Avalanche Simulation Tools for OpenFOAM

The OpenFOAM user guide can be found on https://www.openfoam.com/documentation/user-guide/

- **Author: Matthias Rauter**
- **Mail: matthias (@) rauter (.) it**
- **Twitter: igt_matti**

Introduction

This module provides a model and tools for the simulation of:

- **dense snow avalanches**,
- **powder snow avalanches**,
- **mixed snow avalanches**,
- **turbidity currents** and
- other **gravitational mass flows** that fit into the scheme. This can either be flows of water, debris flows or pyroclastic flows. However, some adaptions to e.g. the friction model might be required.

A **depth-integrated flow model** is applied and solved with the **Finite Area Method**. Tools for simulation setup and post processing are provided. Tools focus on a tight integration with Geographic Information System and provide in- and output routines for Shapefiles and Gridfiles.

Note: Variable names differ between the academic publications and the code and this documentation. The reason is that some of the variable names in literature (e.g. ϕ for phase fraction) are already used in the internals of OpenFOAM (where phi is the flux). Therefore the full equations are repeated here.

Note: This readme and its latex syntax aim to look good when used with LaTeX and StackEdit. Some interpreter struggle to parse the math expression.

Copyright

Source code

Real case example

Wolfsgrube

The digital elevation model for tutorial **wolfsgrube** is provided by AMT DER TIROLER LANDESREGIERUNG (AdTLR) Abteilung Geoinformation

Monterey Canyon

The digital elevation model for tutorial **montereycanyon** is provided by Monterey Bay Aquarium Research Institute (MBARI)

Models (Solver)

faSavageHutterFoam

The solver can be found `applications/solver/faSavageHutterFoam` and called with `faSavageHutterFoam` if the OpenFOAM module is loaded. For some options see `faSavageHutterFoam -help`.
The solver is based on a depth-integrated flow model, similar to the Savage-Hutter model (Savage and Hutter; 1989, 1991).
The theory of this solver is described by Rauter and Tukovic (2018). The respective preprint is availabel on arxiv.org.
The application to natural terrain is described by Rauter, Kofler, Huber and Fellin (2018).

The **governing equations** can be expressed in terms of surface partial differential equations as

$$\frac{\partial h}{\partial t} + \boldsymbol{\nabla}\cdot\left(h\,\overline{\mathbf{u}}\right) = S_e - S_d,$$

$$\frac{\partial\left(h\,\overline{\mathbf{u}}\right)}{\partial t} + \boldsymbol{\nabla}_\mathrm{s}\cdot\left(h\,\overline{\mathbf{u}}\,\overline{\mathbf{u}}\right) = -\frac{1}{\rho}\boldsymbol{\tau}_\mathrm{b} + h\,\mathbf{g}_\mathrm{s} - \frac{1}{2\,\rho}\boldsymbol{\nabla}_\mathrm{s}\left(h\,p_\mathrm{b}\right),$$

$$\boldsymbol{\nabla}_\mathrm{n}\cdot\left(h\,\overline{\mathbf{u}}\,\overline{\mathbf{u}}\right) = h\,\mathbf{g}_\mathrm{n} - \frac{1}{2\,\rho}\boldsymbol{\nabla}_\mathrm{n}\left(h\,p_\mathrm{b}\right) - \frac{1}{\rho}\mathbf{n}_\mathrm{b}\,p_\mathrm{b},$$

with **unknown flow fields**

- depth-averaged flow velocity $\overline{\mathbf{u}}$,
- flow thickness h and
- basal pressure p_b.

Constant parameters are

- the density ρ and
- the gravitational acceleration $\mathbf{g} = \mathbf{g}_\mathrm{s} + \mathbf{g}_\mathrm{n}$.

User-selectable models (see below) have to be provided for

- the basal friction $\boldsymbol{\tau}_\mathrm{b}$ (`frictionModel`)
- the volumetric entrainment rate S_e (`entrainmentModel`)
- the volumetric deposition rate S_d (`depositionModel`)

Initial conditions have to be provided for

- the flow thickness h,
- the depth-integrated flow velocty $\overline{\mathbf{u}}$ and
- the mountain snow cover thickness h_msc (for entrainment of fresh snow).

The classic surface pressure equation (e.g. Fischer et al. 2013) can be applied by deactivating the second term on the right hand side of Equation (3) (switch `pressureFeedback` in `tansportProperties`).

Spatial differential operators

$$\boldsymbol{\nabla}_\mathrm{n} = \left(\mathbf{n}_\mathrm{b}\,\mathbf{n}_\mathrm{b}\right)\cdot\boldsymbol{\nabla}$$

and

$$\boldsymbol{\nabla}_\mathrm{s} = \left(\mathbf{I} - \mathbf{n}_\mathrm{b}\,\mathbf{n}_\mathrm{b}\right)\cdot\boldsymbol{\nabla}$$

are described in detail by Rauter and Tukovic (2018).

The numerical solution is based on the **Finite Area Method**.

faParkerFukushimaFoam

This solver implements the Turbidity Current model of Parker et al. (1986).
It can be found in `applications/solver/faParkerFukushimaFoam` and called with `faParkerFukushimaFoam`. For some options see `faParkerFukushimaFoam -help`.

$$\frac{\partial\,h}{\partial t} + \nabla\cdot\left(h\,\overline{\mathbf{u}}\right) = S_e^{(w)}$$

$$\frac{\partial\,h\,\overline{\mathbf{u}}}{\partial t} + \nabla_\mathrm{s}\cdot\left(h\,\overline{\mathbf{u}}\,\overline{\mathbf{u}}\right) = R\,\mathbf{g}_\mathrm{s}\,\overline{c}\,h - \frac{1}{2}\,R\,g_\mathrm{n}\,\nabla_\mathrm{s}\left(\overline{c}\,h^2\right) - \boldsymbol{\tau}_\mathrm{b}$$

$$\frac{\partial\,\overline{c}\,h}{\partial t} + \nabla\cdot\left(\overline{c}\,h\,\overline{\mathbf{u}}\right) = S_e^{(s)} - S_d^{(s)}$$

with **unknown flow fields**

- depth-averaged flow velocity $\overline{\mathbf{u}}$,
- flow thickness h and
- depth-averaged sediment (or particle) concentration $\overline{c}$.

Constant parameters are

- the density ration between sediment and water R (or more generally, particles and fluid) and
- the gravitational acceleration $\mathbf{g} = \mathbf{g}_\mathrm{s} + \mathbf{g}_\mathrm{n}$.

User-selectable models (see below) have to be provided for

- the basal friction $\boldsymbol{\tau}_\mathrm{b}$ (`suspensionFrictionModel`)
- the volumetric sediment entrainment rate $S_e^{(s)}$ (`suspensionEntrainmentModel`)
- the volumetric sediment deposition rate $S_e^{(s)}$ (`suspensionDepositionModel`)
- the volumetric ambient fluid entrainment rate $S_e^{(w)}$ (`ambientEntrainmentModel`)

Initial conditions and boundary conditions have to be provided for

- the flow thickness h,
- the depth-integrated flow velocty $\overline{\mathbf{u}}$ and
- the depth-integrated sediment concentration $\overline{c}$
- the erodible sediment thickness h_msc

faTwoLayerAvalancheFoam

The solver implements a two layer approach for a mixed snow avalanche. The dense flow layer is simulated with the Savage-Hutter model (Savage and Hutter; 1989, 1991), similar to faSavageHutterFoam and the powder cloud layer is simulated with the model of Parker et al. (1986), similar to faParkerFukushima. The equations can be found in the sections above. It can be found in `applications/solver/faTwoLayerAvalancheFoam` and called with `faTwoLayerAvalancheFoam`. For some options see `faTwoLayerAvalancheFoam -help`.

The dense flow model fields are marked with a index 1, the suspension flow fields are marked with an index 2. The two layers are coupled with two fluxes: The suspension deposition that brings particles from the suspension to the dense core and the dense core fluidisation flux that brings particles from the dense core to the suspension, S_f.

$$\frac{\partial h_1}{\partial t} + \nabla^\Gamma \cdot (h_1 \, \overline{\mathbf{u}}_1) = S_{e,1} - S_{d,1} - S_f + S_{d,2}^{(s)}$$

$$\frac{\partial h_1 \, \overline{\mathbf{u}}_1}{\partial t} + \xi_1 \, \nabla_\mathrm{s}^\Gamma \cdot (h_1 \, \overline{\mathbf{u}}_1 \, \overline{\mathbf{u}}_1) = -\frac{\boldsymbol{\tau}_{b,1}}{\rho_1} + h_1 \, \mathbf{g}_\mathrm{s} - \frac{1}{2 \, \rho_1} \, \nabla_\mathrm{s}^\Gamma \, (h_1 \, p_1)$$

$$\xi_1 \, \nabla_\mathrm{n}^\Gamma \cdot (h_1 \, \overline{\mathbf{u}}_1 \, \overline{\mathbf{u}}_1) = h_1 \, \mathbf{g}_\mathrm{n} - \frac{1}{2 \, \rho_1} \, \nabla^\Gamma \, (h_1 \, p_1) - \frac{1}{\rho_1} \, \mathbf{n}^\Gamma \, p_1$$

$$\frac{\partial \, h_2}{\partial t} + \boldsymbol{\nabla}^\Gamma \cdot (h_2 \, \overline{\mathbf{u}}_2) = S_{e,2}^{(w)}$$

$$\frac{\partial \, c_2 \, h_2}{\partial t} + \boldsymbol{\nabla}^\Gamma \cdot (c_2 \, h_2 \, \overline{\mathbf{u}}_2) = S_{e,2}^{(s)} - S_{d,2}^{(s)} + S_f$$

$$\frac{\partial \, (1 + R \, c_2) \, h_2 \, \overline{\mathbf{u}}_2}{\partial t} + \xi_2 \, \boldsymbol{\nabla}_\mathrm{s}^\Gamma \cdot ((1 + R \, c_2) \, h_2 \, \overline{\mathbf{u}}_2 \, \overline{\mathbf{u}}_2) = -\frac{\tau_2}{\rho_\mathrm{c}} + R \, c_2 \, h_2 \, \mathbf{g}_\mathrm{s} - \frac{1}{2} \, \boldsymbol{\nabla}_\mathrm{s}^\Gamma \left((1 + R \, c_2) \, g_n \, h_2^2 \right)$$

Further, the coupling flux S_f is associated with a momentum flux that is not shown in the Equations for simplicity. The suspension entrainment $S_{e,2}^{(s)}$ is only active if there is no dense flow present at the respective position.

User-selectable models (see below) have to be provided for:

- the dense flow basal friction $\boldsymbol{\tau}_{b,1}$ (`frictionModel`)
- the dense flow entrainment rate $S_{e,1}$ (`entrainmentModel`)
- the dense flow deposition rate $S_{d,1}$ (`depositionModel`)
- the dense flow fluidisation rate S_f (`couplingModel`)
- the suspension basal friction $\boldsymbol{\tau}_{b,2}$ (`suspensionFrictionModel`)
- the suspension entrainment rate $S_{e,2}^{(s)}$ (`suspensionEntrainmentModel`)
- the suspension deposition rate $S_{d,2}^{(s)}$ (`suspensionDepositionModel`)
- the suspension ambient fluid entrainment rate $S_{e,2}^{(w)}$ (`ambientEntrainmentModel`)

Friction models

Dense flow friction models (frictionModel)

The friction model describes the friction $\boldsymbol{\tau}_\mathrm{b}$ between flowing mass (avalanche) and the terrain. The direction of the friction vector always aligns with the velocity vector $\overline{\mathbf{u}}$. These models are used in the solvers:

- faSavageHutterFoam
- faTwoLayerAvalancheFoam

The friction model has to be set in the file `constant/transportProperties` (see below).
These friction models can be found in the folder `src/avalanche/friction` and are based on the class `frictionModel`.

Currently there are the following friction models available:

- DarcyWeisbach (Liquid, formerly used for solver tests)
- ManningStrickler (Liquid, formerly used for solver tests)
- MuI (Granular flow, avalanche)
- PoliquenForterre (Granular flow, avalanche)
- Voellmy (Granular flow, avalanche)
- kt (Granular flow, avalanche)

Darcy-Weisbach

`DarcyWeisbach`

Weisbach (1845) (wikipedia.org):

This friction model was applied in comparisons with analytical solutions in Rauter and Tukovic (2018), section 6.1.3.

$$|\boldsymbol{\tau}_\mathrm{b}| = C_f^2 \, \rho \, g \, |\mathbf{u}|^2$$

```
frictionModel       DarcyWeisbach;
DarcyWeisbachCoeffs
{
    Cf              Cf [0 -1 2 0 0 0 0] 0.000625;

    g               g  [0 1 -2 0 0 0 0] 9.81;
}
```

Manning-Strickler

`ManningStrickler`

Manning (1890) (wikipedia.org):

$$|\boldsymbol{\tau}_\mathrm{b}| = n^2 \, \rho \, g \, \frac{|\mathbf{u}|^2}{h^{1/3.}}$$

Parameters:

```
frictionModel       ManningStrickler;
ManningStricklerCoeffs
{
    n               n [ 0 -0.333333333333333 1 0 0 0 0] 1.0;

    g               g [0 1 -2 0 0 0 0] 9.81;
}
```

Voellmy

`Voellmy`

Friction model following Voellmy (1955):

This friction model was applied in Rauter, Kofler, Huber and Fellin (2018).

$$|\boldsymbol{\tau}| = \mu \, p + \frac{\rho \, g}{\xi} \, |\mathbf{u}|^2$$

Parameters (see also tutorial `wolfsgrube`):

```
frictionModel       Voellmy;
VoellmyCoeffs
{
    mu              mu [0 0 0 0 0 0 0] 0.25;        //dry friction coefficient

    xi              xi [0 1 -2 0 0 0 0] 10000;      //voellmy turbulence coefficient
}
```

Kinetic Theory

`kt`

Simplified Kinetic Theory following Rauter et al. (2016).

$$|\boldsymbol{\tau}| = \mu\, p + \frac{\rho\, g}{\chi} \frac{|\mathbf{u}|^2}{h^2}$$

Parameters:

```
frictionModel       kt;
ktCoeffs
{
    mu              mu [0 0 0 0 0 0 0] 0.25;       //dry friction coefficient

    chi             chi [0 -1 -2 0 0 0 0] 10000;   //turbulent friction coefficient
}
```

mu(I)

`muI`

Popular mu(I) following Jop et al. (2006) (see also wikipedia.org)

Shear rate at the base following Bagnold Profile:

$$\dot{\gamma} = \frac{5}{2} \frac{|\mathbf{u}|}{h}$$

Inertia number:

$$I = \frac{\dot{\gamma}\, d}{\sqrt{p/\rho_p}}$$

Friction coefficient depending on inertia number:

$$\mu = \mu_s + \frac{\mu_2 + \mu_s}{I_0/I + 1}$$

Basal friction:

$$|\boldsymbol{\tau}| = \mu(I)\, p$$

Parameters (see also tutorial `simpleslope`):

```
frictionModel       MuI;
MuICoeffs
{
    d               d [ 0 1 0 0 0 0 0] 0.005;        //particle diameter

    rho_p           rho_p [ 1 -3  0 0 0 0 0 ] 2500.; //particle density

    mu_s            mu_s [0 0 0 0 0 0 0 ] 0.38;      //friction coefficient (low limit)

    mu_2            mu_2 [0 0 0 0 0 0 0 ] 0.65;      //friction coefficient (high limit)

    I_0             I_0 [0 0 0 0 0 0 0 ] 0.30;       //reference inertia number
}
```

Poliquen Forterre (2008)

`PoliquenForterre`

First mu(I)-rheology following Pouliquen & Forterre (2002). Similar to `muI`, however with the parameterisation of Pouliquen & Forterre (2002).
See also Johnson and Gray, (2011).

$$Fr = \frac{|\mathbf{u}|}{\sqrt{h\, g_n}}$$

$$h_s = h \frac{\beta}{Fr}$$

$$\mu_{\text{stop}} = \tan(\zeta_1) + \frac{\tan(\zeta_2) - \tan(\zeta_1)}{1 + h_s/L}$$

$$\mu_{\text{start}} = \tan(\zeta_3) + (\tan(\zeta_2) - \tan(\zeta_1))\, \exp(-h_s/L)$$

$$\mu = \left(\frac{Fr}{\beta}\right)^{\gamma} (\mu_{\text{stop}} - \mu_{\text{start}}) + \mu_{\text{start}}$$

$$|\boldsymbol{\tau}| = \mu(I)\, p$$

Parameters:

```
frictionModel       PoliquenForterre;
PoliquenForterreCoeffs
{
    L               L [ 0 1 0 0 0 0 0 ] 0.010;

    zeta1           zeta1 [ 0 0 0 0 0 0 0 ] 21;

    zeta2           zeta2 [ 0 0 0 0 0 0 0 ] 30.7;

    zeta3           zeta3 [ 0 0 0 0 0 0 0 ] 22.2;

    beta            beta [ 0 0 0 0 0 0 0 ] 0.136;

    gamma           gamma [0 0 0 0 0 0 0] 1e-3;
}
```

Suspension friction models (suspensionFrictionModels)

The friction model describes the friction $\boldsymbol{\tau}_\mathrm{b}$ between flowing mass (turbidity current) and the bottom. The direction of the friction vector always aligns with the velocity vector $\overline{\mathbf{u}}$. These models are used in the solvers:

- faParkerFukushima
- faTwoLayerAvalancheFoam

The friction model has to be set in the file `constant/transportProperties` (see below).
Friction models can be found in the folder `src/avalanche/friction`.
To implement a new friction model, copy an existing one, rename it and modify it.

Currently there are the following friction models available:

- laminar (Parker et al. (1986) 3-Equations model)
- turbulent (Parker et al. (1986) 4-Equations model)

laminarSuspension

`laminarSuspension`

Parker et al. (1986).

This friction model can be used to get the 3-Equation model of Parker et al. (1986).

$$|\boldsymbol{\tau}_\mathrm{b}| = c_\mathrm{D}\, |\overline{\mathbf{u}}|\, \overline{\mathbf{u}}$$

```
suspensionFrictionModel laminarSuspension;
laminarSuspensionCoeffs
{
    cd      cd      [0 0 0 0 0 0 0] 0.0006;
}
```

turbulentSuspension

`turbulentSuspension`

Parker et al. (1986).

This friction model can be used to get the 4-Equation model of Parker et al. (1986). Note that this model is only compatible with faParkerFukushimaFoam.

Note: This model does not work with faTwoLayerAvalanceFoam due to the relative velocity (between dense flow and suspension layer) that is delivered to the friction model of the suspension layer.

$$|\boldsymbol{\tau}_\mathrm{b}| = \alpha\, \overline{k}$$

with the depth-averaged kinetic energy $\overline{k}$, calculated with the PDE

$$\frac{\partial\, \overline{k}\, h}{\partial t} + \nabla \cdot \left(\overline{k}\, h\, \overline{\mathbf{u}}\right) = \boldsymbol{\tau}_\mathrm{b} \cdot \overline{\mathbf{u}} + \frac{1}{2} |\overline{\mathbf{u}}|^2\, S_e^{(w)} - \epsilon\, h - R\, g_\mathrm{n}\, v_\mathrm{s}\, \overline{c}\, h - \frac{1}{2} R\, g_\mathrm{n}\, h \left(S_e^{(s)} - S_d^{(s)}\right)$$

with

$$\epsilon = \beta \frac{\overline{k}^{3/2}}{h}$$

$$Ri = \frac{R\, g_{\mathrm{n}}\, \overline{c}\, h}{|\overline{\mathbf{u}}|^2}$$

β is an **optional parameter**. If not given, it will be calculated as

$$\beta = \frac{2\,\frac{S_e^{(w)}}{\overline{\mathbf{u}}}\left(1 - Ri - 2\,\frac{c_{\mathrm{D}}}{\alpha}\right) + c_{\mathrm{D}}}{\left(\frac{c_{\mathrm{D}}}{\alpha}\right)^{3/2}}$$

```
suspensionFrictionModel turbulentSuspension;
turbulentSuspensionCoeffs
{
    cd      cd      [0 0 0 0 0 0 0] 0.01;

    alpha   alpha   [0 0 0 0 0 0 0] 0.1;

    beta    beta    [0 0 0 0 0 0 0] 0.7;

    R       R       [ 0 0 0 0 0 0 0] 1.65;

    Ds      Ds      [ 0 1 0 0 0 0 0] 0.00005;

    kmin    kmin    [0 2 -2 0 0 0 0] 1e-7;

    nu      nu      [0 2 -1 0 0 0 0] 1e-6;
}
```

Entrainment models

Dense flow entrainment models (entrainmentModel)

Entrainment describes the erosion of the intact snow cover and intake of the respective material $S_{\mathrm{e}}\,(\mathrm{m/s})$ into the avalanche. It is assumed that the snow cover has the same density as the avalanche.

The entrainment model has to be set in the file `constant/transportProperties` (see below).

Entrainment models can be found in the folder `src/avalanche/entrainment`. Currently there are the following available:

- entrainmentOff
- Front
- Erosionenergy
- Medina
- Ramms

No entrainment

`entrainmentOff`

Choose to turn off entrainment ($S_{\mathrm{e}} = 0$), no parameters.

Front

`Front`

Simple front entrainment. Entrainment of the total mountain snow cover within a cell is triggered when $h > h_{\mathrm{trigger}}$.

Parameters:

```
entrainmentModel    Front;
FrontCoeffs
{
    htrigger        htrigger [ 0 1 0 0 0 0 0 ] 0.01;
}
```

Erosionenergy

`Erosionenergy`

Entrainment model following SamosAT, see e.g., Rauter et al. (2016).

The entrainment rate S_e is calculated as

$$S_\mathrm{e} = \frac{\boldsymbol{\tau} \cdot \mathbf{u}}{\rho\, e_\mathrm{b}}$$

Parameters:

```
entrainmentModel    Erosionenergy;
ErosionenergyCoeffs
{
    eb              eb [0 2 -2 0 0 0 0] 11500;      //specific erosion energy
}
```

Medina

`Medina`

Entrainment model following the approach of Medina et al. (2008).
Entrainment is calculated from stability considerations of the basal layer.

The entrainment rate S_e is calculated as

$$S_e = \frac{|\tau_b| - \tau_r}{\rho\left(|g_n| - \mu|g_s|\right)},$$

with the resisting shear strength

$\tau_r = \tau_c + \mu\, p_b$.

The cohesion τ_c and the friction coefficient μ of the basal layer are input parameters.
To improve stability, entrainment can be relaxed:
$S_e = \mathrm{relaxation\ Factor} \cdot S_{m,\mathrm{trial}}$.

Parameters:

```
entrainmentModel    Medina;
MedinaCoeffs
{
    tauc            tauc [1 -1 -2 0 0 0 0]        100;

    mu              mu [0 0 0 0 0 0 0]            0.45;

    relax 0.1;
}
```

Ramms

`Ramms`

Entrainment model following the approach from RAMMS, see e.g., Christen et al. (2016).

The entrainment rate S_e is calculated as

$$S_e = \kappa\, |\overline{\mathbf{u}}|.$$

Parameters:

```
entrainmentModel    Ramms;
RammsCoeffs
{
    kappa           kappa [0 0 0 0 0 0 0]       0.001;
}
```

Suspension entrainment models (suspensionEntrainmentModel)

The entrainment model has to be set in the file `constant/transportProperties` (see below).

Entrainment models can be found in the folder `src/avalanche/entrainment`. Currently there are the following available:

- ParkerFukushimaEntrainment

ParkerFukushimaEntrainment

`ParkerFukushimaEntrainment`

Parker et al. (1986).

This model implements the standard entrainment model as used by Parker et al. (1986).

$$S_e^{(s)} = v_s \begin{cases} 0.3 & \text{for } Z > Z_m, \\ 3 \cdot 10^{-12}\, Z^{10} \left(1 - \frac{Z_c}{Z}\right) & \text{for } Z_c < Z < Z_m, \\ 0 & \text{for } Z < Z_c, \end{cases}$$

with

$Z = \sqrt{R_s}\,\mu$

$R_s = \frac{\sqrt{R\, g_n\, d_s^3}}{\nu}$

$\mu = \frac{\sqrt{|\boldsymbol{\tau}_b|}}{v_s}$

$v_s = \frac{R\, g_n\, d_s}{18\,\nu}$

```
suspensionEntrainmentModel ParkerFukushimaEntrainment;
ParkerFukushimaEntrainmentCoeffs
{
    R       R       [ 0 0 0 0 0 0 0] 1.65;

    Ds      Ds      [ 0 1 0 0 0 0 0] 0.00005;

    Zc      Zc      [ 0 0 0 0 0 0 0] 0.5;

    Zm      Zm      [ 0 0 0 0 0 0 0] 13.2;

    nu      nu      [0 2 -1 0 0 0 0] 1e-6;
}
```

Ambient fluid entrainment model (ambientEntrainmentModel)

The ambient fluid entrainment model controls the entrainment of ambient fluid (air for powder snow avalanches, water for turbidity currents) into the suspension flow. Ambient fluid entrainment models can be found in the folder `src/avalanche/entrainment`.

They have all in common that they depend on the Richardson Number:

$$Ri = \frac{R\, g_n\, c\, h}{\overline{\mathbf{u}}^2}$$

The following models are available:

- ParkerFukushima
- Ancey
- Turner

ParkerFukushima

This is the standard model from Parker et al. (1986).

$$S^{(w)} = \frac{e^{wf}}{Ri_0 + Ri} |\overline{\mathbf{u}}|$$

Parameters:

```
ambientEntrainmentModel ParkerFukushimaEntrainment;
ParkerFukushimaEntrainmentCoeffs
{
    ewf     ewf     [ 0 0 0 0 0 0 0] 0.00153;
```

```
    Ri0     Ri0     [ 0 0 0 0 0 0 0] 0.0204;
}
```

Ancey

$$S^{(w)} = |\overline{\mathbf{u}}|\,\alpha_2 \begin{cases} \exp\left(-\alpha_1\, Ri^2\right) & \text{for } Ri < 1, \\ \exp\left(-\alpha_1\right)/Ri & \text{for } Ri \geq 1. \end{cases}$$

```
ambientEntrainmentModel AnceyEntrainment;
AnceyEntrainmentCoeffs
{
    alpha1       alpha1       [ 0 0 0 0 0 0 0] 1.6;

    alpha2       alpha2       [ 0 0 0 0 0 0 0] 0.05;
}
```

Turner

$$S^{(w)} = |\overline{\mathbf{u}}| \begin{cases} \dfrac{Ri_0 - Ri}{\alpha_1 + \alpha_2\, Ri} & \text{for } Ri < Ri_0, \\ 0 & \text{for } Ri \geq Ri_0. \end{cases}$$

Parameters:

```
ambientEntrainmentModel AnceyEntrainment;
AnceyEntrainmentCoeffs
{
    alpha1      alpha1       [ 0 0 0 0 0 0 0] 10;

    alpha2      alpha2       [ 0 0 0 0 0 0 0] 50;

    Ri0         Ri0          [ 0 0 0 0 0 0 0] 0.8;
}
```

Deposition models

Dense flow deposition models (depositionModel)

Deposition takes into account that mass is gradually lost in the avalanche during deceleration. The deposition model has to be set in the file `constant/transportProperties` (see below).

Deposition models can be found in the folder `src/avalanche/deposition`. The following models are available:

- depositionOff
- StoppingProfile

No deposition

`depositionOff`

Choose to turn off deposition.

Stoppingprofile

`Stoppingprofile`
A deposition model derived from a decelerating velocity profile.

Rauter and Köhler (2019), “Constraints on entrainment and deposition models in avalanche simulations from high-resolution radar data”, Geosciences, 2019

Deposition model based on a decelerating Bagnold profile.

$$S_d = \frac{a}{|\overline{\mathbf{u}}|}\,\frac{\mathrm{d}\left(h|\overline{\mathbf{u}}|\right)}{\mathrm{d}t}$$

with

$$a = \begin{cases} \left(\dfrac{u_{dep} - |\overline{\mathbf{u}}|}{u_{dep}}\right)^{a_{dep}} & \text{for} \quad |\overline{\mathbf{u}}| \leq u_{dep} \;\wedge\; \dfrac{\mathrm{d}|\overline{\mathbf{u}}|}{\mathrm{d}t} < 0 \\ 0 & \text{else} \end{cases}$$

and the parameter u_{dep} and a_{dep}.

Parameters:

```
depositionModel     Stoppingprofile;
StoppingprofileCoeffs
{
    ud              ud [0 1 -1 0 0 0 0]         1.5;

    ad              ad [0 0  0 0 0 0 0]         1.0;
}
```

Suspension deposition models (suspensionDepositionModel)

The supsension deposition model controls the settling of particles from the suspension. The suspension deposition model has to be set in the file `constant/transportProperties` (see below).

Suspension deposition models can be found in the folder `src/avalanche/deposition`. The following models are available:

- ParkerFukushima

ParkerFukushima

This is the standard deposition model from Parker et al. (1986). It is based on the settling velocity of grains in fluid.

$S_d^{(s)} = v_\mathrm{s}\, r_0\, \overline{c}$
with
$r_0 = 1 + 31.5\, \mu^{-1.46}$,
$\mu = \frac{\sqrt{|\boldsymbol{\tau}_\mathrm{b}|}}{v_\mathrm{s}}$
$v_\mathrm{s} = \frac{R\, g_\mathrm{n}\, d_\mathrm{s}}{18\, \nu}$

Parameters:

```
suspensionDepositionModel ParkerFukushimaDeposition;

ParkerFukushimaDepositionCoeffs
{
    nu      nu      [0 2 -1 0 0 0 0] 1e-6;

    Ds      Ds      [ 0 1 0 0 0 0 0] 0.00005;
}
```

Coupling models (couplingModel)

These models describe the sediment flux from the dense core to the powder cloud S_f. These models are required in

- faTwoLayerAvalancheFoam

Inertial (couplingInertial)

This model represents on the simplest methods to couple the dense core with the powder cloud. The flux depends solely on flow parameters of the dense flow.

$S_f = \max\left(I_1 - I_0, 0\right)\, s_f$
with
$I_1 = \frac{\dot{\gamma}_1\, d_1}{\sqrt{p_1 / \rho_s}}$
$\dot{\gamma}_1 = \frac{5}{2}\, \frac{|\mathbf{u_1}|}{h_1}$

Parameters:

```
couplingModel couplingInertial;

couplingInertialCoeffs
{
    I0       I0       [ 0 0 0 0 0 0 0] 0.5;
```

```
    u0       u0       [ 0 0 0 0 0 0 0] 1e-5;

    d        d        [ 0 1 0 0 0 0 0] 0.01;

    rhos     rhos     [ 1 -3 0 0 0 0 0] 800;
}
```

Example: wolfsgrube_mixed

functionObjects

Various functionObjects are available and can be added to the execution of a solver to extend its capabilities. They are added in the subDict `functions` in `system/controlDict`:

```
functions
{
    gridfileWrite
    {
        type        gridfileWrite;
        ...
        ...
    }
}
```

shapefileWrite

This functionObject writes one shapefile in each timestep that is marked for writing. The shapefile contains a selection of OpenFOAM fields and can be openend in e.g. QGSI.

Parameters:

- `fields`: A list of fields that will be added to the shapefile. Accepts regular expressions, e.g. ".*" to export all fields.
- `writeOption`: `autoWrite` / `anyWrite`. Write only fields that are marked by the solver for writing or wirte any fields in the given list.
- `prefix`: The first part of the filename. Will be extended with the time.
- `offset`: Mapping back to world coordinates from simulation coordinates (see gridToSTL/releaseAreaMapping)

```
shapefileWrite
{
    type    shapefileWrite;
    libs    (faAvalanche);

    //fields     (".*"); //All fields
    fields      (h Us);

    writeOption autoWrite;
    //writeOption anyWrite;

    prefix  "polys";

    offset (5000.0 -220000.0 0);
}
```

gridfileWrite

This functionObject generates gridfiles out of fields in each timestep that is marked for writing.

The parameters of the gridfile can be given in 3 ways:

- standard parameters `dx`, `dy`, `ncols`, `nrows`, `xllcorner`, `yllcorner`.
- grid-extends and cellsize `xmin`, `xmax`, `ymin`, `ymax`, `dx`, `dy`.
- cellsize `dx`, `dy` and the extends will be calculated automatically from the mesh.

Parameters:

- `fields`: A list of fields that will be written as grid file. Accepts regular expressions, e.g. ".*" to export all fields.
- `writeOption`: `autoWrite` / `anyWrite`. Write only fields that are marked by the solver for writing or wirte any fields in the given list.
- `dx`: Cell size of the gridfile - x-direction

- `dy` : Cell size of the gridfile - y-direction
- `ncols` : Number of columns in the gridfile (optional)
- `nrows` : Number of rows in the gridfile (optional)
- `xllcorner` : Position of lower left cell in gridfile (optional)
- `yllcorner` : Position of lower left cell in gridfile (optional)
- `xmin` : extend in x-direction (optional)
- `xmax` : extend in x-direction (optional)
- `ymin` : extend in y-direction (optional)
- `ymax` : extend in y-direction (optional)
- `postfix` : The postfix of the files.
- `offset` : Mapping back to world coordinates from simulation coordinates (see gridToSTL/releaseAreaMapping). Note that grid2Stl can write a file with the correct offset, that can be included into the file with `"#include "../constant/offset";`

```
gridfileWrite
{
    type    gridfileWrite;
    libs    (faAvalanche);

    fields  (h Us);

    secondOrder on;

    writeOption autoWrite;
    //writeOption anyWrite;

    ncols        426;
    nrows        258;
    xllcorner    -803;
    yllcorner    -83;
    dx           10;
    dy           10;

    postfix  ".asc";

    offset (5000.0 -220000.0 0); //or #include "../constant/offset";
}
```

isoLine

The isoSurface functioObject allows a direct export of isolines to GIS.

Parameters:

- `field` : The name of the field to be processed and exported.
- `fileName` : The filename where the isoline will be saved as a shapefile (*.shp)
- `values` : List of values for isolines
- `offset` : Mapping back to world coordinates from simulation coordinates (see gridToSTL/releaseAreaMapping). Note that grid2Stl can write a file with the correct offset, that can be included into the file with `"#include "../constant/offset";`

```
h1iso
{
    type    isoLine;
    libs    (faAvalanche);

    field   h;
    fileName "h";

    values   (0.5 2 4 6 8 10 12 14 16);

    offset (5000.0 -220000.0 0); //or #include "../constant/offset";
}


// ************************************************************************* //
```

autoAreaToVolumeMapping

The native Paraview reader is not able to read areaFields. This functionObjects maps areaFields to volumeFields so Paraview can read them.

Parameters:

- `fields` : A list of fields that will be written as grid file. Accepts regular expressions, e.g. ".*" to export all fields.
- `writeOption` : `autoWrite` / `anyWrite` . Write only fields that are marked by the solver for writing or wirte any fields in the given list.
- `prefix` : prefix for filenames so they don't overwrite the areaFields.

```
autoAreaToVolumeMapping
{
    type    autoAreaToVolumeMapping;
    libs    (faAvalanche);

    fields  (".*");

    writeOption autoWrite;
    //writeOption anyWrite;

    prefix  "fa_";
}
```

Utilities

slopeMesh

`slopeMesh`

This simple utility can be used to create simple slopes as used in many of the tutorials.

For an example see `tutorials/simpleslope/` .
Parameters are set in `constant/slopeMeshDict` .
The source code can be found in `applications/utilities/slopeMesh` .

gridToSTL

This utility allows to generate STL-files from topography data (ESRI grid files). STL-files can then be used to generate the mesh required for simulations.

For an example see `tutorials/wolfsgrube` .
Parameters are set in `system/gridToSTLDict` .
The source code can be found in `applications/utilities/slopeMesh` .

Example reading topography from grid file (*.asc) and boundary polygon from shape file* (.shp, *.shx, *.dbf)

```
stlname "constant/surface.stl";          //output file

gridname "constant/gisdata/dem.asc";     //topography file

boundary fromShape;                      //read boundary of the meshed area from shapefile

shapeBoundary "constant/gisdata/aoi";    //shapefile containing the boundary polygon

divisions 300;                           //number of points a polygon side is divided into

domainHeight 500.0;                      //height of the volume mesh

offset (5000.0 -220000.0 0);             //offset the final mesh by this value
```

Example reading topography from grid file (*.asc) and boundary polygon from dictionary

```
stlname "constant/surface.stl";          //output file

gridname "constant/gisdata/dem.asc";     //topography file

boundary fromPoints;                     //read boundary of the meshed area from list below

boundaryPoints                           //list with vertices describing the meshed area
(
    (-4666.57 221593 0)
    (-4259.78 222062 0)
    (-3749.64 221983 0)
    (-3020.48 220056 0)
    (-3642.49 220017 0)
);
```

```
divisions 300;                          //number of points a polygon side is divided into

domainHeight 500.0;                     //height of the volume mesh

offset (5000.0 -220000.0 0);            //offset the final mesh by this value

autoOffset on;                          //automatically calculate a good offset value. Overwrites offset.
```

releaseAreaMapping

`releaseAreaMapping`

This utility can be used to set the initial condition (e.g. the initial release zone). This utility reads input from the file `constant/releaseArea`.
Usually the initial condition is set on `0/h` and `0/hentrain`.
The initial velocity `0/Us` is usually set to `constant (0,0,0)`.

Three following forms of release areas can be created:

- Sphere
- Polygon
- Shapefile

Sphere

`sphere`

Spherical release area, as often used for small scale experiments.
For an example see `tutorials/releaseAreaMapping`.
Setting a spherical mass on a slope:

```
fields
{
    h                                   //name of the field
    {
        default 0;                      //default value outside the release area
        regions                         //regions can contain many release areas
        (
            sphereExample               //name of the release area
            {
                type       sphere;      //type sphere for spherical release
                center     (2.0 0.0 3.5);  //centre of the sphere
                r          2.0;         //radius of the sphere
                scale      1.0;         //scale up/down the resulting field
            }
        );
    }
}
```

Polygon

`polygon`

Classic slab release.
For an example see `tutorials/releaseAreaMapping` and `tutorials/entrainment`.
Setting a slab release on a slope:

```
fields
{
    h                                   //name of the field
    {
        default 0;                      //default value outside the release area
        regions                         //regions can contain many release areas
        (
            PolygonWithConstantValue    //name of the release area
            {
                type polygon;           //type polygon for a simple slab
                filltype constant;      //filltype: either constant or linear
                vertices                //list of vertices defining the polygon
                (
                    (0 0 0)
                    (2 0 0)
```

```
                (2 2 0)
                (0 2 0)
            );
            value 0.5;              //value within the polygon (filltype constant)
            offset (5 13 0);        //offset of the polygon
            projectToNormal no;     //convert from vertical height to thickness
        }

        PolygonWithLinearValue      //name of the release area
        {
            type polygon;           //type polygon for a simple slab
            filltype linear;        //filltype: either constant or linear
            vertices                //list of vertices defining the polygon
            (
                (0 0 0)
                (2 0 0)
                (2 2 0)
                (0 2 0)
            );
            valueAtZero 0.2;        //value if field at (x0, y0, z0)
            x0 15.;                 //reference point x-coordinate
            y0 13.;                 //reference point y-coordinate
            z0 0.;                  //reference point z-coordinate
            dfdx 0.1;               //change of field with x
            dfdy 0.1;               //change of field with y
            dfdz 0.;                //change of field with z
            offset (15 13 0);       //offset of the polygon
            projectToNormal no;     //convert from vertical height to thickness
        }
    );
  }
}
```

Shapefile

`shapefile`

Classic slab release.
For an example see `tutorials/releaseAreaMapping` and `tutorials/wolfsgrube`.
Setting a slab release on a slope:

```
fields
{
    h                               //name of the field
    {
        default 0;                  //default value outside the release area
        regions                     //regions can contain many release areas
        (
            ShapefileWithEmbeddedValues   //name of the release area
            {
                type shapefile;           //type shapefile to read POLYGON from an ESRI shapefile
                filltype shapefile;       //fill the polygon. filltype shapefile reads values from shapefile
                filename "data/h0";       //filename of the shapefile without filetype
                fieldname "h0";           //the name of the field in the shapefile which is used for filling
                offset (0 5 0);           //offset of the polygon
                projectToNormal no;       //convert from vertical height to thickness
            }

            ShapefileWithConstantValue    //name of the release area
            {
                type shapefile;           //type shapefile to read POLYGON from an ESRI shapefile
                filltype constant;        //filltype constant fills the polygon with a constant value
                filename "data/h0";       //filename of the shapefile without filetype
                value 0.4;                //constant value to fill the polygin
                offset (0 10 0);          //offset of the polygon
                projectToNormal no;       //convert from vertical height to thickness
            }

            ShapefileWithLinearValue     //name of the release area
            {
                type shapefile;           //type shapefile to read POLYGON from an ESRI shapefile
                filltype linear;          //filltype constant fills the polygon with a linear value
                filename "data/h0";       //filename of the shapefile without filetype
                valueAtZero 0.1;          //value if field at (x0, y0, z0)
                x0 7;                     //reference point x-coordinate, other coordinates are 0
```

```
                dfdx 0.05;              //change of field with x, dfdy = dfdz = 0
                offset (10 5 0);        //offset of the polygon
                projectToNormal no;     //convert from vertical height to thickness
            }
        );
    }
}
```

Rasterfile

`rasterfile`

Import of rasterfiles.

```
fields
{
    h
    {
        default default [0 1 0 0 0 0 0] 0;
        regions
        (
            RasterfileNN
            {
                type rasterfile;
                interpolation nearestneighbor;
                filename "data/h0.asc";
                offset (0 0 0);
            }
    }
}
```

The source code can be found in `applications/utilities/releaseAreaMapping`.

Solver Settings

Simulation Time, Timesteps

Simulation time and time stepping controls can be found in the file `system/control`. See OpenFOAM User Manual for details. Most important settings:

- `endTime`: Time until the solver runs
- `writeInterval`: Intervalls at which results are saved
- `maxCo`: Maximum Courant-number. Recommended value is 1.

Initial and Boundary Conditions

Initial and boundary conditions can be set in the files `0/h`, `0/Us` and `0/hentrain`, similar as in other OpenFOAM solver. The tool `ReleaseAreaMapping` can be used to create appropriate initial conditions.

Transport Properties

Most physical constants can be found in `constant/transportProperties`. Example:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  avalanche                             |
|   \\  /    A nd           | https://develop.openfoam.com/Community/avalanche|
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

pressureFeedback    off;                        //turn curvature and lateral pressure interaction on/off
```

```
explicitDryAreas    on;                             //eliminate dry cells in the linear system

hmin                hmin [ 0 1 0 0 0 0 0]    1e-5;  //binding the height

rho                 rho [ 1 -3  0 0 0 0 0 ]  200.;  //avalanche/snow cover density

u0                  u0 [ 0 1 -1 0 0 0 0]     1e-4;  //small value for the velocity used in divisions

h0                  h0 [ 0 1 0 0 0 0 0]      1e-4;  //small value for the flow height used in divisions

xi                  xi [ 0 0 0 0 0 0 0]      1;     //shape factor for the velocity profile

frictionModel       Voellmy;                        //define the friction model

entrainmentModel    Erosionenergy;                  //define the entrainment model

depositionModel     depositionOff;                  //define the deposition model

VoellmyCoeffs                                       //parameter for the friciton model (Voellmy)
{
    mu              mu [0 0 0 0 0 0 0]       0.26;

    xi              xi [0 1 -2 0 0 0 0]      8650;
}

ErosionenergyCoeffs                                 //parameter for the entrainment moidel (Eroisionenergy)
{
    eb              eb [0 2 -2 0 0 0 0]      11500;
}

// ************************************************************************* //
```

Numerical Schemes

See `system/faSchemes` and the OpenFOAM User Manual. Note that this solver is based on the Finite Area Method. Therefore, the numerical schemes are found in the file **fa**Schemes (instead of **fv**Schemes).

Numerical Solver

See `system/faSolution` and the OpenFOAM User Manual. Note that this solver is based on the Finite Area Method. Therefore, the numerical solution algorithms are found in the file **fa**Solution (instead of **fv**Solution).

Post Processing

Sometimes it is required to rerun functionObjects, either after a distributed (parallel) case was reconstructed or simply because you forgot to add the functionObject. For this usecase **all** solvers have a `-postProcess` flag. With this flag, only the postProcessing steps of the functionObjects will be executed. The source for the postProcessing is read from existing files from the file system.

Example:

```
cd avalanche/tutorials/simpleslope  // change into the directory
./Allrun                            // run the case
faSavageHutterFoam -postProcess     // execute functionObjects again
```

Tutorials

All tutorials contain a `Allrun` script which will conduct all steps required to run a simulation.

simpleslope

Demonstrates:

- faSavageHutterFoam

Example from Rauter and Tukovic (2018), section 6.3. This example demonstrates a popular shallow granular flow test case on a simply curved slope.

wolfsgrube

Demonstrates:

- faSavageHutterFoam
- natural terrain
- shapefile export
- gridfile export
- isoline export

Example from Rauter et al. (2018).

This tutorial shows the recalculation of a real scale avalanche in Tirol/Austria. Find out more about this avalanche in Fischer et al. (2015).
This tutorial applies the solver and some tools of this package. Moreover it is using some python scripts which can be found in the folder `scripts`.

montereycanyon

Demonstrates:

- faParkerFukushimaFoam
- natural terrain

Simplified example similar to Rauter et al. (2019) of a turbdity current in Monterey Canyon.

wolfsgrube_mixed

Demonstrated:

- faTwoLayerAvalancheFoam

The same as the wolfsgruben tutorial, however, set up for faTwoLayerAvalancheFoam, simulating a mixed snow avalanche (dense flow + powder cloud).