051176 - Computational Techniques for Thermochemical Propulsion

Master of Science in Aeronautical Engineering

Wall-Film Modeling in the FV Framework

Prof. **Federico Piscaglia**

Dept. of Aerospace Science and Technology (DAER)

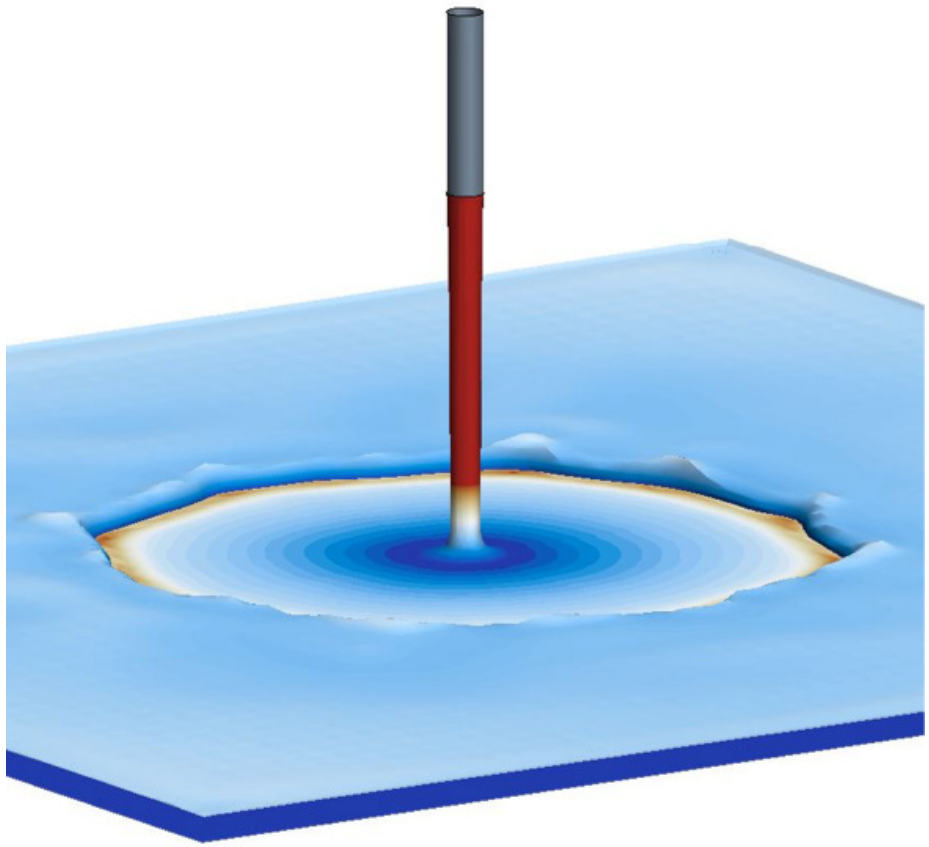
POLITECNICO DI MILANO, Italy

federico.piscaglia@polimi.it

Introduction

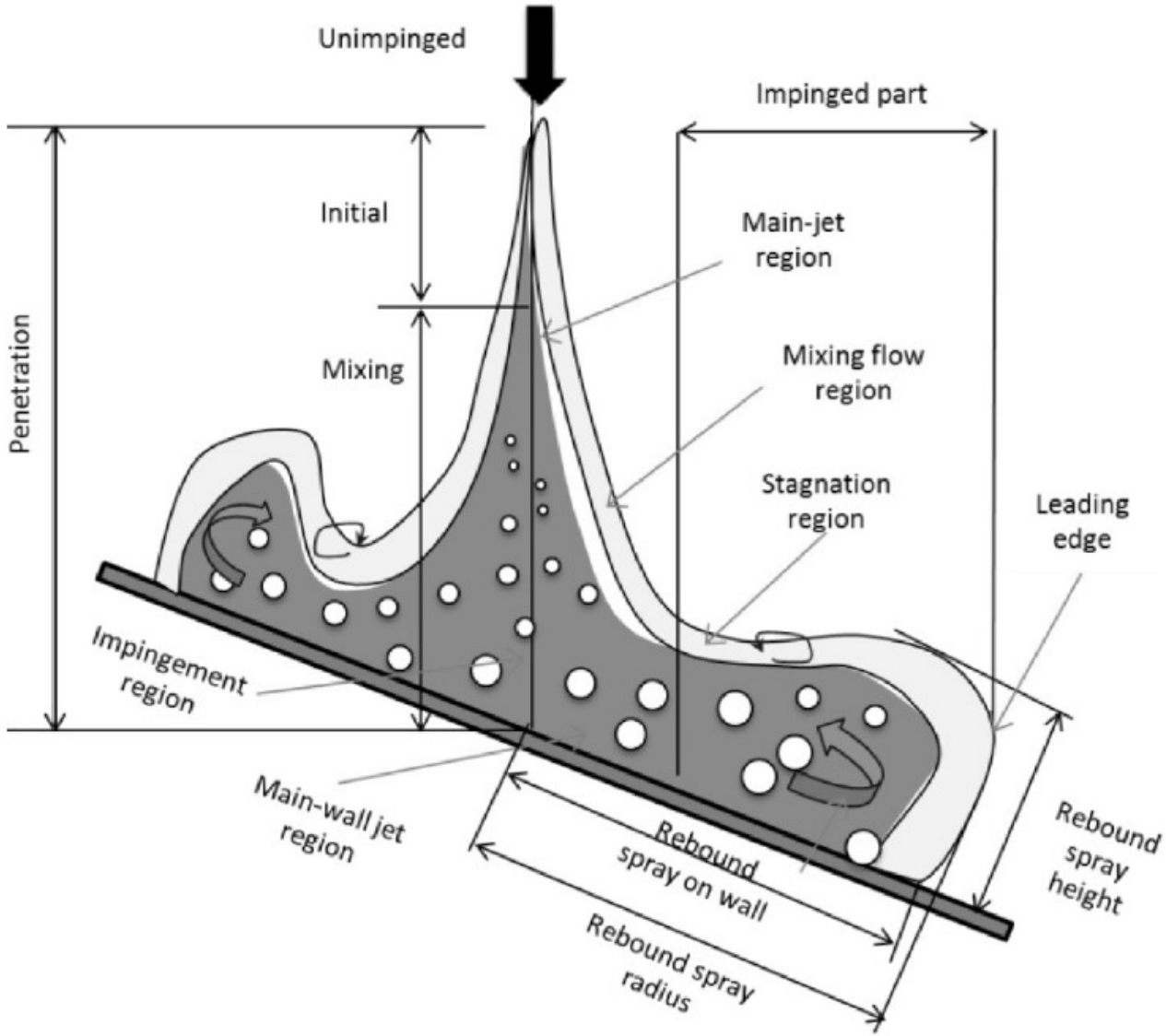
`surfaceFilm` modeling involves the simulation of the evolution of a 2d liquid film over a 3d solid surface, including:

- Impinging Spray
- Heat transfer between the wall, the liquid and the gas phase;
- Fuel evaporation;
- Film defermation, stripping
- particle/wall interaction

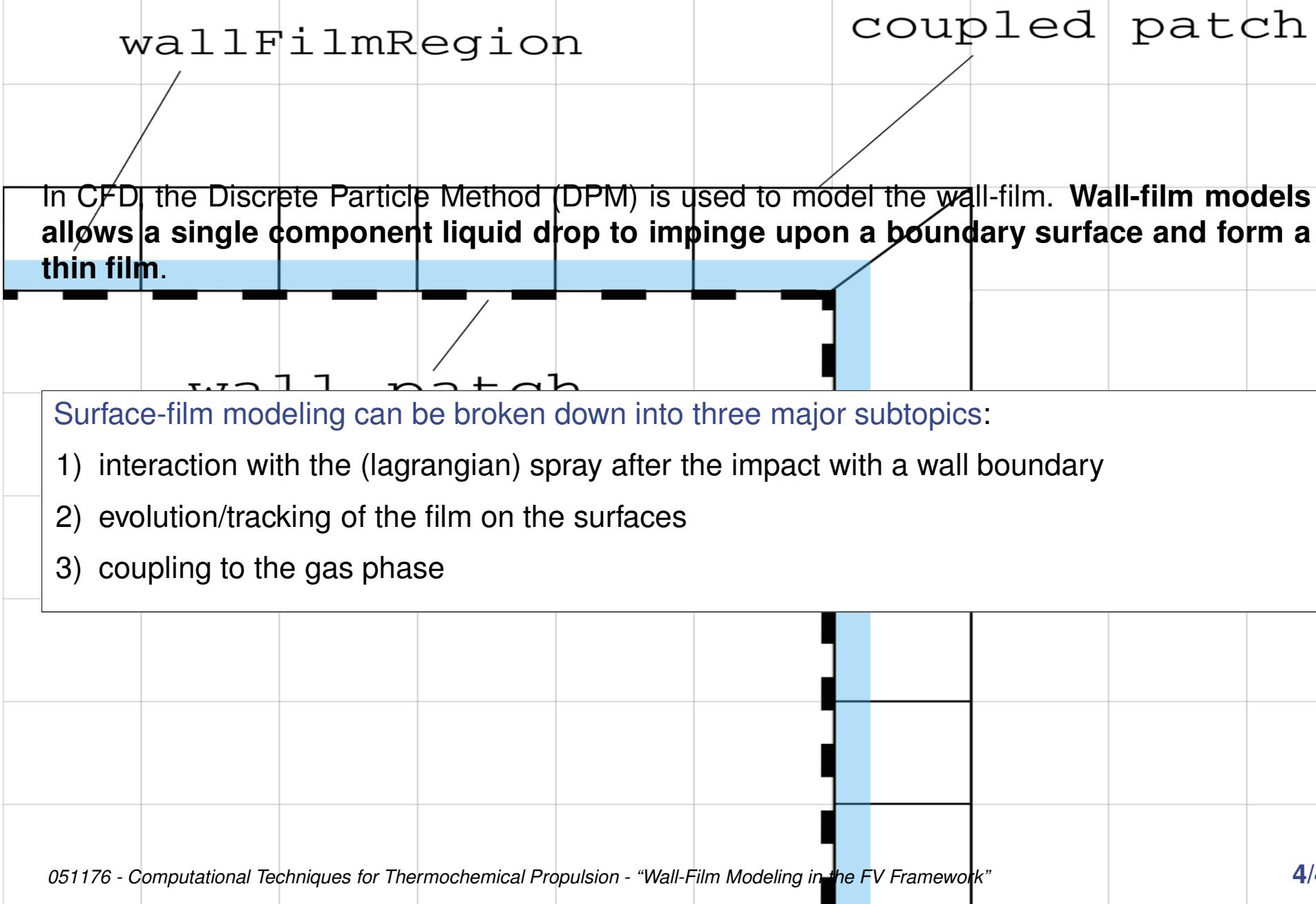


In OpenFOAM-7, the equations of mass, momentum and energy are written and integrated following a **finite volume approach**. Because of the variation of mass inside a control volume these equations resemble a virtual compressibility.

Evolution of the liquid wall-film



surfaceFilm modeling

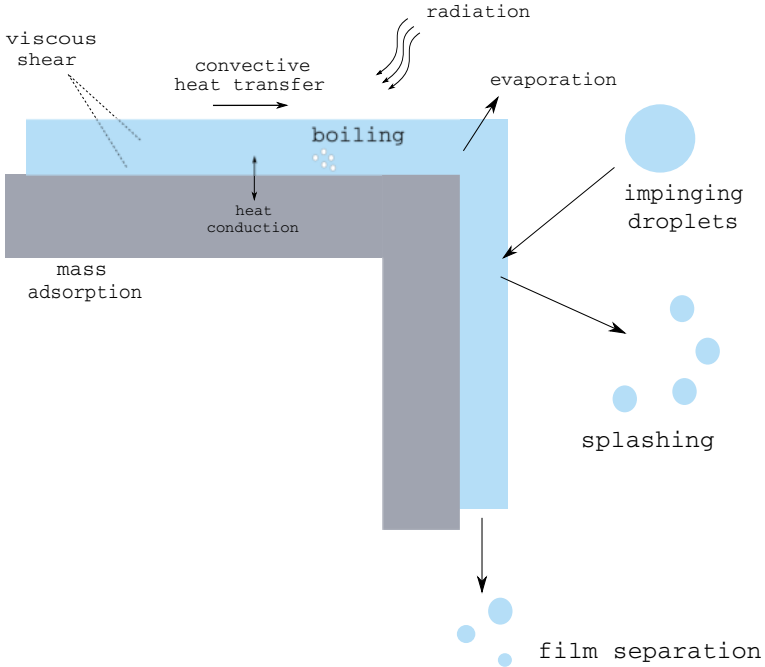


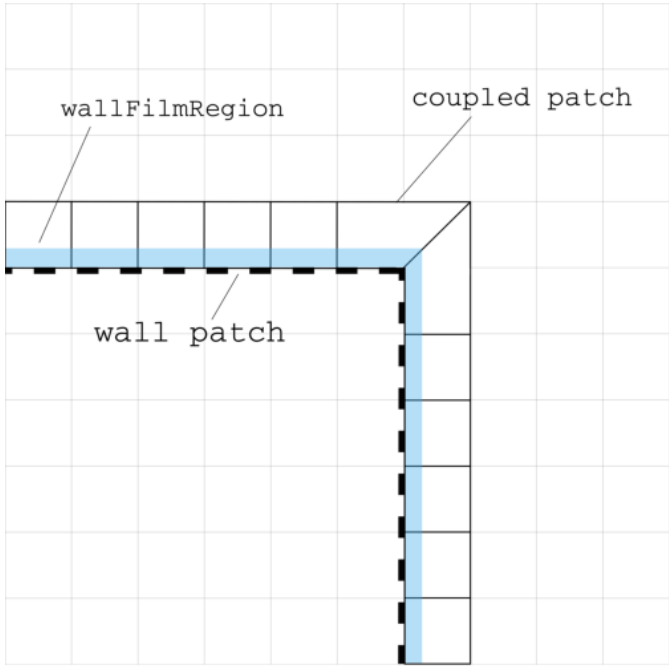
In CFD, the Discrete Particle Method (DPM) is used to model the wall-film. **Wall-film models allows a single component liquid drop to impinge upon a boundary surface and form a thin film**.

Surface-film modeling can be broken down into three major subtopics:

1) interaction with the (lagrangian) spray after the impact with a wall boundary
2) evolution/tracking of the film on the surfaces
3) coupling to the gas phase

surfaceFilm - Modeling Assumptions



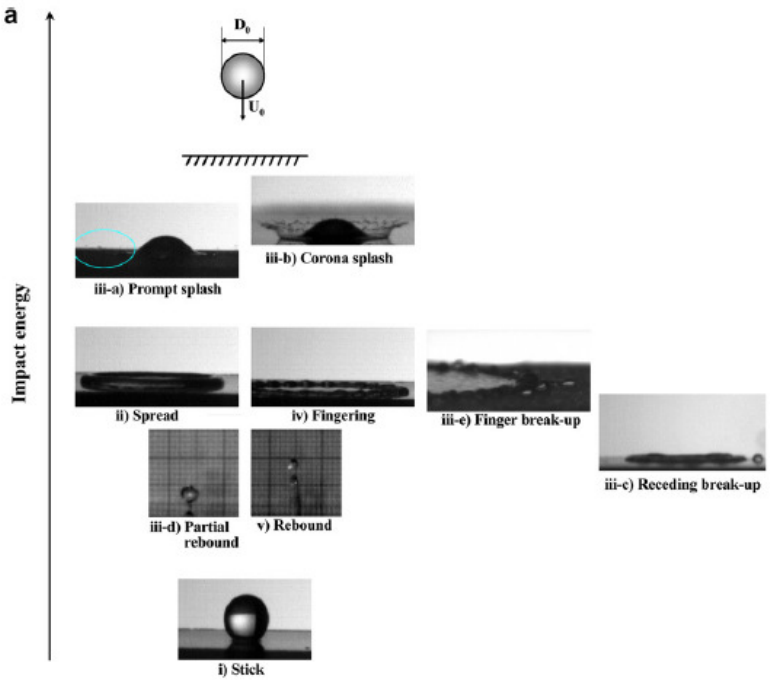


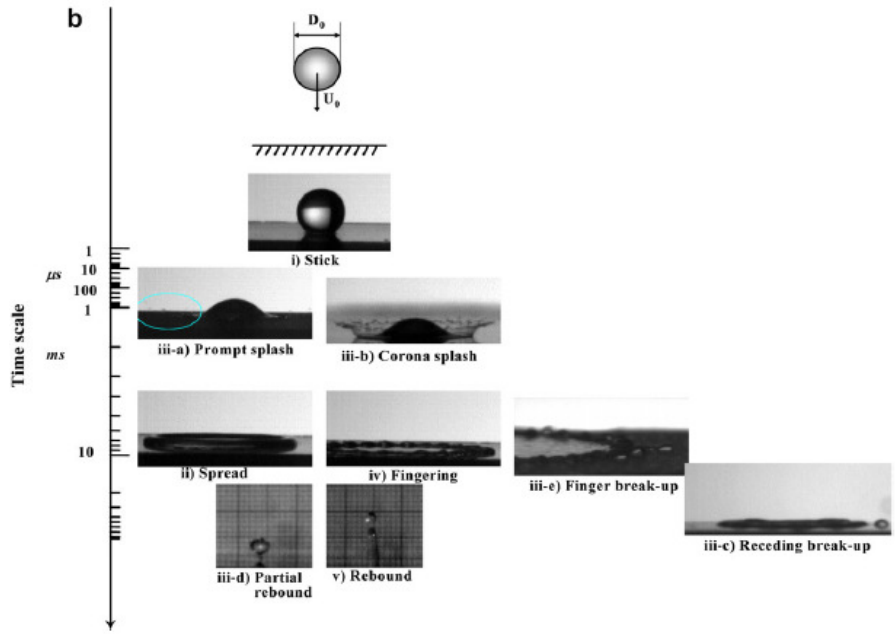
Main assumptions/restrictions for the wall-film model are:

- the layer is thin $\rightarrow$ linear velocity profile in the film;
- particle change their temperature in the film relatively slowly so that an analytical integration scheme can be utilized;
- direct contact between the `surfaceFilm` and the wall surface $\rightarrow$ the heat transfer from the wall to the film takes place by conduction;
- film temperature never exceeds the boiling temperature for the liquid;
- the simulation is transient.

1) spray/wall interaction

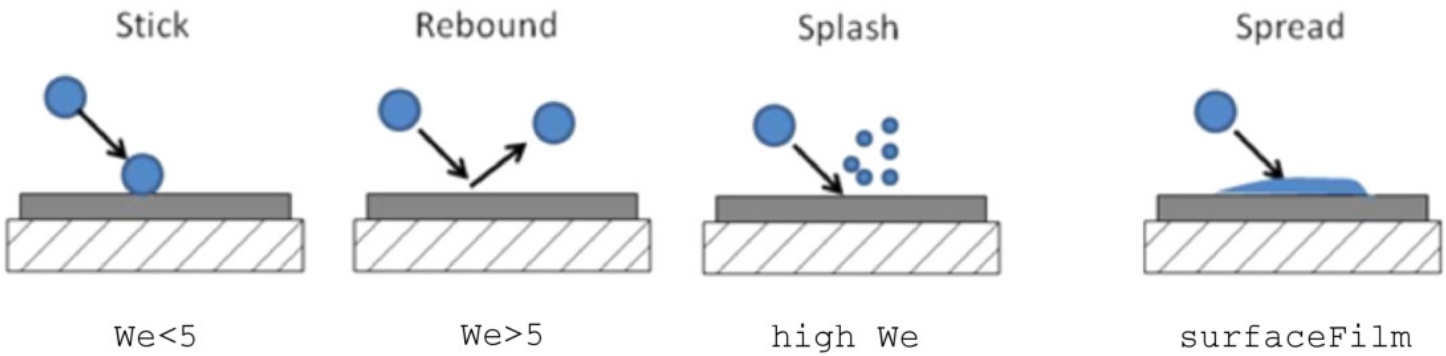
Particle/Wall Interaction





Prompt splash occurs as the inertial forces overcome capillary effects.

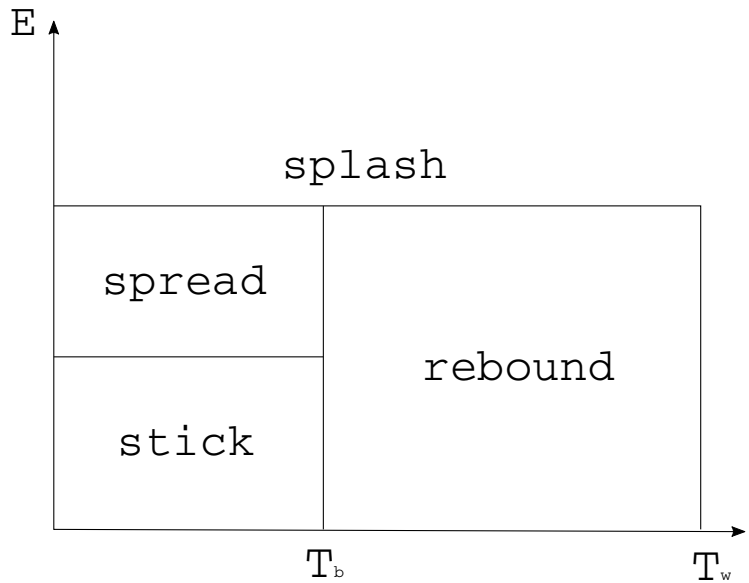
surfaceFilm modeling



The wall interaction is based on the work of Stanton and O'Rourke, where the regimes are calculated for a drop-wall interaction based on local information.

Four regimes are defined:

- stick
- rebound
- spread
- splash



Distinction between the different regimes is based on the impact energy and wall temperature of the droplets.

Splash model by O'Rourke-Amsden

The criteria by which the regimes are partitioned are based on the **impact energy** and the **boiling temperature of the liquid**. The impact energy is defined by

$$\text{Splash:} \qquad E^2 = \frac{\rho V_r^2 D}{\sigma} \left(\frac{1}{\min(\frac{h_0}{D}, 1) + \delta_{bl}/D} \right) > (57.7)^2$$

where:

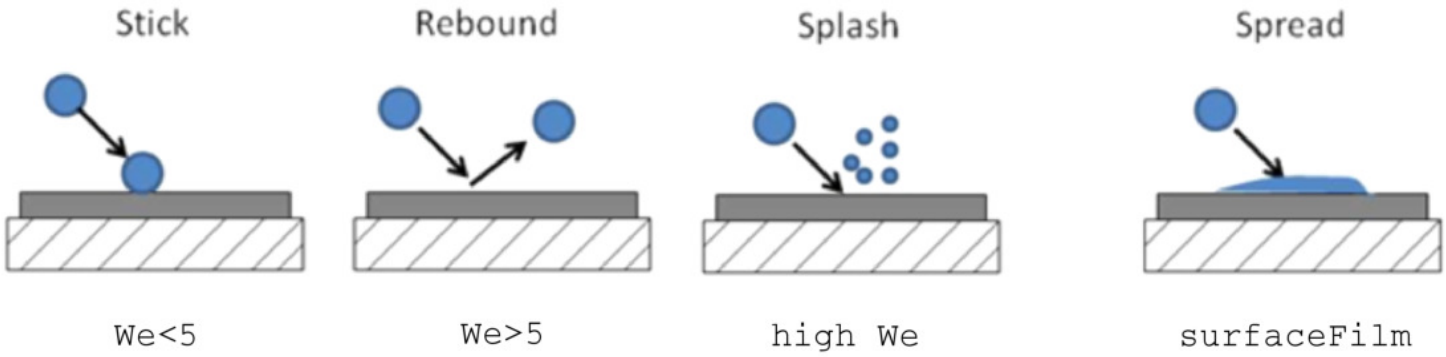
- h_0 is the film thickness;
- ρ is the liquid density;
- $V_r^2 = (V_p - V_w)^2$ is the relative velocity of the particle in the boundary cell;
- D is the diameter of the droplet;
- σ is the surface tension of the liquid;
- δ_{bl} is a boundary layer thickness, defined as

$$\delta_{bl} = \frac{D}{\sqrt{\text{Re}}}$$

being the Reynolds number is defined as $Re = \rho V_r D/\mu$.

By the definition of E, the presence of the film on the wall (throgh the term h_0) suppresses the splash, but does not give unphysical results when the film height approaches zero.

Particle/Wall Interaction Models



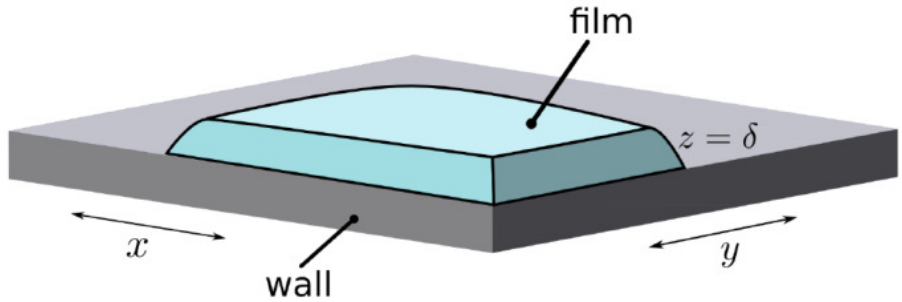
- The **sticking regime** is applied when the dimensionless energy E is less than 16, and the particle velocity is set equal to the wall velocity.
- **rebounding from the wall** occurs if the wall temperature is above the boiling temperature of the liquid and the energy of the particle is smaller than the critical impact energy (E_{cr}). For the rebound regime, the particle rebounds with the following coefficient of restitution:

$$e = 0.993 - 1.76\Theta_I + 1.56\Theta_I^2 - 0.49\Theta_I^3$$

  where Θ_I is the impingement angle as measured from the wall surface.
- In the **spreading regime**, the initial direction and velocity of the particle are set using the wall-jet model, where the probability of the drop having a particular direction along the surface is given by an analogy of an inviscid liquid jet with an empirically defined radial dependence for the momentum flux.
- **Splashing** occurs when the impingement energy is above a critical energy threshold, defined as $E_{cr} = 57.7$. The number of splashed droplet parcels is set in the Wall boundary condition dialog box with a default number of 2.

2) evolution/tracking of the `surfaceFilm`

Integrated Film-Transport Equations



- wall-normal velocity is zero;
- wall-tangential diffusion (momentum, energy) negligible compared to wall-normal diffusion;
- transport equations can be integrated in the wall-normal direction;
- the resulting governing equations are:

$$\texttt{Continuity:} \quad \frac{\partial \rho\delta}{\partial t} + \nabla_s \cdot (\rho\delta \boldsymbol{U}) = \boldsymbol{S}_\delta$$

$$\texttt{Momentum:} \quad \frac{\partial \rho\delta \boldsymbol{U}}{\partial t} + \nabla_s \cdot (\rho\delta \boldsymbol{U}\boldsymbol{U}) = -\delta\nabla_s p + \boldsymbol{S}'_\delta$$

$$\texttt{Energy:} \quad \frac{\partial \rho\delta h}{\partial t} + \nabla_s \cdot (\rho\delta \boldsymbol{U} h) = \boldsymbol{S}''_\delta$$

Mass conservation

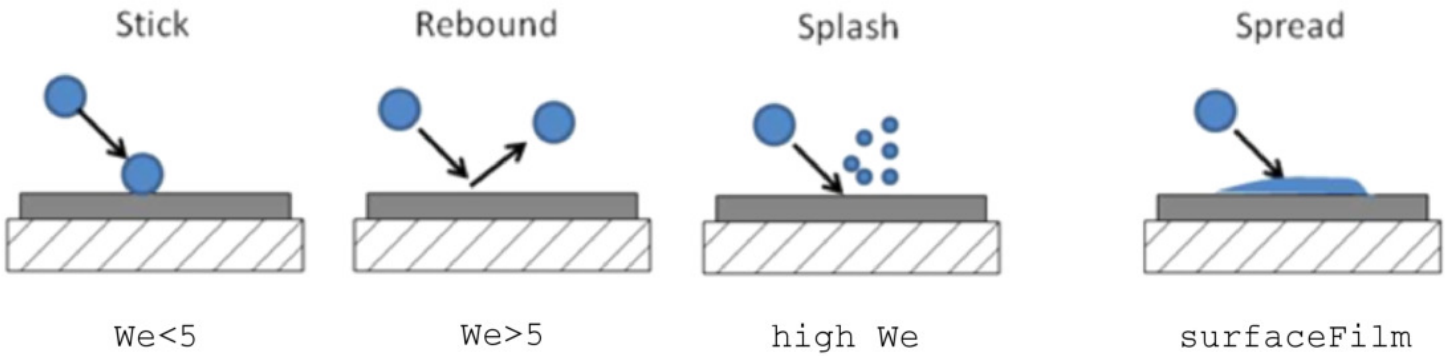
The integrated mass continuity equation is defined as:

$$\frac{\partial \rho \delta}{\partial t} + \nabla_s \cdot (\rho \delta \boldsymbol{U}) = \boldsymbol{S}_\delta$$

where:

- ρ is the liquid density
- δ is the film thickness
- $\boldsymbol{U}$ is the film velocity
- ∇_s is the vector differential operator tangential to the surface $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$
- $S_{\rho\delta}$ is the mass source per unit wall area due to impingement, splashing, evaporation, absorption. into the solid, and film separation.

Mass conservation: sink terms

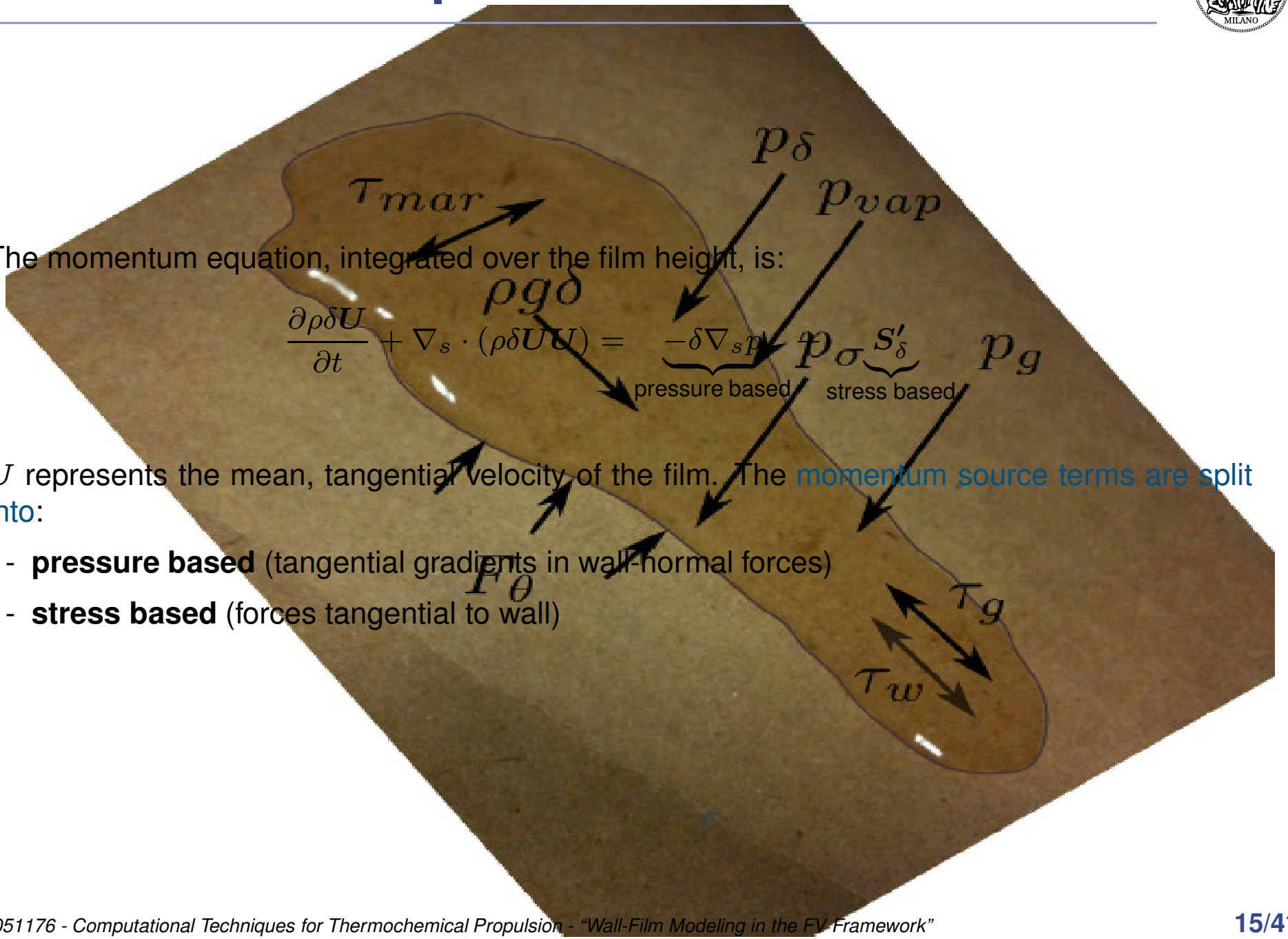


In the continuity equation:

$$\frac{\partial \rho\delta}{\partial t} + \nabla_s \cdot (\rho\delta \boldsymbol{U}) = \boldsymbol{S}_\delta$$

$S_{\rho\delta}$ is the mass source per unit wall area due to impingement, splashing, evaporation, absorption into the solid, and film separation.

Momentum Transport



The momentum equation, integrated over the film height, is:

$$\frac{\partial \rho \delta \boldsymbol{U}}{\partial t} + \nabla_s \cdot (\rho \delta \boldsymbol{U}\boldsymbol{U}) = \underbrace{-\delta \nabla_s p}_{\text{pressure based}} + \underbrace{\nabla_s \cdot \boldsymbol{S}'_\delta}_{\text{stress based}} + \rho \boldsymbol{g} \delta + \boldsymbol{\tau}_{mar} + \boldsymbol{\tau}_{vap} + \boldsymbol{\tau}_\sigma + \boldsymbol{\tau}_g + \boldsymbol{\tau}_\theta + \boldsymbol{\tau}_w$$

$\boldsymbol{U}$ represents the mean, tangential velocity of the film. The momentum source terms are split into:

- **pressure based** (tangential gradients in wall-normal forces)
- **stress based** (forces tangential to wall)

Momentum Transport

$$\frac{\partial \rho\delta \boldsymbol{U}}{\partial t} + \nabla_s \cdot (\rho\delta \boldsymbol{U}\boldsymbol{U}) = \underbrace{-\delta \nabla_s p}_{\text{pressure based}} + \underbrace{\boldsymbol{S}'_\delta}_{\text{stress based}}$$

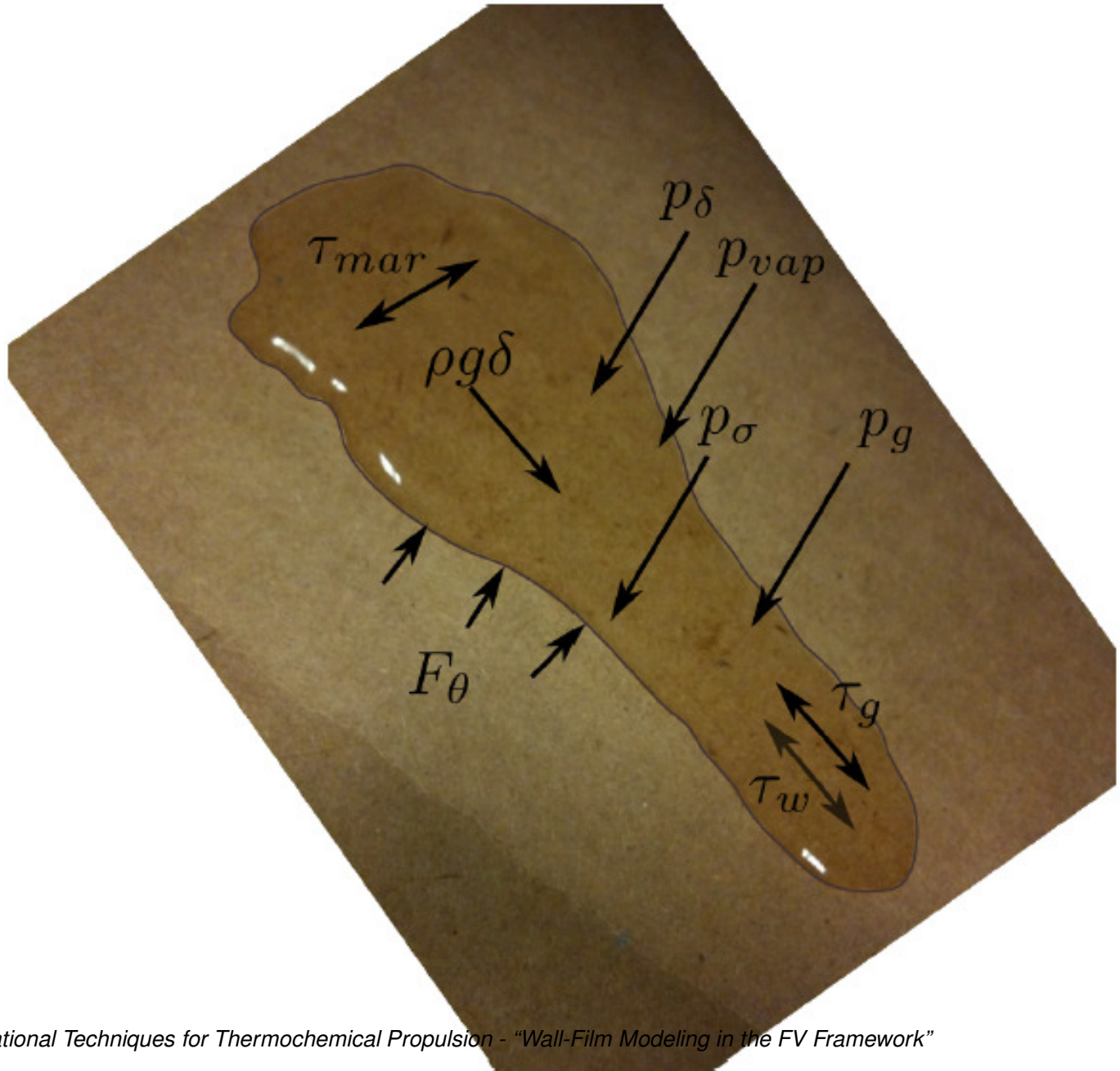
1) The **pressure-based term**, p , is comprised of forces in the wall-normal direction and consists of:
   - capillary effects (p_σ)
   - hydrostatic pressure head (p_δ)
   - local gas-phase pressure (p_g)

$$p = p_\sigma + p_\delta + p_g$$

2) The **stress-based term**, $\boldsymbol{S}_{\rho\delta\boldsymbol{U}}$, includes:
   - the viscous shear stresses: τ_g, τ_w
   - gravity body force: $\rho \boldsymbol{g}_t \delta$
   - stress related contact-angle force τ_θ

$$\boldsymbol{S}_{\rho\delta\boldsymbol{U}} = \tau_g + \tau_w + \rho \boldsymbol{g}_t \delta + \tau_\delta$$

Momentum Transport



Pressure-based (source) terms

- **Capillary pressure**: p_σ is the pressure component due to surface tension based on the curvature of the film surface and represented by the Laplace-Young boundary condition at the air-fluid interface. Using the Laplacian of film thickness to estimate the curvature, the term for the pressure contribution of surface tension is given as

$$p_\sigma = -\sigma \nabla_s^2 \delta$$

  where σ is the surface tension and ∇_s^2 approximates the curvature of the liquid surface. **This relationship is valid only f or surfaces with slight curvatures.**

- The **hydrostatic pressure**, p_δ is the pressure component due to hydrostatic pressure head, and is given by

$$p_\delta = -\rho(\boldsymbol{n} \cdot \boldsymbol{g})\delta$$

  where $\boldsymbol{g}$ is the gravity force vector and n is the surface normal vector. This pressure term is eliminated for purely vertical surfaces $(\boldsymbol{n} \cdot \boldsymbol{g})$.

- The **gravity body force term** is represented by $\rho \boldsymbol{g} \delta$ where $\boldsymbol{g}$ denotes the gravity components tangential to the wall. This term is zero for horizontal surfaces, and is the main driving force for flow over vertical and inclined surfaces.

Stress-based (source) terms

- **Shear stress**: the shear stress terms τ_g and $\tau_w$represent the shear at the film-gas interface and the fil m-wall interface respectively.
  - In most cases for film flow in the presence of a quiescent gas, $\tau_g << \tau_w$ because gas-phase velocities are typically very low and also because the viscosity of the gas is low compared to that of the liquid.
  - The film-wall stress term, τ_w, is modeled based on an assumed velocity profile in the wall-normal direction. Assuming laminar flow and taking the velocity at the wall to be zero and the velocity gradient at the gas interface to be zero, the expression for the velocity distribution through the film can be shown to be

$$\boldsymbol{u}_z = \frac{3\boldsymbol{U}}{\delta}\left(z - \frac{z^2}{2\delta}\right)$$

    where $\boldsymbol{U}$ represents the mean film velocity, z represents the surface-normal coordinate, and δ represents the film thickness. Employing the quadratic velocity profile assumption, the shear stress at the wall is found as

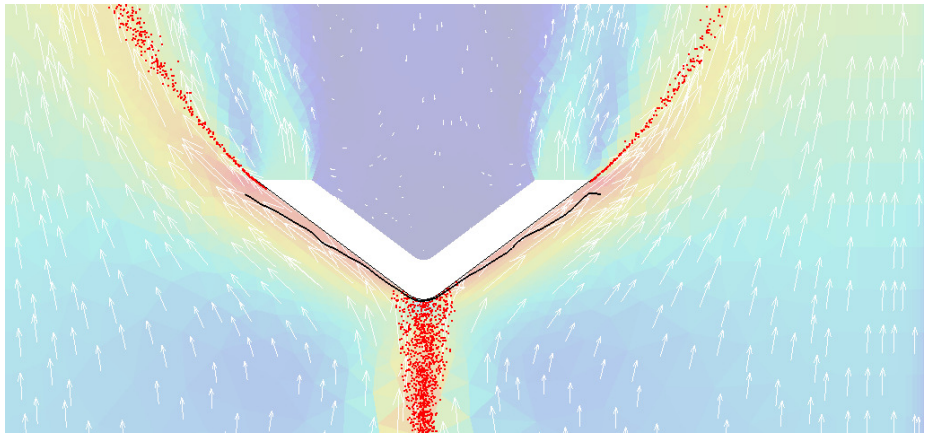
$$\tau_w = -\mu\frac{3\boldsymbol{U}}{\delta}$$

- The **partial wetting behavior** finally involves the treatment of the contact line.

`surfaceFilm` modeling in the FV-framework

surfaceFilm modeling in OpenFOAM

Wall film modeling in CFD softwares is a challenge because of the complex formation of the liquid film at the wall and also the multiphase treatment required to handle both gas and liquid.

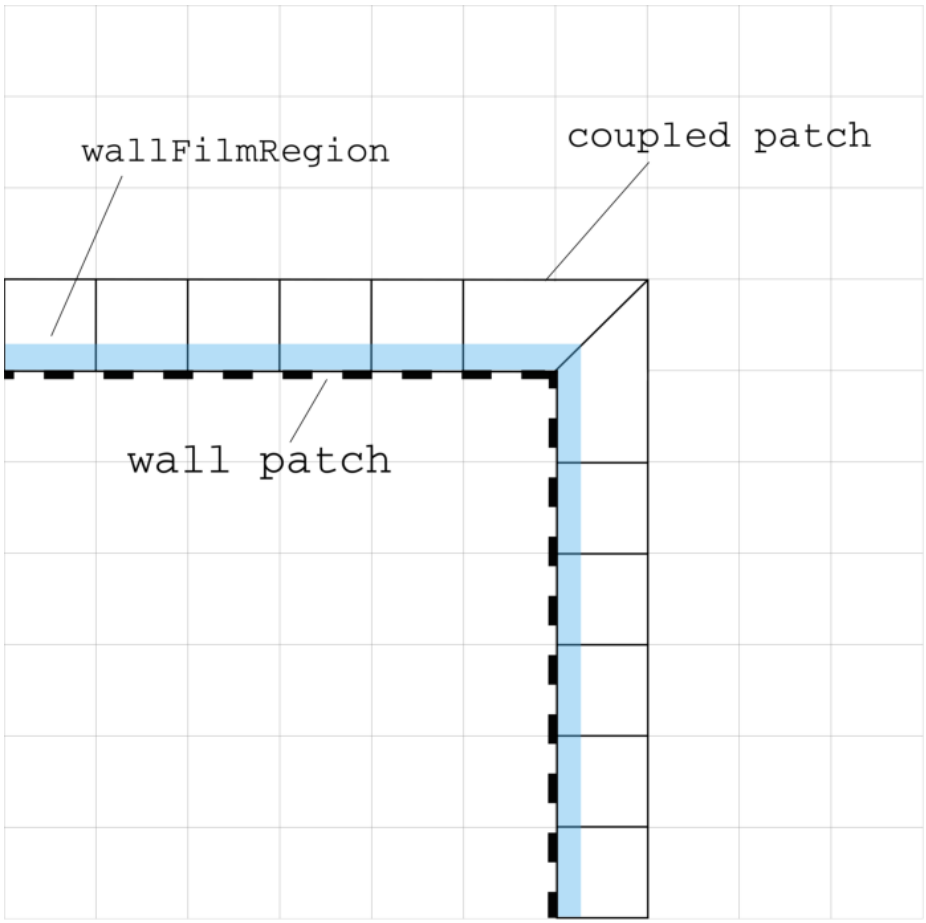


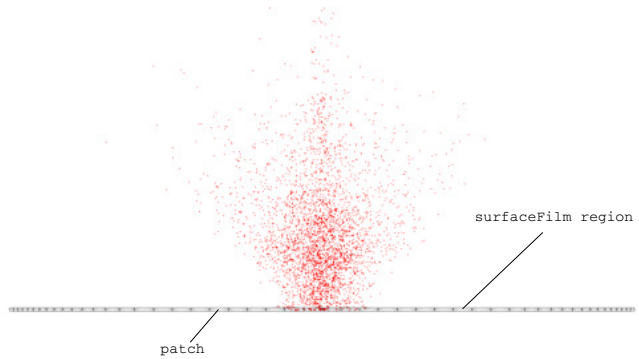
In OpenFOAM the wall film is modeled to be in an own external mesh region extruded from the fluid region.

- **This mesh region is one layer thick** and the flow perpendicular to the surface is neglected which imply a two-dimensional flow along the wall.
- **The flow is modeled with a continuous Eulerian phase** with the momentum, continuity and the energy equation. The thickness of the liquid film is derived from the momentum and continuity equation.

The Baffle layer

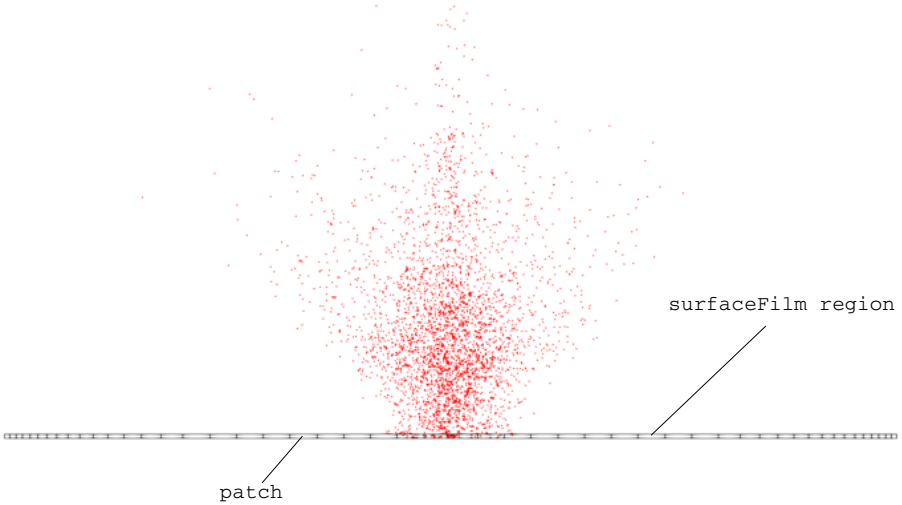
`surfaceFilm` modeling in OpenFOAM: equations of **mass**, **momentum** and **energy** are written and integrated in a FV framework approach. Because of the variation of mass inside a control volume these equations resemble a virtual compressibility.





- Extruded, 2D Mesh
- Uses 3D FV operations
- Solution variables mapped to/from gas phase mesh
- Source terms handle interfacial transport
- Can be solve in “stand-alone” mode

surfaceFilm modeling: some notes



$$\frac{\partial \delta}{\partial t} + \frac{1}{A_w} \sum_{i}^{\text{nSides}} (\phi \delta_i l_i) = \frac{\boldsymbol{S}_d}{\rho A_w}$$

- The time derivative of film height in a control volume is related to the convective fluxes and to the interaction with gas and spray through the source terms S_d, which represents the interaction between the lagrangian particles and the surfaceFilm model (droplet entrainment/release).
- In the wall film region in OpenFOAM The connection between a fluid region and a wall film region is carried out via the settings for the boundary and is handled by an extruted 2D region, which is created by `extrudeToRegionMesh`.

Particle/Wall Interaction in OpenFOAM

In OpenFOAM-7, particle/wall interaction models are included in the class:

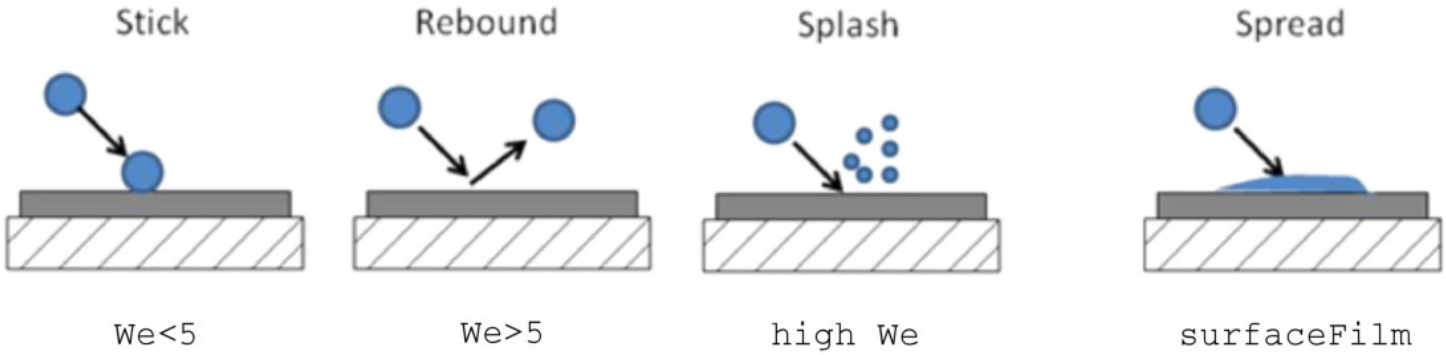
`ThermoSurfaceFilm`

which is part of the `$FOAM_SRC/lagrangian` class:

`$FOAM_SRC/lagrangian/intermediate/submodels/Thermodynamic/SurfaceFilmModel/ThermoSurfaceFilm`

```
Class
    Foam::ThermoSurfaceFilm

Description
    Thermo parcel surface film model.

    Responsible for:
    - injecting parcels from the film model into the cloud, e.g. for dripping
    - parcel interaction with the film, e.g absorb, bounce, splash

    Splash model references:

        Bai and Gosman, 'Mathematical modelling of wall films formed by
        impinging sprays', SAE 960626, 1996

        Bai et al, 'Modelling of gasoline spray impingement', Atom. Sprays,
        vol 12, pp 1-27, 2002
```

ThermoSurfaceFilm<CloudType>



The abstract class `ThermoSurfaceFilm` includes the following particle/wall interaction models

- `absorbInteraction`
- `bounceInteraction`
- `wetSplashInteraction`
- `drySplashInteraction`
- `splashInteraction`

ThermoSurfaceFilm<CloudType>

ThermoSurfaceFilm<CloudType>::**absorbInteraction**

```
//- Absorb parcel into film
void absorbInteraction
(
    regionModels::surfaceFilmModels::surfaceFilmRegionModel&,
    const parcelType& p,
    const polyPatch& pp,
    const label facei,
    const scalar mass,
    bool& keepParticle
);
```

ThermoSurfaceFilm<CloudType>::**bounceInteraction**

```
//- Bounce parcel (flip parcel normal velocity)
void bounceInteraction
(
    parcelType& p,
    const polyPatch& pp,
    const label facei,
    bool& keepParticle
) const;
```

ThermoSurfaceFilm<CloudType>

ThermoSurfaceFilm<CloudType>::**drySplashInteraction**

```
//- Parcel interaction with dry surface
void drySplashInteraction
(
    regionModels::surfaceFilmModels::surfaceFilmRegionModel&,
    const parcelType& p,
    const polyPatch& pp,
    const label facei,
    bool& keepParticle
);
```

ThermoSurfaceFilm<CloudType>::**wetSplashInteraction**

```
//- Parcel interaction with wetted surface
void wetSplashInteraction
(
    regionModels::surfaceFilmModels::surfaceFilmRegionModel&,
    parcelType& p,
    const polyPatch& pp,
    const label facei,
    bool& keepParticle
);
```

splashInteraction

ThermoSurfaceFilm<CloudType>::**splashInteraction**

```
//- Bai parcel splash interaction model
void splashInteraction
(
    regionModels::surfaceFilmModels::surfaceFilmRegionModel&,
    const parcelType& p,
    const polyPatch& pp,
    const label facei,
    const scalar mRatio,
    const scalar We,
    const scalar Wec,
    const scalar sigma,
    bool& keepParticle
);
```

regionModels::regionModel::evolve()

```
//- from: $FOAM_SOLVERS/lagrangian/reactingParcelFoam/reactingParcelFoam.C
parcels.evolve();
surfaceFilm.evolve();
...
```

In the solvers supporting wall-film modeling, lagrangian/wallFilm interaction is handled in two classes:

1) calculation of the lagrangian/wall interation, in `$FOAM_SRC/lagrangian`, which is called at the solver level by:

   `parcels.evolve();`

2) calculation of the evolution of the surfaceFilm is calculated in the class `surfaceFilmModel` by the command in the solver:

   `surfaceFilm.evolve();`

   being the surfaceFilm defined as:

```
autoPtr<regionModels::surfaceFilmModel> tsurfaceFilm
(
    regionModels::surfaceFilmModel::New(mesh, g)
);
```

SprayCloud

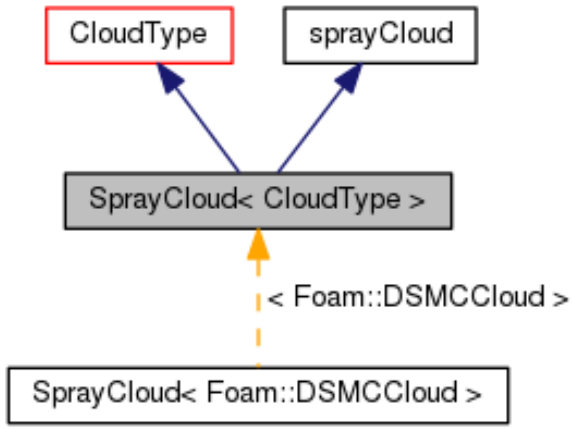
Being Lagrangian parcels defined as:

```
basicSprayCloud parcels
(
    "sprayCloud",
    rho,
    U,
    g,
    slgThermo
);
```

When lagrangian parcels are transported (see `basicReactingMultiphaseCloud.H`) patch-Interaction models are inherently activated.

```
    typedef SprayCloud
    <
        ReactingCloud
        <
            ThermoCloud
            <
                KinematicCloud
                <
                    Cloud
                    <
                        basicSprayParcel
                    >
                >
            >
        >
    > basicSprayCloud;
}
```

lagrangian/surfaceFilm interaction




```
template<class CloudType>
void Foam::SprayCloud<CloudType>::evolve()
{
    if (this->solution().canEvolve())
    {
        typename parcelType::template
            TrackingData<SprayCloud<CloudType>> td(*this);

        this->solve(td);
    }
}
```

where “td” is the transported parcel, with its own properties.

lagrangian/surfaceFilm interaction

The line "`this->solve(td)`" calls the public function of the inherited class `KinematicCloud`:

```
template<class CloudType>
template<class TrackData>
void Foam::KinematicCloud<CloudType>::solve(TrackData& td)
{
    ...
    td.cloud().preEvolve();

    evolveCloud(td);

    if (solution_.coupled())
    {
        td.cloud().scaleSources();
    }
    ...
}
```

lagrangian/surfaceFilm interaction

In the file:

```
$FOAM_SRC/lagrangian/intermediate/parcels/Templates/KinematicParcel/KinematicParcel.C
```

the function `evolveCloud` is called:

```
template<class CloudType>
template<class TrackData>
void Foam::KinematicCloud<CloudType>::evolveCloud(TrackData& td)
{
     ...
     // Assume that motion will update the cellOccupancy as necessary
     // before it is required.
     td.cloud().motion(td);
```

which in turn calls:

```
template<class CloudType>
template<class TrackData>
void Foam::KinematicCloud<CloudType>::motion(TrackData& td)
{
    td.part() = TrackData::tpLinearTrack;
    CloudType::move(td,  solution_.trackTime());

    updateCellOccupancy();
}
```

lagrangian/surfaceFilm interaction

```
template<class CloudType>
template<class TrackData>
void Foam::KinematicCloud<CloudType>::motion(TrackData& td)
{
    td.part() = TrackData::tpLinearTrack;
    CloudType::move(td,  solution_.trackTime());

    updateCellOccupancy();
}
```

`CloudType` means `Cloud` if the `basicSprayCloud` class is used in the solver. Hence, the command:

```
CloudType::move(td, solution_.trackTime());
```

recalls:

```
template<class ParticleType>
template<class TrackData>
void Foam::Cloud<ParticleType>::move(TrackData& td, const scalar trackTime)
{
   //from Cloud.C, line 255:
   ...
   // Move the particle
   bool keepParticle = p.move(td, trackTime);
   ...
}
```

From slide 30, `ParticleType` is defined as `basicSprayParcel`.

lagrangian/surfaceFilm interaction

Since basicSprayParcel inherits the properties of KinematicParcel, it follows that the function move of the class KinematicParcel is called.
In:

```
$FOAM_SRC/lagrangian/intermediate/parcels/Templates/KinematicParcel/KinematicParcel.C
```

at line 361 the particle-wall interaction is calculated:

```
template<class ParcelType>
template<class TrackCloudType>
bool Foam::KinematicParcel<ParcelType>::move
{
    ...
    if (p.onFace() && ttd.keepParticle)
    {
        p.hitFace(s, cloud, ttd);
    }
    ...
}
```

Evolution of the `surfaceFilm`

Evolution of the `surfaceFilm` region

Evolution of the `surfaceFilm` region:

Equations of the liquid in the single-cell layer surface film model are solved in the film region. At the solver level, the public member function is called:

```
surfaceFilm.evolve()
```

Function `evolve()` solves the governing equations (mass, momentum, energy) for the wall-film.

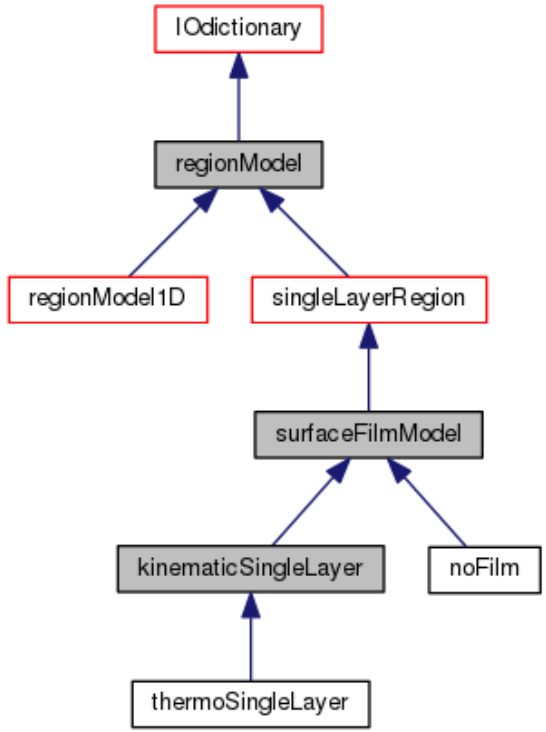
regionModels::regionModel::evolve()

```
1
2   //- from: $FOAM_SOLVERS/lagrangian/reactingParcelFoam/reactingParcelFoam.C
3
4   ...
5
6   runTime++;
7
8   ...
9
10  parcels.evolve();
11  surfaceFilm.evolve();
12
13  if (solvePrimaryRegion)
14  {
15    ...
```

In the solvers supporting wall-film modeling:

- the solver (e.g. `reactingParcelFoam`) calls the public member function `surfaceFilm.evolve()` (see line 7).
- The function `evolve()` is **virtual**; it is implemented in the class `regionModel` (see the Doxygen glyph, next slide) and in the inherited classes.

regionModels::regionModel::evolve()



- The function `evolve()` in `regionModel` calls the **virtual** function `evolveRegion()` to calculate the evolution of the liquid wall film:

```
virtual void Foam::regionModels::regionModel::evolveRegion()
```

- Any inherited class (e.g. `thermoSingleLayer`) will have its own specialization of the function `evolveRegion()`.

thermoSingleLayer::evolveRegion()

```
void thermoSingleLayer::evolveRegion()
{
    //- update submodels and calculate sink terms
    ...

    // Solve continuity for deltaRho_
    solveContinuity();

    for (int oCorr=1; oCorr<=nOuterCorr_; oCorr++)
    {
        // Explicit pressure source contribution
        tmp<volScalarField> tpu(this->pu());

        // Implicit pressure source coefficient
        tmp<volScalarField> tpp(this->pp());

        tmp<fvVectorMatrix> UEqn = solveMomentum(tpu(), tpp());

        // Solve energy for hs_ - also updates thermo
        solveEnergy();

        // Film thickness correction loop
        for (int corr=1; corr<=nCorr_; corr++)
        {
            // Solve thickness for delta_
            solveThickness(tpu(), tpp(), UEqn());
        }
    }

    ...
}
```

Thank you for your attention!

contact: federico.piscaglia@polimi.it