Delivery date:

28-01-2016

Authors:

Patrick D. Anderson,
Daniele Marchisio,
Christos Mitrias,
Mohsen Karimi

MoDeNa

Deliverable D3.3

Rheology, coalescence and nucleation models

Principal investigators:

Patrick D. Anderson, TU/e(NL)
pda@tue.nl

Project coordinator:

Heinz A Preisig

heinz.preisig@chemeng.ntnu.no

empty page

Contents

1 Model Description

Both tools are using a similar model which is a Stokes flow model for incompressible fluids. For the Coalescence Kernel tool we explicitly describe both components while for the Rheology tool we use voids. For both phases, i.e. the bubbles and the outer fluid, at this stage, it is assumed that inertia can be neglected and that the volume is constant. As a result the mass and the momentum balance for both phases reduce to

$$\nabla \cdot \boldsymbol{u} = 0$$
$$\nabla \cdot \boldsymbol{\sigma} = 0$$

the $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ are the velocity vector and the stress tensor. For fluids the stress tensor $\boldsymbol{\sigma}$ can be written as:

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{T}$$

where p is the pressure, $\boldsymbol{I}$ is the identity tensor and $\boldsymbol{T}$ is the extra stress tensor. The model is capable of simulating fluids with different types of rheology, i.e. also viscoelastic fluid behavior, but for now only the case of Newtonian fluid is studied. Therefore, $\boldsymbol{T}$ can be expressed as:

$$\boldsymbol{T} = 2\eta_i \boldsymbol{D}$$

with $\eta_i$the viscosity in either one of the two phases and $\boldsymbol{D}$ the symmetric part of the velocity gradient tensor.

The interface is assumed to be a material interface with boundary conditions

$$(\boldsymbol{\sigma} \cdot \boldsymbol{n})_m - (\boldsymbol{\sigma} \cdot \boldsymbol{n})_b = \Gamma\kappa\boldsymbol{n}$$

and

$$\boldsymbol{u}_m = \boldsymbol{u}_b$$

where the subscripts m and b denote the matrix fluid and bubble phase.

For the rheology tool we use voids and we impose the their volume to be constant. The boundary conditions on the interface reduce to

$$(\boldsymbol{\sigma} \cdot \boldsymbol{n})_m - = \Gamma\kappa\boldsymbol{n}$$

For more detailed derivation we refer to Deliverable 3.2.

The boundary of a single phase or the sharp interface between two phases can be described by a moving curvilinear coordinate system given by

$$\boldsymbol{x} = \tilde{x}(\boldsymbol{\xi}, t)$$

where $\boldsymbol{\xi} = (\xi_1, \xi_2)$are the curvilinear coordinates and $\tilde{x}$ is the function that maps the coordinates $\boldsymbol{\xi}$ onto the spatial coordinates $\boldsymbol{x}$. In order to apply stabilization of convection using SUPG we rewrite the previous equation as follows:

$$\dot{\boldsymbol{x}} + (\boldsymbol{u} - \boldsymbol{u}_m) \cdot \nabla_s \boldsymbol{x} = \boldsymbol{u}$$

2 Numerical Description

2.1 Spatial discretisation

Equations are discretized using the Finite Element Method employing a mesh of quadratic triangles (Figure 1). The interface mesh consists of quadratic lines while maintaining conforming geometry (Figure 1).

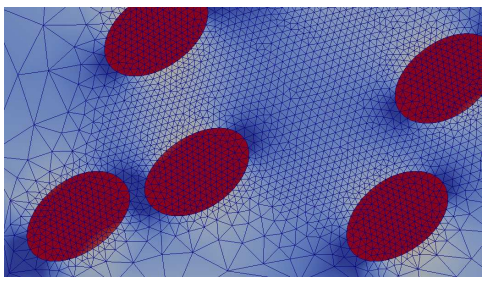
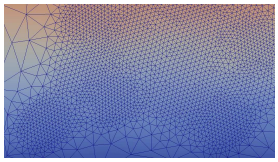
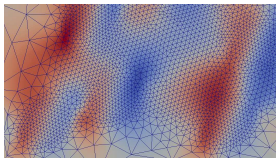


Figure 1: Part of the mesh retrieved from the Rheology tool with the colors represting pressure.

We use quadratic interpolation for the velocity and linear interpolation for the pressure. Since the



(a) Velocity in x-direction



(b) Velocity in y-direction

Figure 2: Part of the mesh retrieved from the Rheology tool with the colors representing different velocity components.

droplet mesh is decoupled from the matrix mesh, all nodes at the interface are double. The velocities in the interface nodes are coupled by using collocated Lagrange multipliers, leading to a continuous velocity field across the interface. For the time discretization a second order Backwards Differencing formula is used.

As described in Section 1, here we consider the Stokes equation in a region Ω:

$$-\nabla \cdot (2\eta \boldsymbol{D}) + \nabla p = \boldsymbol{f}, \quad \text{in } \Omega \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \quad \text{in } \Omega \tag{2}$$

where $\boldsymbol{D} = (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)/2$. The boundary conditions are

$$\boldsymbol{u} = \boldsymbol{u}_D \text{ on } \Gamma_D \tag{3}$$

$$-p\boldsymbol{n} + 2\eta \boldsymbol{D} \cdot \boldsymbol{n} = \boldsymbol{t}_\mathrm{N} \text{ on } \Gamma_\mathrm{N} \tag{4}$$

where the boundary $\Gamma = \Gamma_D \cup \Gamma_\mathrm{N}$ has been split into a Dirichlet and a Neumann part. The Neumann condition is equivalent to an imposed traction of $\boldsymbol{t}_\mathrm{N}$.

The weak form can be obtained by multiplying Eqs. (1)–(2) with test functions $\boldsymbol{v}$ and q:

$$(\boldsymbol{v}, -\nabla \cdot \boldsymbol{\sigma} - \boldsymbol{f}) = 0, \quad \text{for all } \boldsymbol{v}$$

$$(q, \nabla \cdot \boldsymbol{u}) = 0, \quad \text{for all } q$$

where

$$(\boldsymbol{a}, \boldsymbol{b}) = \int_\Omega \boldsymbol{a} \cdot \boldsymbol{b}\, dx$$

$$(a, b) = \int_\Omega ab\, dx$$

and $\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\eta \boldsymbol{D}$.

Using partial integration and Gauss' theorem we obtain the following weak form: Find $\boldsymbol{u}$ and p such that

$$\big(\boldsymbol{D}_v, 2\eta \boldsymbol{D}\big) - (\nabla \cdot \boldsymbol{v}, p) = (\boldsymbol{v}, \boldsymbol{t}_\mathrm{N})_{\Gamma_\mathrm{N}} + (\boldsymbol{v}, \boldsymbol{f}) \quad \text{for all } \boldsymbol{v} \tag{5}$$

$$(q, \nabla \cdot \boldsymbol{u}) = 0, \quad \text{for all } q \tag{6}$$

using appropriate spaces for $\boldsymbol{u}$, p, $\boldsymbol{v}$ and q. In this equation $\boldsymbol{D}_v$ is defined by

$$\boldsymbol{D}_v = (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T)/2$$

and the bilinear operator $(\,,\,)$ for tensors is given by

$$(\boldsymbol{A}, \boldsymbol{B}) = \int_\Omega \boldsymbol{A} : \boldsymbol{B}\, dx$$

The weak form can be used to obtain an approximate solution using the Galerkin FEM technique. For this we divide the region into elements:

$$\Omega = \cup_e \Omega_e, \quad \Omega_e \cap \Omega_f = \emptyset \quad \text{for } e \neq f$$

and interpolate $\boldsymbol{u}$ and p (and similarly $\boldsymbol{v}$ and q) by polynomials on each element:

$$\boldsymbol{u}_h = \textstyle\sum_k \phi_k(\boldsymbol{x}) \boldsymbol{u}_k = \underset{\sim}{\phi}^T(\boldsymbol{x}) \underset{\sim}{\boldsymbol{u}}$$

$$p_h = \textstyle\sum_k \psi_k(\boldsymbol{x}) p_k = \underset{\sim}{\psi}^T(\boldsymbol{x}) \underset{\sim}{p}$$

where $\underset{\sim}{\phi}(\boldsymbol{x})$ and $\underset{\sim}{\psi}(\boldsymbol{x})$ are global shape functions. Substituting into the weak form leads to

$$\underset{\sim}{\boldsymbol{v}}^T \cdot (\underline{\boldsymbol{S}} \cdot \underset{\sim}{\boldsymbol{u}} - \underline{\boldsymbol{L}}^T \underset{\sim}{p}) = \underset{\sim}{\boldsymbol{v}}^T \cdot \underset{\sim}{\boldsymbol{f}} \quad \text{for all } \underset{\sim}{\boldsymbol{v}}$$

$$\underset{\sim}{q}^T \underline{\boldsymbol{L}} \cdot \underset{\sim}{\boldsymbol{u}} = 0 \quad \text{for all } \underset{\sim}{q}$$

or

$$\begin{bmatrix} \underline{\boldsymbol{S}} & -\underline{\boldsymbol{L}}^T \\ \underline{\boldsymbol{L}} & \underline{0} \end{bmatrix} \begin{bmatrix} \underset{\sim}{\boldsymbol{u}} \\ \underset{\sim}{p} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{\boldsymbol{f}} \\ \underline{0} \end{bmatrix}$$

with

$$\underline{\boldsymbol{S}} = \int_\Omega \eta[(\nabla \underset{\sim}{\phi} \cdot \nabla \underset{\sim}{\phi}^T)\boldsymbol{I} + (\nabla \underset{\sim}{\phi} \nabla \underset{\sim}{\phi}^T)^c]\, dx$$

$$\underline{\boldsymbol{L}} = (\underset{\sim}{\psi}, \nabla \underset{\sim}{\phi}^T)$$

$$\underset{\sim}{\boldsymbol{f}} = (\underset{\sim}{\phi}, \boldsymbol{t}_N)_{\Gamma_\mathrm{N}} + (\underset{\sim}{\phi}, \boldsymbol{f})$$

or in index notation:

$$S_{ij}^{km} = \textstyle\sum_l (\phi_{k,l}, \eta \phi_{m,l}) \delta_{ij} + (\phi_{m,i}, \eta \phi_{k,j})$$

$$L_i^{km} = (\psi_k, \phi_{m,i})$$

$$f_i^k = (\phi_k, t_{iN})_{\Gamma_\mathrm{N}} + (\phi_k, f_i)$$

Here k and m denote nodal points and i and j component directions of vectors. In practice we multiply the continuity equation by -1 in order to obtain a symmetric system matrix:

$$\begin{bmatrix} \underline{\boldsymbol{S}} & -\underline{\boldsymbol{L}}^T \\ -\underline{\boldsymbol{L}} & \underline{0} \end{bmatrix} \begin{bmatrix} \underset{\sim}{\boldsymbol{u}} \\ \underset{\sim}{p} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{\boldsymbol{f}} \\ \underset{\sim}{0} \end{bmatrix}$$

2.2 Interface tracking

The boundary of a single phase or the sharp interface between two phases can be described *explicitly* (for example by discrete particles or by a surface mesh) or *implicitly* (for example by a levelset function). Following the motion of the interface is called *interface tracking* for an explicit interface and *interface capturing* for an implicit interface. In case of an explicit interface described by a surface mesh, we might as well use a FEM-based approach, see M.M. Villone and Maffettone. (2014).

An interface in 3D (a surface) can be described by a moving curvilinear coordinate system given by

$$\boldsymbol{x} = \bar{\boldsymbol{x}}(\underset{\sim}{\xi}, t) \tag{7}$$

where $\underset{\sim}{\xi} = (\xi^1, \xi^2)$ are the curvilinear coordinates and $\bar{\boldsymbol{x}}$ is the function that maps the coordinates $\underset{\sim}{\xi}$ onto the spatial coordinates $\boldsymbol{x}$. Note, that the mapping is a function of time (moving coordinates). In the following we will call $\underset{\sim}{\xi}$ the grid coordinates, or simply the grid. Note, that in 2D we will have curves instead of surfaces and are described by a single curvilinear coordinate ξ.

We are only considering material interfaces, i.e. the velocity of the interface $\dot{\boldsymbol{x}}$ must be such that:

$$\dot{\boldsymbol{x}} \cdot \boldsymbol{n} = \boldsymbol{u} \cdot \boldsymbol{n} \tag{8}$$

where $\boldsymbol{u}$ is the material velocity at the interface and $\boldsymbol{n}$ is the normal vector of the surface. We have used the following notation here:

$$\dot{\boldsymbol{x}} = \frac{\partial \bar{\boldsymbol{x}}}{\partial t}|_{\underset{\sim}{\xi} \text{ is constant}}$$

Note, that $\dot{\boldsymbol{x}}$ is equal the velocity of the material in the normal direction only. In the tangential direction the grid can move arbitrarily, similar to the ALE formulation.

The following equation for the motion of an interface has been implemented:

$$\dot{\boldsymbol{x}} = (\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot \boldsymbol{n}\boldsymbol{n} + \boldsymbol{u}_\mathrm{m} + \boldsymbol{u}_\mathrm{t} \equiv \boldsymbol{g}(\boldsymbol{x}, t) \tag{9}$$

where $\boldsymbol{u}_\mathrm{m}$ is an arbitrary vector field and $\boldsymbol{u}_\mathrm{t}$ is a vector field tangential to the interface and proportional to the surface gradient of a scalar field defined on the interface:

$$\boldsymbol{u}_\mathrm{t} = k \nabla_\mathrm{s} c \tag{10}$$

with k a factor specified by the user. A routine is available for solving the Poisson problem:

$$-\nabla_\mathrm{s}^2 c = \frac{1}{f} - \frac{1}{g} \tag{11}$$

where f is a monitor function representing the requested area of the elements and g is the element area function. The right-hand side has been scaled such that $\int_\Gamma 1/f \, dS = \int_\Gamma 1/g \, dS =$ area of Γ. The function f needs to be specified by the user. Setting $f = 1$ will lead to approximately equal sized elements on the surface. Notes:

1. The tangential motion of the interface is given by
$$(\boldsymbol{I} - \boldsymbol{n}\boldsymbol{n}) \cdot \boldsymbol{u}_\mathrm{m} + \boldsymbol{u}_\mathrm{t}$$
which contributes only to the redistribution of the nodes of the mesh.

2. The main purpose for introducing the field $\boldsymbol{u}_\mathrm{m}$ is to make the movement of the nodes on the interface Galilean frame independent. For example, by taking $\boldsymbol{u}_\mathrm{m}$ equal to the velocity of the mass center of a droplet (and setting $\boldsymbol{u}_\mathrm{t} = \boldsymbol{0}$, for simplicity), the motion of the nodes on the droplet surface are identical to the motion of the nodes as when solving
$$\dot{\boldsymbol{x}} = (\boldsymbol{u} \cdot \boldsymbol{n})\boldsymbol{n} \tag{12}$$
in a frame relative to the motion of the mass center.

3. The form of the tangential interface velocity $\boldsymbol{u}_\mathrm{t}$ has been inspired by the grid deformation technique .

4. If $\boldsymbol{u}_\mathrm{m} = \boldsymbol{u}$ the interface movement is Lagrange based:

$$\dot{\boldsymbol{x}} = \boldsymbol{u} + \boldsymbol{u}_\mathrm{t} \tag{13}$$

If in addition $\boldsymbol{u}_\mathrm{t} = \boldsymbol{0}$, the interface moves in a Lagrangian way. It is up to the user which tracking scheme (normal velocity or Lagrange based) is preferred. Notes:

- The normal velocity based scheme gets rid of the tank-treading motion whereas the Lagrange based does not.
- The normal velocity based scheme needs to be stabilized with SUPG whereas the Lagrange based does not.
- The normal velocity based scheme cannot handle non-smooth interfaces very well whereas the Lagrangian scheme can.

2.3 A semi-implicit time integration scheme with SUPG stabilization

In order to apply stabilization of convection using SUPG we need to identify the convection term in Eq. (9). For this, we rewrite the equation as follows:

$$\dot{\boldsymbol{x}} + (\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot (\boldsymbol{I} - \boldsymbol{n}\boldsymbol{n}) = \boldsymbol{u} + \boldsymbol{u}_\mathrm{t}$$

Substituting $\boldsymbol{I} - \boldsymbol{n}\boldsymbol{n} = \boldsymbol{g}^i \boldsymbol{g}_i$ (surface identity tensor), where $\boldsymbol{g}_i = \partial \boldsymbol{x}/\partial \xi^i$, $i = 1, 2$ are the (covariant) base vectors and $\boldsymbol{g}^i$, $i = 1, 2$ the dual base vectors, we get

$$\dot{\boldsymbol{x}} + (\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot \boldsymbol{g}^i \frac{\partial \boldsymbol{x}}{\partial \xi^i} = \boldsymbol{u} + \boldsymbol{u}_\mathrm{t}$$

or

$$\dot{\boldsymbol{x}} + (\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot \nabla_\mathrm{s} \boldsymbol{x} = \boldsymbol{u} + \boldsymbol{u}_\mathrm{t} \tag{14}$$

with $\nabla_\mathrm{s} = \boldsymbol{g}_i \dfrac{\partial}{\partial \xi^i}$ the surface gradient operator. This is a non-linear unsteady convection equation on the surface for the position $\boldsymbol{x}$ of the surface.

Similarly to the height function equation we apply SUPG to stabilize the convection and use implicit time integration. A complication here is that the surface gradient operator ∇_s depends on the position of the surface $\boldsymbol{x}$. Therefore, we use a prediction of $\boldsymbol{x}$ for determining ∇_s. The first and second-order schemes based on backwards differencing (Gear) now become:

first order

$$\begin{aligned}&\hat{\boldsymbol{x}}_{n+1} = \boldsymbol{x}_n \\ &\Big(\boldsymbol{w} + \tau(\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot \hat{\nabla}_\mathrm{s} \boldsymbol{w}, \frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{\Delta t} + \\ &\quad (\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot \hat{\nabla}_\mathrm{s} \boldsymbol{x}_{n+1} - \boldsymbol{u} - \boldsymbol{u}_\mathrm{t}\Big) = 0\end{aligned}$$

second order

$$\begin{aligned}&\hat{\boldsymbol{x}}_{n+1} = 2\boldsymbol{x}_n - \boldsymbol{x}_{n-1} \\ &\Big(\boldsymbol{w} + \tau(\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot \hat{\nabla}_\mathrm{s} \boldsymbol{w}, \frac{\frac{3}{2}\boldsymbol{x}_{n+1} - 2\boldsymbol{x}_n + \frac{1}{2}\boldsymbol{x}_{n-1}}{\Delta t} + \\ &\quad (\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot \hat{\nabla}_\mathrm{s} \boldsymbol{x}_{n+1} - \boldsymbol{u} - \boldsymbol{u}_\mathrm{t}\Big) = 0\end{aligned}$$

where $\boldsymbol{w}$ is the test function of $\boldsymbol{x}$, τ is the SUPG parameter, $\hat{\nabla}_\mathrm{s}$ is evaluated using $\hat{\boldsymbol{x}}_{n+1}$ and $\boldsymbol{u}$, $\boldsymbol{u}_\mathrm{m}$, $\boldsymbol{u}_\mathrm{t}$ need to be evaluated at time t_{n+1}.

Remarks:

1. For the SUPG parameter we take

$$\tau = \frac{\beta h}{2U_\mathrm{t}}$$

where h is a characteristic element length, β is a scalar between 0 and 1 and $U_\mathrm{t} = |(\boldsymbol{u} - \boldsymbol{u}_\mathrm{m}) \cdot (\boldsymbol{I} - \boldsymbol{n}\boldsymbol{n})|$, the length of the tangential velocity. Note, that U_t is evaluated in each integration point separately. In the current code we use $h = \sqrt{A_\mathrm{e}}$ for a surface and adjust β to get the best results, usually around 1.0 for linear elements and 0.5 for quadratic elements.

2. The time-discretization is semi-implicit and based on the Gear schemes. The second-order scheme is a two-step scheme which needs to be started by the first-order scheme in the first time step.

3. The system matrix is non-symmetric.

4. The semi-implicit scheme is more robust than an explicit scheme and, based on stability, time steps can be taken roughly ten times larger than for the explicit scheme.

5. Another variant is obtained by rewriting Eq. (14) to

$$\dot{\boldsymbol{x}} + (\boldsymbol{u} - \boldsymbol{u}_{\mathrm{m}} - \boldsymbol{u}_{\mathrm{t}}) \cdot \nabla_{\mathrm{s}} \boldsymbol{x} = \boldsymbol{u} \tag{15}$$

since $\boldsymbol{u}_{\mathrm{t}}$ is a vector in tangential direction only. However, initial results were worse with this variant and we did not further explore this one.

2.4 Mesh adaptivity

Due to the flow the mesh elements can get really deformed. For that reason we have to measure this deformation so that after a certain threshold we can remesh. The mesh deformation is measured by

$$f_1^e = |\log(A^e/A_0^e)| \tag{16}$$

and

$$f_2^e = |\log(S^e/S_0^e)| \tag{17}$$

where A^e and S^e are the area and the aspect ratio of an element e, respectively. The subindex 0 means the initial value. The aspect ratio of an element S^e is defined by

$$S^e = (L_{\max}^e)^2/A^e \tag{18}$$

where $L_{\max}^e$ is the maximum length of the sides. Remeshing is performed if either of

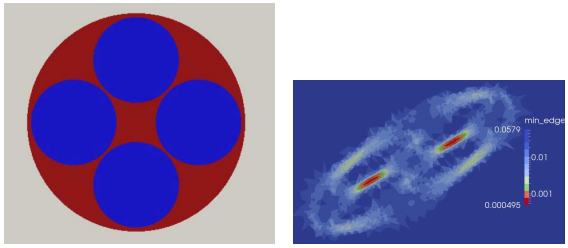
$$f_1 = \max_e f_1^e \tag{19}$$

or

$$f_2 = \max_e f_2^e \tag{20}$$

exceed a certain threshold.

To simulate accurately the thin film layers of fluid between the bubbles we implemented a multilevel adaptive local refinement scheme. The interface mesh is refined by splitting the elements into two new. The rest of the volume is refined by locally defining element size in Gmsh. For each element level the parent and the child elements are stored. In this way it is possible to move back and forth through element levels easily. The model is able to deal with a large range of scales that can reach up to five orders of magnitude difference (Figure 4). In Figure (3) we can see a test case where four initial circular drops inside a bigger one, are deformed due to the flow. In this example the maximum element level reached is 9 which means the final element size is approximately 500 times smaller than the starting size.



(a) (b)

Figure 3: a) Initial configuration of circular drops, all of which consist of Newtonian fluid. b) Plot of the minimum edge size of each element in a scale after applying shear flow.

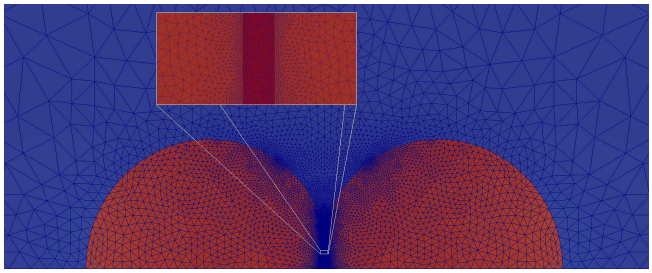


Figure 4: The difference of the element size can get up to five orders of magnitude.

3 Coupled scheme approach

In the previous version of the model we were using a decoupled scheme for the interface tracking and Stokes flow equations. In a single step we were first solving the flow problem and then in a second semi step the interface tracking equation was

solved by using the velocities $\boldsymbol{u}$ on the interface as knowns. The timestep in this approach was limited by the Capillary time which is $\Delta t_{min} = \frac{\Delta x \eta}{\Gamma}$ where Δx is the element size, η is the viscosity of the matrix and Γ the surface tension. To overcome this issue we derived a coupled scheme where the velocities $\boldsymbol{u}$ are coupled with position of the interface $\boldsymbol{x}$ creating a new system of equations

$$\begin{aligned} -\boldsymbol{\nabla} \cdot (2\eta \boldsymbol{D}) + \boldsymbol{\nabla} p &= \boldsymbol{f} - \gamma \kappa n_\Gamma, && \text{in } \Omega \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= 0, && \text{in } \Omega \\ \dot{\boldsymbol{x}} - (\boldsymbol{u} - \boldsymbol{u}_m) \cdot \boldsymbol{n}\boldsymbol{n} - \boldsymbol{u}_m &= 0, && \text{in } \Gamma \end{aligned}$$

where Ω is the matrix space and Γ the interface. The boundary conditions remain the same as in the decoupled scheme. This is a non linear system of equation and to solve that we need an iterative method. Our system is not fully implicit since we do not take the change of area into account for the calculation of the volume terms in the Stokes equations and thus we are going to use a Picard like scheme to solve it.

We were able to verify that our timestep is not limited as in the decoupled scheme by running some benchmark simulations. We used a drop with surface tension $\gamma = 0.1$ in a matrix of viscosity $\eta = 1$ and a mesh with minimum element size on on the interface $\Delta x = 0.015$. In the case of the case of the decoupled scheme we are limited by $\Delta t = 0.15$ while with the decoupled scheme our only time limitation comes from the nature of the problem.

Figure 5: A drop relaxed without any flow the colors represent pressure.

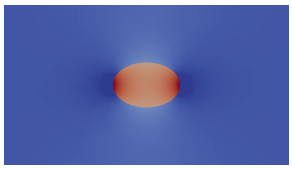


Figure 6: A stretched drop under surface tension forces without any flow.

For the relaxed drop, theoretically it is possible to use an infinite big timestep. We were able to use timestep up 10^3 without any problems. In the second example we are limited by the nature of the problem, it is obvious that one can not use timestep bigger or even close to the time needed for the relaxation.

References

M.M. Villone, M.A. Hulsen, P. A. and Maffettone., P. (2014). Simulations of deformable systems in fluids under shear flow using an arbitrary lagrangian eulerian technique. *Computers & Fluids*, 90:88 – 100.

A

MileStone 3.1 Developed software for solving rheology, coalescence and nucleation models

This milestone relates to the developed software for solving rheology, coalescence and nucleation models. The related tasks are 3.1 and 3.2. For task 3.1 the layout of the tools has been define, this layout contains a review of the existing software that will be used for the macroscale computations. The task has been completed and more details can be found in Deliverable 3.1. For the second task all the necessary connections have been completed. The implemented interfaces and adaptors are described in detail in Deliverable 3.1. The development of the tools is still work in progress (Task 3.4). For more details about the developed tools we refer to Deliverable 3.3.

B

MileStone 3.3 Initial validation of macro-scale modelling predictions

This milestone relates to the initial validation of macro-scale tool. It starts with Task 3.1 which presents the layout of this tool. Deliverable 3.1 describes the details of this task and explains the communication methods for the software development. This task has been accomplished and a model prototype has been validated using the available literature data for 12 polyurethane recipes. The prototype is already available through the official MoDeNa github page.

The other task referring to the macro-scale model is Task 3.4. The major activity throughout this task involves the development of a macro-scale model for fluid dynamics of the foam. This has been achieved by incorporating a population balance equation with a computational fluid dynamics code. The result has been appeared as a new CFD-PBE solver in the open-source structure of OpenFOAM CFD code. However, the details will be reported on Deliverable 3.4. The primary validations of the solver have been performed for a typical mixing-cup experiment. Current activities related to M3.3 focus on the validations of test cases, recently presented by BASF, and preparing the solver for public.