through the porous material also implies that the pore water pressure field is unknown. Therefore, another approximation is necessary and the pore water pressure will be assumed to be hydrostatic.

10.3 Yield stress and effective pressure

There are many yield criteria that are adapted to the different kinds of viscoplastic materials. Although the Mohr-Coulomb criterion is widely used in geotechnical engineering, its implementation raises numerical issues because the corresponding yield surface is an irregular hexagonal pyramid. Thus, the Drucker-Prager yield criterion is usually used as a smooth alternative. It is a pressure-dependent model that can be expressed in terms of internal friction angle ψ for non-cohesive soils as:

$$\tau_{\mathrm{y}} = \frac{2\sqrt{3}\sin\psi}{3-\sin\psi}\, p_{\mathrm{eff}} \tag{92}$$

where p_{eff} is the effective pressure, that is to say the normal stress that the grains actually exert on each other. Considering a situation where the stress configuration is close to the failure, almost no motion occurs within the material. Therefore, the isotropic stress (*i.e.* the total pressure p_{tot}) is solely due to pressure derived from the weight of the column of material and interstitial fluid above a specified level. Consequently, denoting z_{i} the soil-water interface vertical position, z_{fs} the free-surface vertical position and z the vertical position of the point of interest, the total pressure reads:

$$p_{\text{tot}} = \rho_{\text{w}} g \left(z_{\text{fs}} - z_{\text{i}}\right) + \rho_{\text{eq}}^{\text{sat}} g \left(z_{\text{i}} - z\right) \tag{93}$$

Moreover, Terzaghi's principle (Terzaghi, 1936) states that the effective pressure depends on the total pressure p_{tot} and the pore water pressure p_{pw} according to the following relation:

$$p_{\text{eff}} = p_{\text{tot}} - p_{\text{pw}} \tag{94}$$

Assuming no porous flow inside the soil, the pore water pressure reads:

$$p_{\text{pw}} = \rho_{\text{w}} g \left(z_{\text{fs}} - z\right) \tag{95}$$

Finally, effective pressure can be obtained using (94):

$$p_{\text{eff}} = g \left(z_{\text{i}} - z\right) \left(\rho_{\text{eq}}^{\text{sat}} - \rho_{\text{w}}\right) \tag{96}$$

Nevertheless, getting p_{eff} from (96) requires to detect soil-water interface beforehand and can lead to inaccurate results when interface is highly deformed. Recall that ρ_{w} and $\rho_{\text{eq}}^{\text{sat}}$ are constant. In addition, z_{i} is the function that gives the mixture-water interface above the point of interest, so it does not depend on the vertical coordinate. As long as the water-mixture interface is not too much deformed, we can besides assume that the interface varies linearly in the horizontal directions in the vicinity of the point of interest (this is the strongest assumption done in this reasoning). Thus, taking the Laplacian of (96) yields:

$$\Delta p_{\text{eff}} = 0 \tag{97}$$

taking the Laplacian of (96) with suitable boundary conditions we get the following system:

$$\begin{cases} \Delta p_{\text{eff}} \left(\boldsymbol{r}\right) &= 0 & \text{for} \quad \boldsymbol{r} \in \Omega \\ p_{\text{eff}} \left(\boldsymbol{r}\right) &= 0 & \text{for} \quad \boldsymbol{r} \in \mathcal{I} \\ \dfrac{\partial p_{\text{eff}}}{\partial \boldsymbol{n}} \left(\boldsymbol{r}\right) &= \boldsymbol{g} \cdot \boldsymbol{n} & \text{for} \quad \boldsymbol{r} \in \mathcal{S} \end{cases} \tag{98}$$

with Ω the simulation domain, $\mathcal{S}$ the solid boundaries, $\boldsymbol{n}$ the inward boundary normal and $\mathcal{I}$ the boundary corresponding to soil free-surface and soil-water interface.
In the present model p_{eff} is only used to compute the yield stress. It cannot be used in the momentum conservation equation, because (98) is valid when the total pressure is approximately lithostatic, which is not true in yielded regions. Thus, the total pressure calculated with the state equation is used to compute the pressure gradient in the momentum conservation equation. This amounts to assume that the pore water pressure gradient can be neglected compared to the effective pressure gradient:

$$\boldsymbol{\nabla} p_{\text{eff}} = \boldsymbol{\nabla} p_{\text{tot}} - \boldsymbol{\nabla} p_{\text{pw}} \approx \boldsymbol{\nabla} p_{\text{tot}} \tag{99}$$

Note that it would be possible to eliminate this approximation observing that (98) can easily be adapted to compute the hydrostatic pore water pressure. The pressure gradient in the momentum conservation equation could thus be computed from $p_{\text{tot}} - p_{\text{pw}}$. This should be tested in future work.

10.4 Effective viscosity

The granular material is modelled as a shear thinning liquid. Thus the viscous fluid behaviour law is used to model the shear stress:

$$\boldsymbol{\tau} = 2\,\eta_{\text{eff}} \boldsymbol{S} \tag{100}$$

where the effective viscosity is thus defined by:

$$\eta_{\text{eff}} = \frac{\tau_{\text{y}}}{S} \tag{101}$$

where τ_{y} is the yield stress calculated according to (92). Therefore, similarly to the $\mu(I)$ rheology (Jop et al., 2006; Revil-Baudard and Chauchat, 2013), the effective viscosity is related to the friction within the granular material, *i.e.* it relates the shear stress to the isotropic stress and a friction coefficient μ.

$$\tau(\psi, p_{\text{eff}}) = \sqrt{\sum_{i,j} \frac{\tau_{ij}\tau_{ij}}{2}} = \eta_{\text{eff}} S = \tau_{\text{y}}(\psi, p_{\text{eff}}) = \frac{2\sqrt{3}\sin\psi}{3 - \sin\psi} p_{\text{eff}} = \mu p_{\text{eff}} \tag{102}$$

with

$$\mu = \frac{2\sqrt{3}\sin\psi}{3 - \sin\psi} \tag{103}$$

Here the friction coefficient is constant in time since the internal friction ψ is a constant parameter in our model. However, experimentally, the friction coefficient

increases for large strain rate. Therefore, in future work, $\mu(I)$ rheology should be implemented to take this phenomenon into account.

10.5 SPH implementation

To compute effective viscosity from (101), one needs to compute the shear rate and the effective pressure.

10.5.1 Shear rate computation

The shear rate requires the computation of the velocity gradient. In GPUSPH, the velocity gradient is computed as follows, regardless of the chosen formulation:

$$(\boldsymbol{\nabla u})_a = -\sum_{b\in\mathcal{F}} V_b(\boldsymbol{u}_a - \boldsymbol{u}_b) \otimes \boldsymbol{\nabla} w_{ab} \tag{104}$$

This formula is based of the equality:

$$\nabla A = \nabla A - A\nabla 1 \tag{105}$$

which does not involve density gradient, and thus is compatible is multi-phase flows. This velocity gradient is consistent with Colagrossi and Landrini multi-phase formulation (Colagrossi and Landrini, 2003). The SA form reads:

$$(\boldsymbol{\nabla u})_a = -\frac{1}{\gamma_a}\sum_{b\in(\mathcal{F}\cup\mathcal{V})} V_b(\boldsymbol{u}_a - \boldsymbol{u}_b) \otimes \boldsymbol{\nabla} w_{ab} + \frac{1}{\gamma_a}\sum_{s\in\mathcal{S}}(\boldsymbol{u}_a - \boldsymbol{u}_b) \otimes \boldsymbol{\nabla}\gamma_{as} \tag{106}$$

Note that, to be consistent with SPH_F1 formulation, one should use:

$$(\boldsymbol{\nabla u})_a = \frac{-1}{\rho_a}\sum_{b\in\mathcal{F}} m_b(\boldsymbol{u}_a - \boldsymbol{u}_b) \otimes \boldsymbol{\nabla} w_{ab} \tag{107}$$

This formula is based on the equality:

$$\nabla A = \frac{1}{\rho}\nabla(\rho A) - \frac{A}{\rho}\nabla\rho \tag{108}$$

Similarly to all SPH_F1 operators, the velocity gradient involves a density gradient, which is not defined at the interface of two fluids of different densities. Thus, it cannot be used for multi-phase flows.

To be consistent with SPH_HA formulation (Hu and Adams, 2006), the following formula should be used:

$$(\boldsymbol{\nabla u})_a = -V_a \sum_{b \in \mathcal{F}} (\boldsymbol{u}_a - \boldsymbol{u}_b) \otimes \boldsymbol{\nabla} w_{ab} \tag{109}$$

This formula is based of the equality:

$$\nabla A = V \nabla \left(\frac{A}{V} \right) - V A \nabla \frac{1}{V} \tag{110}$$

which does not involve density gradient.

10.5.2 Effective pressure solver

A Jacobi solver is used to solve the linear system (98). The solver is described in the framework of semi-analytical boundary conditions. The following notations are used:

- $\mathcal{M}$ set of free particle of granular material
- $\mathcal{I} \subset \mathcal{M}$ particles at the interface/free-surface
- $\mathcal{V}$ set of vertex particles
- $\mathcal{S}$ set of boundary elements
- Effective pressure is denoted p for the sake of clarity

For dynamic particles and Lennard-Jones boundary conditions, boundary particles are treated like the SA vertex particles, and the SA boundary elements terms are just ignored. Enforce the Dirichlet condition on interface/free-surface particles:

$$\forall a \in \mathcal{I} \qquad p_a = \Delta \rho g \delta r$$

Enforce the Neumann condition on SA vertex particles (and non-SA boundary particles):

$$\forall a \in \mathcal{V} \qquad p_a = \frac{1}{\sum\limits_{b \in \mathcal{M}} V_b w_{ab}} \left(\sum_{b \in \mathcal{M}} V_b \left(p_b + \Delta \rho \, \boldsymbol{g} \cdot (\boldsymbol{r}_a - \boldsymbol{r}_b) \right) w_{ab} \right)$$

Using the following notations:

$$\boldsymbol{\nabla} w_{ab} = w'_{ab} \, \boldsymbol{e}_{ab} \qquad \boldsymbol{\nabla} \gamma_{as} = \gamma'_{as} \, \boldsymbol{n}_s$$

The Morris' Laplacian of effective pressure reads:

$$L_a\{p_b\} = \frac{2}{\gamma_a} \sum_{b \in \mathcal{M} \cup \mathcal{V}} V_b \frac{w'_{ab}}{r_{ab}} (p_a - p_b) - \frac{1}{\gamma_a} \sum_{s \in \mathcal{S}} \gamma'_{as} (\boldsymbol{\nabla} p_a + \boldsymbol{\nabla} p_s) \cdot \boldsymbol{n}_s$$

Boundary term treatment (only for SA boundary conditions):

$$(\boldsymbol{\nabla} p_a + \boldsymbol{\nabla} p_s) \cdot \boldsymbol{n}_s = \left(\frac{\partial p}{\partial n_s}\bigg|_a + \frac{\partial p}{\partial n_s}\bigg|_s \right)$$

$$\approx 2 \frac{\partial p}{\partial n_s}\bigg|_s = 2 \Delta\rho \, \boldsymbol{g} \cdot \boldsymbol{n}_s$$

Finally:

$$L_a\{p_b\} = \frac{2}{\gamma_a} \sum_{b \in \mathcal{M} \cup \mathcal{V}} V_b \frac{w'_{ab}}{r_{ab}} (p_a - p_b) - \frac{2}{\gamma_a} \sum_{s \in \mathcal{S}} \gamma'_{as} \Delta\rho \, \boldsymbol{g} \cdot \boldsymbol{n}_s$$

Equation to solve within the granular media domain:

$$\forall a \in \mathcal{M} \setminus \mathcal{I} \qquad L_a\{p_b\} = 0$$

$$\sum_{b \in \mathcal{M} \cup \mathcal{V}} V_b \frac{w'_{ab}}{r_{ab}} (p_a - p_b) = \sum_{s \in \mathcal{S}} \gamma'_{as} \Delta\rho \, \boldsymbol{g} \cdot \boldsymbol{n}_s$$

$$\sum_{b \in \mathcal{M} \cup \mathcal{V}} V_b \frac{w'_{ab}}{r_{ab}} p_a - \sum_{b \in \mathcal{M} \setminus \mathcal{I}} V_b \frac{w'_{ab}}{r_{ab}} p_b = \sum_{b \in \mathcal{I} \cup \mathcal{V}} V_b \frac{w'_{ab}}{r_{ab}} p_b + \sum_{s \in \mathcal{S}} \gamma'_{as} \Delta\rho \, \boldsymbol{g} \cdot \boldsymbol{n}_s$$

$$\boldsymbol{A} \, \boldsymbol{x} = \boldsymbol{b}$$

$$\begin{cases} x_a = p_a \\ A_{aa} = \sum\limits_{b \in \mathcal{M} \cup \mathcal{V}} V_b \dfrac{w'_{ab}}{r_{ab}} \\ A_{ab} = - \sum\limits_{b \in \mathcal{M} \setminus \mathcal{I}} V_b \dfrac{w'_{ab}}{r_{ab}} \\ b_a = \sum\limits_{b \in \mathcal{I} \cup \mathcal{V}} V_b \dfrac{w'_{ab}}{r_{ab}} p_b + \sum\limits_{s \in \mathcal{S}} \gamma'_{as} \Delta\rho \, \boldsymbol{g} \cdot \boldsymbol{n}_s \end{cases}$$

The linear system to solve is:

$$\boldsymbol{A} \, \boldsymbol{x} = \boldsymbol{b}$$

$\boldsymbol{A}$ can be decomposed into diagonal and remainder matrices $\boldsymbol{D}$ and $\boldsymbol{R}$:

$$\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{R}$$

The linear system can be re-written as:

$$\boldsymbol{D}\ \boldsymbol{x} = \boldsymbol{b} - \boldsymbol{R}\ \boldsymbol{x}$$

Denoting k the Jacobi solver iteration index:

$$\boldsymbol{x}^{k+1} = \boldsymbol{D}^{-1}\left(\boldsymbol{b} - \boldsymbol{R}\ \boldsymbol{x}^{k}\right)$$

The following stopping criterion is used:

$$\frac{||\boldsymbol{A}\ \boldsymbol{x}^{k} - \boldsymbol{b}||_{\infty}}{p_{\text{ref}}} < \epsilon_{\text{res}}$$

with $p_{\text{ref}} = \dfrac{\Delta\rho c_0^2}{100}$. Denoting k the Jacobi solver iteration index:

$$\boldsymbol{x}^{k+1} = \boldsymbol{D}^{-1}\left(\boldsymbol{b} - \boldsymbol{R} \cdot \boldsymbol{x}^{k}\right)$$

We need to build the vector $\boldsymbol{d}$ (diagonal elements), $\boldsymbol{b}$ (right hand-side term) and $\boldsymbol{rx}$ (matrix-vector product $\boldsymbol{R} \cdot \boldsymbol{x}^{k}$).

$$\begin{cases} d_a = \displaystyle\sum_{b \in \mathcal{M} \cup \mathcal{V}} V_b \frac{w'_{ab}}{r_{ab}} \\ rx_a = -\displaystyle\sum_{b \in \mathcal{M} \setminus \mathcal{I}} V_b \frac{w'_{ab}}{r_{ab}} p_b \\ b_a = \displaystyle\sum_{b \in \mathcal{I} \cup \mathcal{V}} V_b \frac{w'_{ab}}{r_{ab}} p_b + \sum_{s \in \mathcal{S}} \gamma'_{as} \Delta\rho\, \boldsymbol{g} \cdot \boldsymbol{n}_s \end{cases}$$

In our case, the right hand-side term $\boldsymbol{b}$ partly depends on the previous Jacobi iteration:

- → the vertex particles effective pressure must be updated every Jacobi iteration to ensure that the Neumann condition is properly enforced.
- → the stopping criterion must also depend and the backward error on vertex:

$$\max_{a \in \mathcal{V}} \frac{|p_a^{k+1} - p_a^{k}|}{p_{\text{ref}}} < \epsilon_{\text{err}}$$

Algorithm

```
-enforce the Dirichlet BC at interface and free-surface free particles
of the granular phase;
-enforce the Neumann BC at vertex from the previous time-step;
while stopping criterion is false do:
      -build d, b and rx^k;
      -compute x^{k+1} for free particles;
      -compute the residual inside the fluid domain;
      -enforce the Neumann BC at vertex from x^{k+1};
      -compute the backward error at vertex particles;
end
```

Remark The criterion on the residual should be:

$$\frac{||\boldsymbol{A}\ \boldsymbol{x}^k - \boldsymbol{b}^k||_\infty}{p_{\text{ref}}} < \epsilon_{\text{res}}$$

However, the residual is computed while computing $\boldsymbol{x}$ for free particles. At this step, vertex particles still have the effective pressure of the previous iteration. But, to save an additional loop over the neighbours, we use the available variables anyway:

$$\frac{||\boldsymbol{D}\ \boldsymbol{x}^k + \boldsymbol{R}\ \boldsymbol{x}^{k-1} - \boldsymbol{b}^{k-1}||_\infty}{p_{\text{ref}}} < \epsilon_{\text{res}}$$

References

A. Colagrossi and M. Landrini. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, 191(2):448–475, 2003.

Martin Ferrand, Dominique Laurence, Benedict D. Rogers, Damien Violeau, and Christophe Kassiotis. Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method. *International Journal for Numerical Methods in Fluids*, 71(4):446–472, 2012. doi: 10.1002/fld.3666. DOI: 10.1002/fld.3666.

A. Ferrari, M. Dumbser, E. F. Toro, and A. Armanini. A new 3D parallel SPH scheme for free surface flows. *Computers & Fluids*, 36(6):1203–1217, 2009.

Alex Ghaïtanellis, Damien Violeau, Martin Ferrand, Kamal El Kadi Abderrezzak, Agnès Leroy, and Antoine Joly. A sph elastic-viscoplastic model for granular flows and bed-load transport. *Advances in Water Resources*, 111:156–173, 2018.

Alex Ghaitanellis. *Modelling bed-load sediment transport through a granular approach in SPH*. PhD thesis, Université Paris-Est, 2017.

X.Y. Hu and N.A. Adams. A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, 213(2):844–861, 2006.

P. Jop, Y. Forterre, and O. Pouliquen. A constitutive law for dense granular flows. *Nature*, 441(7094):727–730, 2006.

A. Leroy, D. Violeau, M. Ferrand, and C. Kassiotis. Unified semi-analytical wall boundary conditions applied to 2-D incompressible SPH. *Journal of Computational Physics*, 261:106–129, 2014. ISSN 0021-9991. doi: 10.1016/j.jcp.2013.12.035.

A. Mayrhofer, D. Laurence, B. D. Rogers, D. Violeau, and M. Ferrand. Direct numerical simulation of 3-D turbulent wall bounded flows with SPH. In *Proc. 8th international SPHERIC workshop*, pages 130–138, 2013a.

Arno Mayrhofer, Benedict D. Rogers, Damien Violeau, and Martin Ferrand. Investigation of wall bounded flows using SPH and the unified semi-analytical wall boundary conditions. *Computer Physics Communications*, 184(11):2515–2527, November 2013b. ISSN 0010-4655. doi: 10.1016/j.cpc.2013.07.004.

Arno Mayrhofer, Martin Ferrand, Christophe Kassiotis, Damien Violeau, and François-Xavier Morel. Unified semi-analytical wall boundary conditions in SPH: analytical extension to 3-{D}. *Numerical Algorithms*, pages 1–20, February 2014. ISSN 1017-1398, 1572-9265. doi: 10.1007/s11075-014-9835-y.

J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68:1703–1759, 2005.

Stephen B Pope. *Turbulent Flows*. Cambridge University Press, 7 edition, 2001. URL `http://stacks.iop.org/0957-0233/12/i=11/a=705`.

T. Revil-Baudard and J. Chauchat. A two-phase model for sheet flow regime based on dense granular flow rheology. *Journal of Geophysical Research: Oceans*, 118(2): 619–634, 2013.

K.V. Terzaghi. The shearing resistance of saturated soils and the angle between the planes of shear. In *Proceedings of the 1st international conference on soil mechanics and foundation engineering*, volume 1, pages 54–56. Harvard University Press Cambridge, MA, 1936.

J. P. Vila. On particle weighted methods and smooth particle hydrodynamics. *Mathematical Models and Methods in Applied Sciences*, 9(2):161–209, 1999.

Damien Violeau, Agnès Leroy, and Arno Mayrhofer. Exact computation of SPH wall renormalising integrals in 3-D. In *Proceedings of the 9th International SPHERIC Workshop*, pages 95–102, Paris, May 2014.