Adaptative sloshing simulation using model reduction and GENERIC structure

Flavien Alonzo

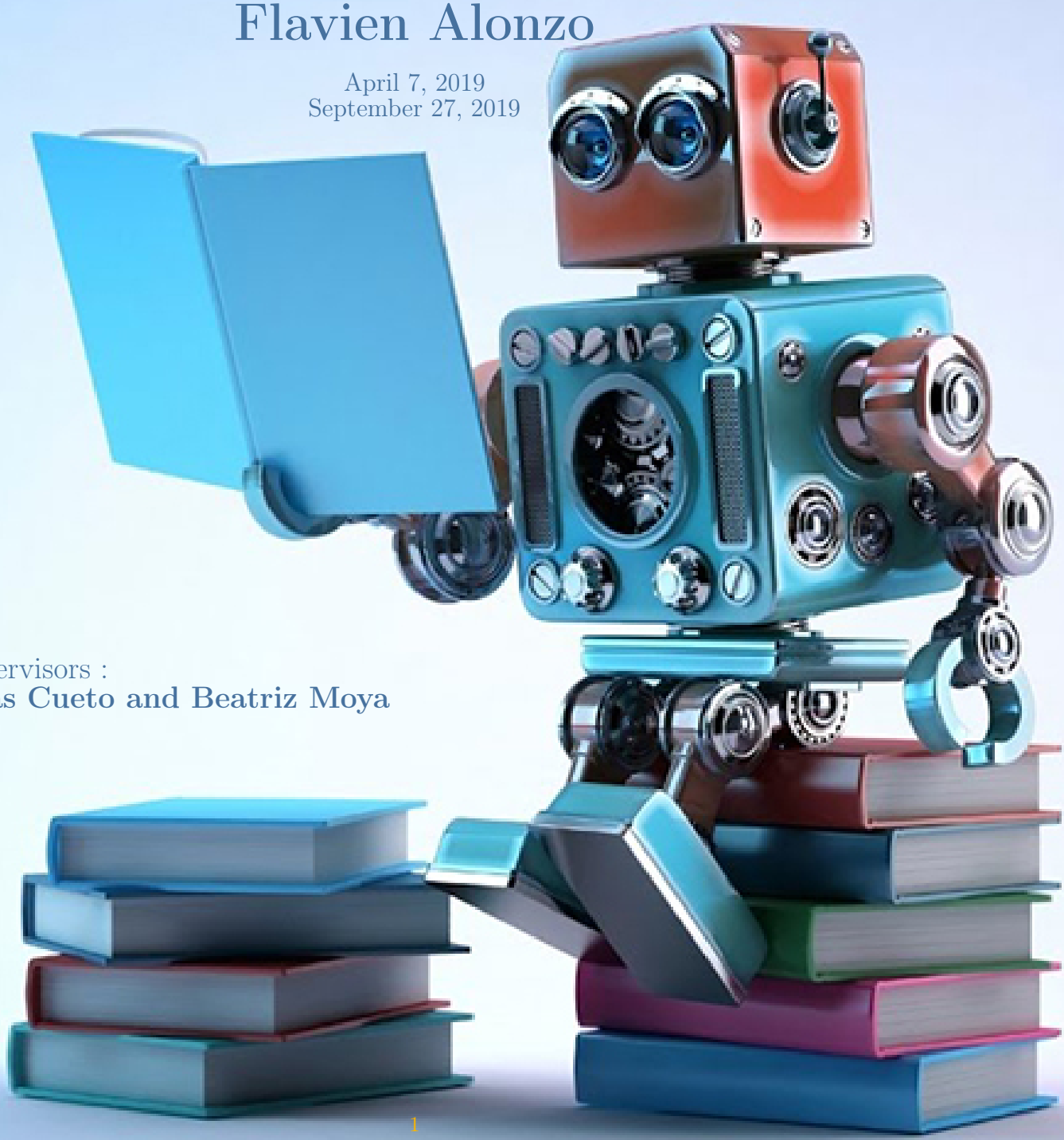
Instituto Universitario de Investigación
en Ingeniería de Aragón
Universidad Zaragoza

CENTRALE NANTES

Adaptative sloshing simulation using model reduction and GENERIC structure

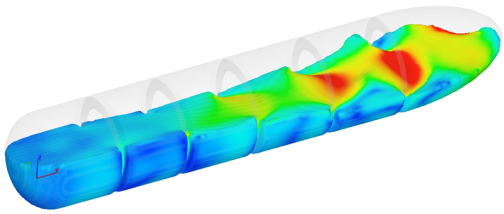
Flavien Alonzo

April 7, 2019
September 27, 2019

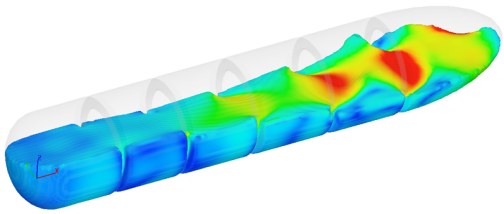


Supervisors :
Elias Cueto and Beatriz Moya

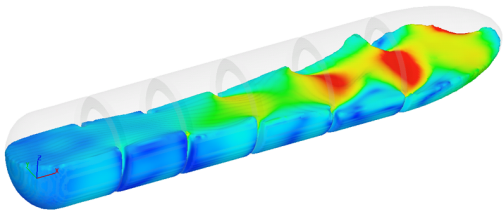
Contents



List of Figures



List of Tables



cmn2019
Congress on Numerical Methods in Engineering

DATA-DRIVEN LEARNING OF SLOSH DYNAMICS

Beatriz Moya, David González, Icíar Alfaro, Francisco Chinesta, Elías Cueto

MOTIVATION

- ***Goal:*** development of physically consistent integrator to learn fluid behavior from perception

METHOD

1. Obtain discretized data to construct the integrator
2. Find reduced order manifold from data
3. Build physically sound structure of the problem

TEST

- ✓ A new trajectory was simulated
- ✓ The height reconstruction error remained under 7%
- ✓ Simulation of 1.7 seconds was performed in 1.64 seconds

RESULTS

- Model built with **non-linear** methodologies
- **Real-time** achieved
- Accurate enough to develop computer vision applications

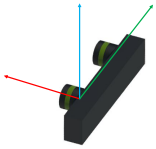
FUTURE RESEARCH LINES

- Widen data base to include Newtonian and Non Newtonian fluids
- Test the integrator with information from the scene

@BeatrizMoyaG
beam@unizar.es

APPLICATION

This technology is now under the spotlight of simulation-based control of robots.



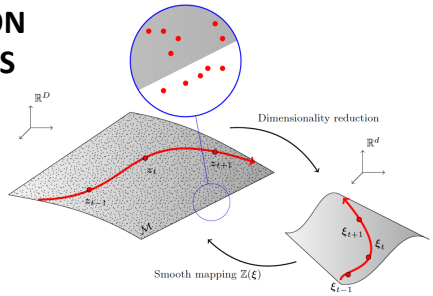
BACKGROUND

Models built from data usually employ deep leaning techniques, but they:

- ✗ Are not physically coherent
- ✗ Deviate from ground truth in long term simulations

MODEL ORDER REDUCTION OF NONLINEAR DYNAMICS

- Proper orthogonal decomposition
- Locally linear embedding
- Topological Data Analysis



THE GENERIC FORMALISM

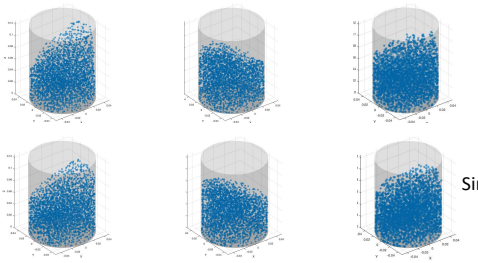
Generic provides a physically consistent structure of the dynamics to build the integrator from the evolution of energy and entropy:

$$\frac{dz}{dt} = L\frac{\partial E}{\partial z} + M\frac{\partial S}{\partial z}.$$

Degeneracy conditions ensure accomplishment of thermodynamics laws:

$$L\frac{\partial S}{\partial z} = 0, M\frac{\partial E}{\partial z} = 0.$$

TESTING RESULTS



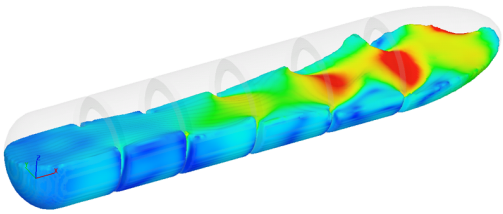
References

ROWEIS S. T. and SAUL L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323(2326, 2000.
WASSERMAN L., *Annual Review of Statistics and Its Application* 5(1), 501 (2018).
ÖTTINGER H.C., Beyond Equilibrium Thermodynamics (Wiley, 2005)
MOYA B., GONZALEZ D., ALFARO I., CHINESTA F., CUETO E. Learning Slosh Dynamics by means of data. *Comp. Mech.* (2019). P1:13

APMAC SEMNI
Universidad Zaragoza
Instituto Universitario de Investigación de Ingeniería de Aragón
Universidad Zaragoza
Unión Europea
Fondo Social Europeo
GOBIERNO DE ARAGON

1 Context of the internship

This work is the result of my third-year internship at the Instituto Universitario de Investigacion en Ingenieria de Aragon in the University of Zaragoza (Spain).
My work follows the one from a PhD student that works on the problematic to solve real-time sloshing problems, from the detection of initial data with a camera to its rendering. Her first paper [1] shows that a numerical scheme based on the GENERIC structure [2] is able to reproduce in real-time the sloshing behavior of water with data-driven fluid simulation. The data were obtained on Abaqus with the SPH method [3]. A poster resuming her work is available in the previous page.
My objectives were to enhance the simulations by adding more liquids, even non-Newtonian, and to change the method to simulate the data on Abaqus.
Finally, I have been able to simulate liquids behavior with the ALE-mesh [4] and to generalize the method to simulate non-Newtonian liquid behavior with it, to extract and convert the data obtained in those simulations to get the manifold of the data, to embed this manifold in a 2D space with a high fidelity of the structure of trajectory using the UMAP [5] method and to compute the GENERIC numerical scheme on those trajectories to reconstruct them from the initial time step with high accuracy compared with the initial data.



2 Introduction to Fluid mechanics

2.1 Characteristics of a fluid

In Fluid mechanics, there are a lot of parameters that can be physically defined and found in equations. A list of those that will be used in the rest of this report are introduced here:

- ρ: fluid density (kg/m^3)
- $\overrightarrow{v}$: velocity (m/s)
- t: time (s)
- p: pressure (N/m^2)
- μ: viscosity (Pa.s)
- e: internal energy (J)
- K: thermal conductivity (W.m^{-1}.K^{-1})
- θ: temperature (K)
- $\overline{\overline{\sigma}}$: Cauchy stress tensor
- $\overline{\overline{\epsilon}}$: Strain tensor
- $\overrightarrow{f}$: specific body force vector
- c: sound speed (m/s)

2.2 Common liquids

In this work, I am only interested in liquids that could be manipulated in a glass by a robot. Which means that I am interested in liquids such as water, melted chocolate, melted butter, blood, mayonnaise,...
In this purpose, I have been looking for the usual parameters of common liquids (such as density, viscosity,...), and have built a data-set composed of 113 liquids that could be found in a glass. This data-set is available in appendix A. Most values of density and viscosity come from [6] and [7], other sources coming from internet.

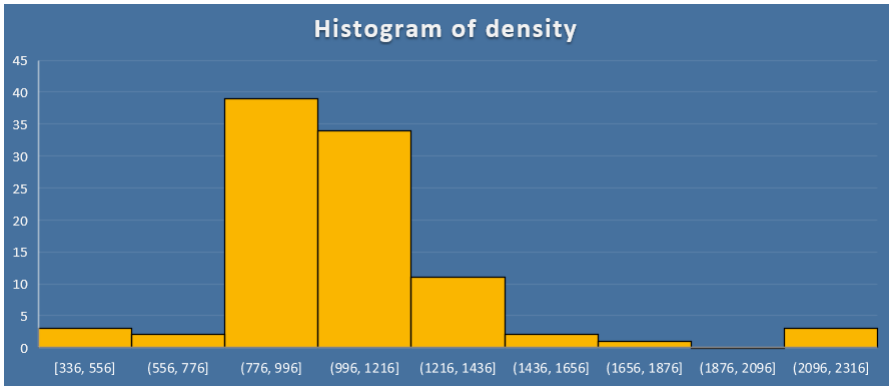
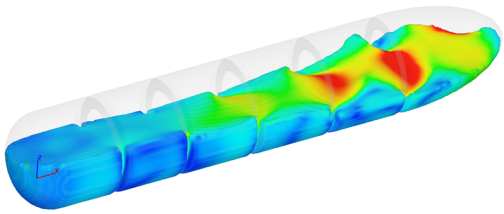


Figure 1: Density's histogram.

The most important values here, are the density and the viscosity of a liquid, I have then inspected the distribution of those two parameters for the liquids in the data-set. The distribution of the density is represented in the figure 1 and the viscosity's one in the figure 2. I have also studied the assumption that density and viscosity are independent parameters or not.



Mean	Standard deviation	Skewness	Kurtosis
-0.54	1.21	0.183	2.64

Table 1: *First momenta of the viscosity's logarithm distribution.*

To be graphically convinced of that independence, each liquids is a point with density and viscosity as coordinate. In the figure 3, there are those points represented and no correlation can be seen. Concretely the correlation coefficient $\rho_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$ is approximately -1.86%. The assumption of independent density and viscosity is taken for the rest of the report.

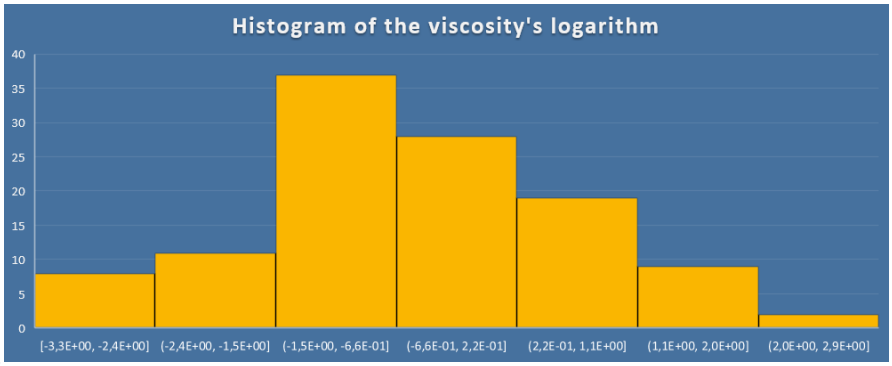


Figure 2: *Viscosity's histogram.*

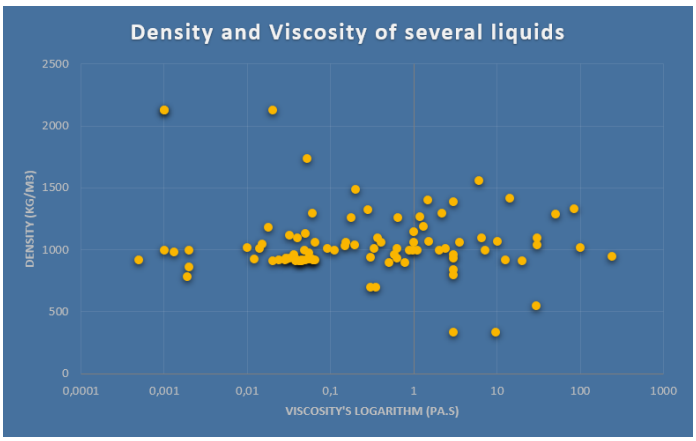
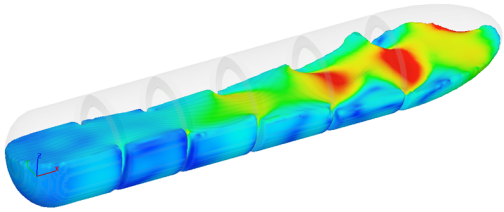


Figure 3: *Density and absolute viscosity of liquids in the data-set.* $\rho_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = -1.86\%$.

I have also tried to approximate the distribution of density and viscosity in case it would be necessary to simulate them.
For the viscosity, it has been observed in the figure 2 that its logarithm distribution was shaped like a Gaussian distribution. Table 1 gives it's first momenta, and, because the values of skewness and kurtosis are close to 0 and 3, the approximation may stand.
Let's $\mathscr{H}_0$: The viscosity's logarithm distribution follows $\mathcal{N}(-0.54, 1.21^2)$, be the assumption here.
Let's $F(X)$ be the distribution function and $F_n(X)$ its approximation.
The Kolmogrov-Smirnov test [8] tells that if

$$D = \max_x |F_n(x) - F(x)| < D_{n,\alpha}$$

then $F_n(X)$ is a good approximation of $F(X)$.



Here we get:

$$0.1053 = D < D_{198,0.02} = 0.1078$$

So Log-$\mathcal{N}(-0.54, 1.21^2)$ is a good approximation of the distribution of viscosity. A comparison of the boxplots of the viscosity's logarithm distribution and its model is showed in figure 4.

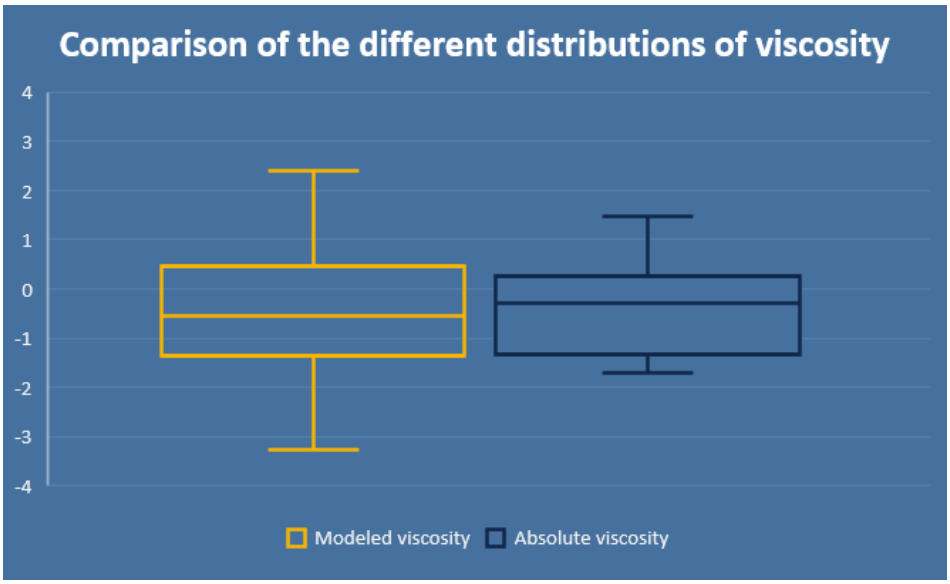
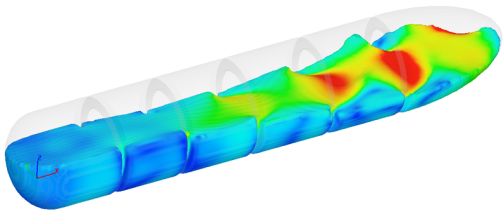


Figure 4: Boxplots of $\mathcal{N}(-0.54, 1.21^2)$ and the viscosity's logarithm distributions.

Modeling the density's distribution was a bit more complicated because the previous way to model was not accurate enough. But I managed to find a good approximation nonetheless:
Let's note $\Phi_{\mu,\sigma}$ the distribution function of $\mathcal{N}(\mu, \sigma^2)$.
The model given to F associated with the density distribution follows:

$$\begin{aligned} X \hookrightarrow F^{-1}(U) &= \Phi^{-1}_{\mu,\sigma}(4U(1-U))1_{\{U\leq 0.1\}} \\ &+ (A + (C-A).\frac{U-0.10}{0.70-0.10})1_{\{0.1<U<0.7\}} \\ &+ (C + 1300 - \Phi^{-1}_{\mu,\sigma}(4U(1-U)))1_{\{U\geq 0.7\}} \end{aligned}$$

With $\mu = 1000, \sigma = 284, A = \Phi^{-1}_{\mu,\sigma}(0.36), C = \Phi^{-1}_{\mu,\sigma}(0.84)$ and $U \sim \mathscr{U}(0,1)$.
A comparison between the density's distribution function and its model is given in figure 5.



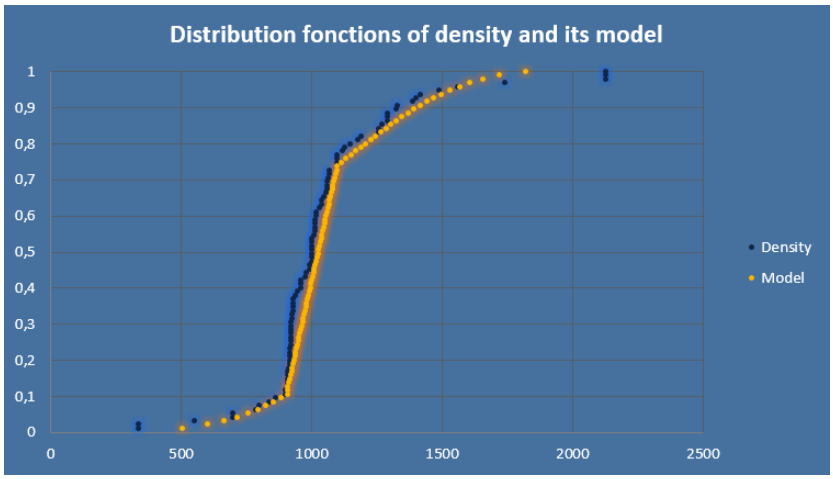


Figure 5: *Comparison between the distribution functions for the values of density and its model.*

Finally, if I simulate 1000 liquids with independent density and viscosity, I can obtain the following results showed in figure 6.

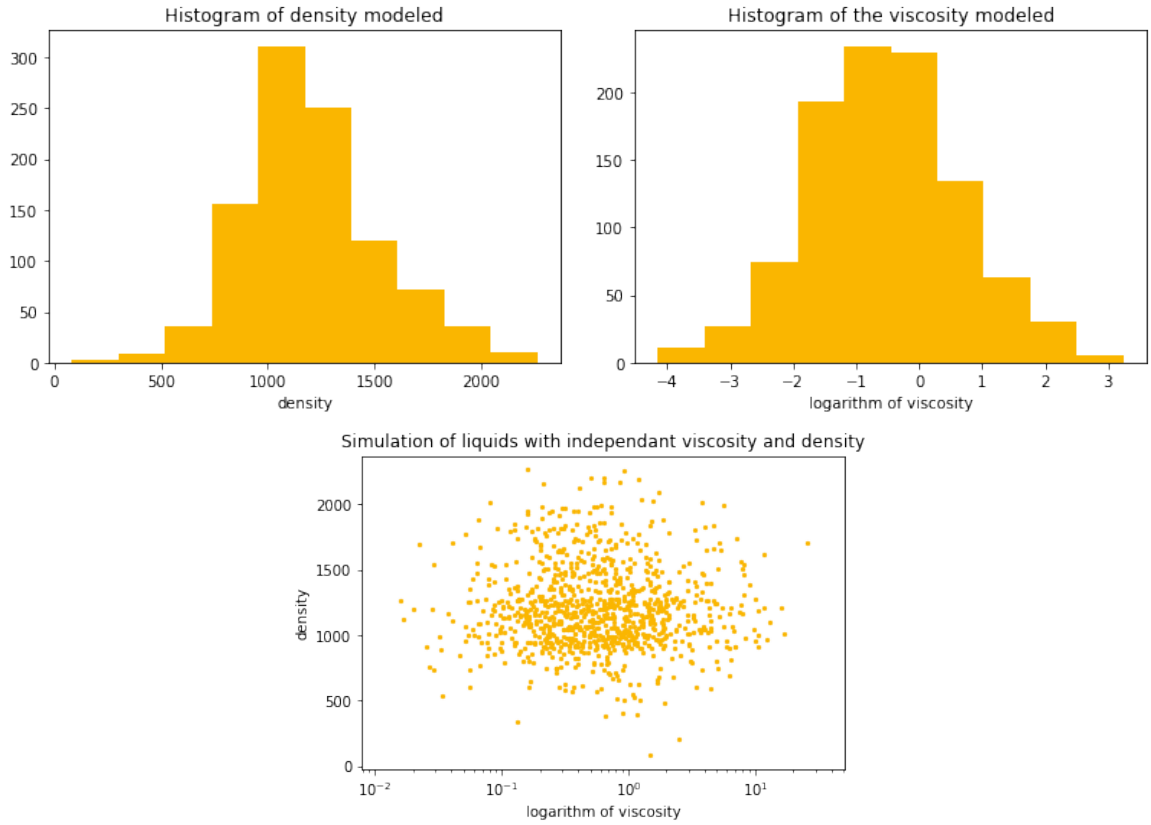
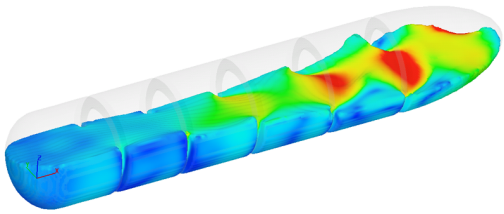


Figure 6: *Histogram of density of 1000 simulated values, histogram of the viscosity's logarithm of 1000 simulated values and simulation of 1000 liquids with independent viscosity and density.*



3 Abaqus

ABAQUS is a professional software for FEM (Finite Element Method) calculus. It has been developed by ABAQUS, Inc (Dassault Systèmes). In this work, I've used ABAQUS/CAE with the solver ABAQUS/Explicit which uses explicit schemes for integrating dynamic or quasi-static, non-linear problems.

3.1 Meshes

All FEM methods are based on the existence of a mesh with nodes where the variables remain to be actualised. ABAQUS can solve differently problems depending on the mesh chosen.

3.1.1 SPH: Smoothed Partial Hydrodynamics

This method comes from a work made in 1988 [3], it has the ambition to assimilate elements as particles, and so, to actualise the variables of each element by an interpolation of the closest points. The method works as follow:
The approximation of the value of f at a point $x \in \Omega$ is:

$$\langle f(x)\rangle = \int_\Omega f(r)w(x-r,h)dr = \int_\Omega \frac{f(r)}{\rho(r)}w(x-r,h)\rho(r)dr \tag{1}$$

where w is a kernel function, that follows two rules:

- $\int_\Omega w(x-r,h)dr = 1, \forall h$
- $\lim_{h\to 0} w(x-r,h) = \delta(x-r)$

Usually on ABAQUS, the kernel function is a cubic spline kernel. Its expression in 3D is:

$$w(q,h) = \frac{1}{\pi h^3}(1-\frac{3}{2}q^2(1-q))1_{\{0\le q\le 1\}} + \frac{1}{4\pi h^3}(2-q)^3 1_{\{1\le q\le 2\}} \tag{2}$$

Where $q = |\frac{x-r}{h}|$. SPH is a particle analysis meaning that we are looking at particles that all have characteristics such as a density ρ and a mass m.
By subdividing Ω in N elementary volumes, it is possible to get a discrete representation of the previous interpolated function, for example with:

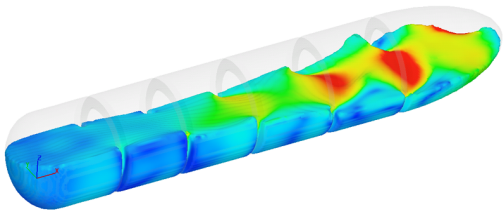
$$\langle f(x)\rangle_S = \sum_{k=1}^{N} \frac{f(r_k)}{\rho_k} w(x-r_k,h)m_k \tag{3}$$

(3) is the Riemann sum associated with (1), in which case the accuracy of the numerical approximation is:

$$|\langle f(x)\rangle - \langle f(x)\rangle_S| \le \frac{||\nabla g||_{L^\infty(\Omega)}|\Omega|^2}{2N} \tag{4}$$

Where $g(r) = f(r)w(x-r,h)$.
By doing an Taylor series of (1) and using the properties of the cubic kernel 2, the accuracy of the kernel approximation is:

$$|f(x) - \langle f(x)\rangle| = \frac{1}{2}H(f)(x)\int_\Omega (x-r)^2 w(x-r,h)dr + \Theta(h^3) = \Theta(h^2) \tag{5}$$



Combining (4) and (5), the accuracy of the SPH is in $\Theta(\frac{1}{N})+\theta(h^2)$, or in $\Theta(\frac{1}{N^2})+\theta(h^2)$ with a better numerical approximation than (3):

$$|f(x)-\langle f(x)\rangle_S| \leq |f(x)-\langle f(x)\rangle| + |\langle f(x)\rangle - \langle f(x)\rangle_S| = \Theta(\frac{1}{N}) + \Theta(h^2) \quad (6)$$

Usually, if h is chosen constant for every particles, its value is $h = \frac{1}{\bar{\rho}^{\frac{1}{3}}}$ in 3D. For liquids, $\bar{\rho} \simeq 1000 kg.m^{-3}$ so usually $h \simeq 0.1$. And for example, the number of particles N in [1] is 2898.

3.1.2 ALE: Arbitrary Lagrangian-Eulerian

ALE method uses both Lagrangian and Eulerian descriptions in order to solve problems without assimilating elements as particles like in the SPH. ALE method is described in [4].
Lagrangian description is good for following the shape of a material because it follows the physical points but that implies that there are possible errors coming if the mesh is too distorted around a position (which is easy to happen with liquids). Eulerian description is good against distortion because the mesh doesn't move through time but it simplifies too much the shape of the material (for example the shape of the free surface for a liquid).
The trick is then to use another configuration for actualising every variables. In figures 7 and 8 are showed the different configurations and the notations used to switch from one to another.
Following the notations of the figure 7, the fact that $\varphi = \phi \circ \psi^{-1}$ applied to Navier-Stokes equations, enables to get the Navier-Stokes equations in χ:

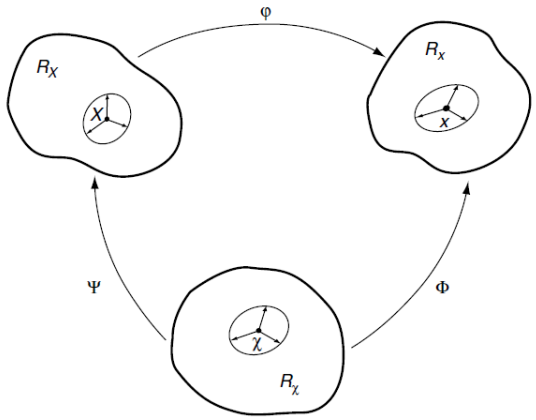
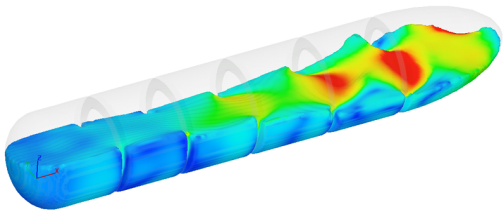


Figure 7: R_X material configuration (Lagrangian), R_x spatial configuration (Eulerian), R_χ reference configuration (ALE).

$$\begin{cases} \frac{\partial \rho}{\partial t}|_\chi + \tilde{c}.\nabla\rho & = -\rho\nabla.\vec{v} \\ \rho(\frac{\partial \vec{v}}{\partial}|_\chi + (\tilde{c}.\nabla)\vec{v}) & = \overline{\nabla}.\overline{\overline{\sigma}} + \rho\bar{b} \\ \rho(\frac{\partial E}{\partial t}|_\chi + \tilde{c}.\nabla E) & = \nabla.(\overline{\overline{\sigma}}.\vec{v}) + \vec{v}.\rho\bar{b} \\ \rho(\frac{\partial e}{\partial t}|_\chi + \tilde{c}.\nabla e) & = \overline{\overline{\sigma}} : (\frac{1}{2}(\nabla\vec{v} + \nabla\vec{v}^T)) \end{cases}$$

With $\tilde{c}$, the convective velocity. It is then possible to calculate the different variables of the nodes in χ.



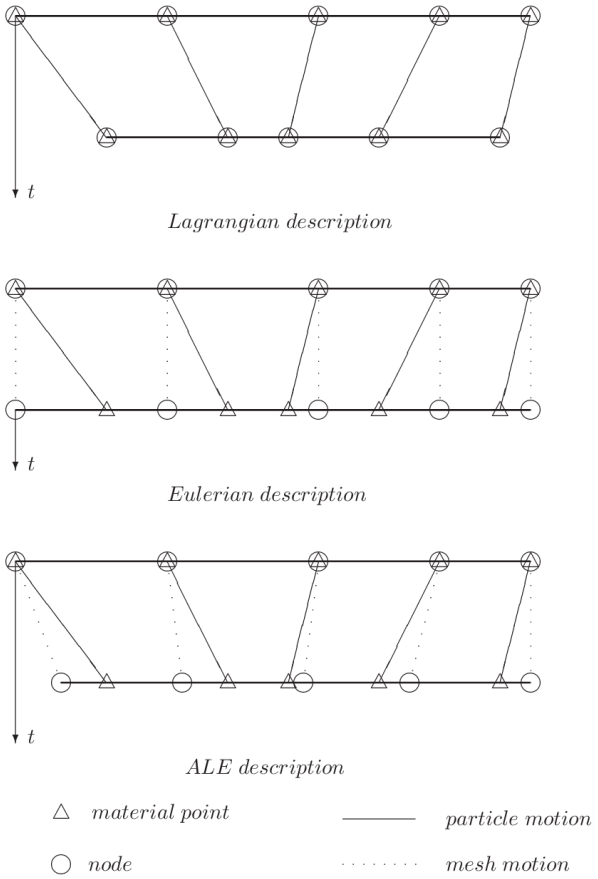


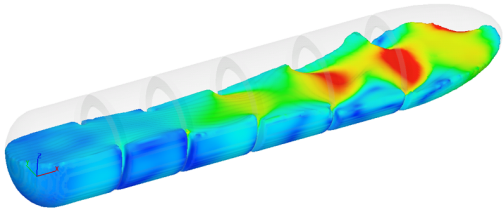
Figure 8: Representation of ALE mesh on the right as a mix between Lagrangian description on the left and Eulerian description in the middle.

3.2 Building model

Some studies ([9] or [10]) showed that the ALE mesh would give more detailed and more accurate results than with the SPH method. This is why I have had to simulate sloshing problems using the ALE mesh.
The next sections will show how it is possible to implement a sloshing problem using an ALE mesh with ABAQUS/Explicit.

3.2.1 Edit geometry

Following the work of [1], I designed the geometry of the liquid part as shown in figure 9.
To sketch that in ABAQUS, I have defined the liquid as a revolved solid. The section sketch is then a rectangle of height 7 cm and width 5 cm and the revolution along the vertical axis.
Even if physically there is a glass to play the role of the container, it can be not physically present in ABAQUS.
Indeed I can simulate the container by using the boundary conditions. Creating the glass physically create a contact between the liquid and the glass but the contact showed to be wrong during simulation which is why the glass is implemented through boundary conditions.



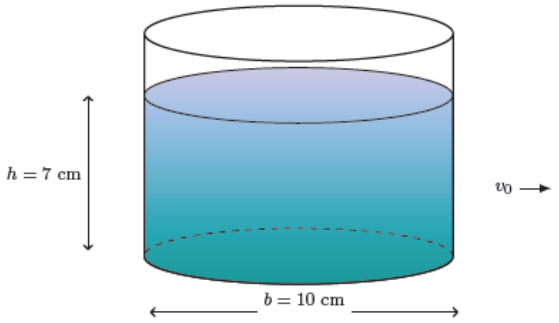


Figure 9: *Geometry part that has to be implemented in ABAQUS.*

In the figure 10, water is represented in green with a physical glass material and in blue with boundary conditions instead with the same other properties. It is possible to see that with the glass, there is a bad shape of water on the free surface, close to the side surface. However, that problem does not show when the glass is implemented with the use of boundary conditions. For these reason, I implemented the container with the use of boundary conditions.

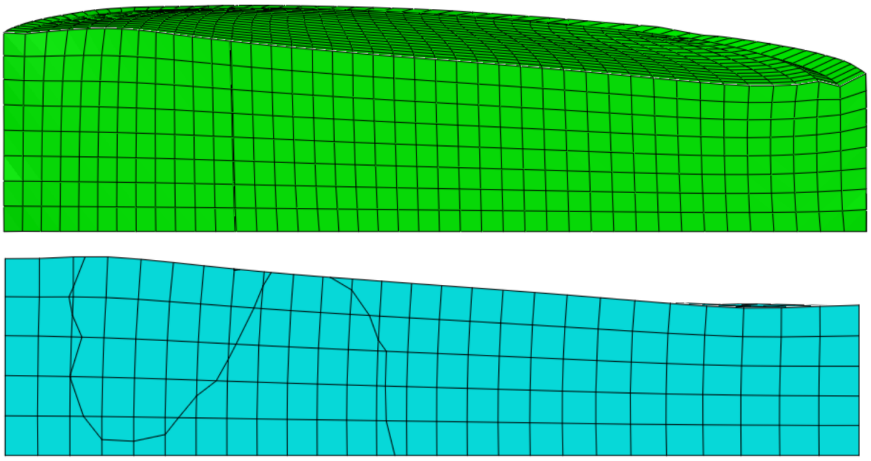


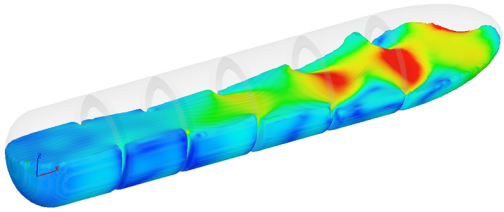
Figure 10: *Screenshots of the meshes on the xy plan at $t = 0.1s$ for water facing a velocity of $v = 0.05m/s$. For the green mesh, a contact between a rigid body and the water is defined. For the blue mesh, I include the container the borders with boundary conditions.*

3.2.2 Edit material

The liquid's material is made using 3 toolboxes: density, EOS (Equation Of State) and viscosity.
The density is temperature independent and set as uniformly distributed through the material.
EOS determine the pressure (positive in compression) as a function of the density and the specific energy: $p = f(\rho, e)$. It is used in the equation of conservation of energy :

$$\rho \frac{\partial e}{\partial t} = (p - p_{bv}) \frac{1}{\rho} \frac{\partial \rho}{\partial t} + \overline{\overline{S}} : \overline{\overline{\dot{e}}} + \rho \dot{q} \tag{7}$$

where $p_{bv} = \nabla.U \times K$ is the pressure induced by the bulk viscosity (K), $\overline{\overline{S}}$ is the deviatoric part of σ, $\overline{\overline{\dot{e}}}$ the one from ϵ and q is the heat rate.



Liquids	Density ($kg.m^{-3}$)	Viscosity ($Pa.s$)	Newtonian?
Blood [12]	1060	0.0035	No
Butter [7]	911	0.042	Yes
Chocolate [7]	1325	0.280	No
Honey [13]	1420	14	Yes
Mayonnaise [7]	910	20	No
Water [14]	983	0.0013	Yes

Table 2: Density and viscosity for blood, melted butter, melted chocolate, honey, mayonnaise and water when they are considered as Newtonian fluid. For each liquid, there is the information if the Newtonian assumption is accurate or not.

In this work, I have used the Mie-Grüneisen equations of state [11] which sets the pressure as a function of three parameters: $p = f(c, \Gamma, s)$, which is available in ABAQUS in the EOS toolbox with the type USUP. I have decided to use the same parameters for every liquids: $(c, \Gamma, s) = (1500m/s, 0, 0)$.
In Abaqus/Explicit the viscosity toolbox can only define viscosity for Newtonian fluids, meaning that the viscosity η through the material is constant and is used as follow:

$$\overline{\overline{S}} = 2\eta.\overline{\overline{\dot{e}}} \tag{8}$$

In table 2 there are the density and viscosity of some liquids that I have simulated on ABAQUS. The viscosity parameter correspond to the value of η in (8) even tough some of this liquids are not well approximated with a Newtonian hypothesis.

3.2.3 Add viscoelastic properties

Most liquids from my data-set don't show Newtonian behavior, only 48.2% of them are considered to have a Newtonian behavior. So it is very interesting to be able to add some viscoelastic properties to the liquid material to be able to model well the behavior of the Non-Newtonian liquids. Using only the Abaqus/Explicit interface, the only way to have viscoelastic properties is to use a Maxwell model for the material. A Maxwell model is the association of a spring and a dash-point in series, as shown in the figure 11. For this model σ and ϵ follow this equations:

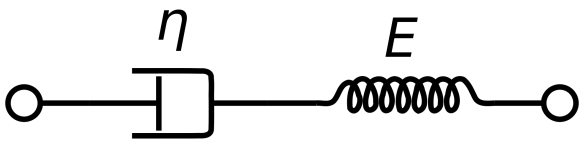
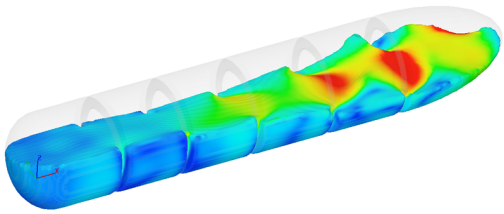


Figure 11: Maxwell model

$$\begin{cases} \sigma &= E.\epsilon_S \\ \sigma &= \eta.\dot{\epsilon}_D \\ \epsilon &= \epsilon_S + \epsilon_D \end{cases} \implies \dot{\sigma} + \frac{E}{\eta}\sigma = E\dot{\epsilon} \tag{9}$$

The problem of the Maxwell model is that it is well suited for viscoelastic solids but not for viscoelastic liquids which means that it is hard to find the couple (E,η) for liquids. Moreover there is no way to give the elastic coefficient E, instead it has to be a shear modulus (G) which is unknown for every liquids. With simulations, there is a possibility to attribute a shear modulus to a liquid based on the behavior that is simulated. For example, I tried with this method to find the shear modulus corresponding to chocolate and water. The results are shown in the table 3.
A model that is more accurate to simulate viscosity's behavior is the Herschel-Bulkley model [15]:

$$\tau(t) = k.\dot{\gamma}^n(t) + \tau_0 \tag{10}$$



Liquids	Density ($kg.m^{-3}$)	Viscosity ($Pa.s$)	Shear modulus (Pa)
Chocolate	1325	0.280	$7.5 \sim 25$
Water	983	0.0013	130

Table 3: Shear modulus of chocolate and water corresponding to the Maxwell model

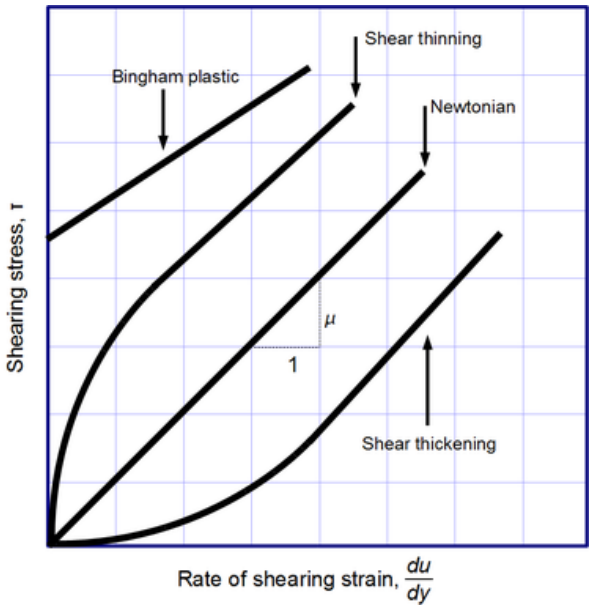
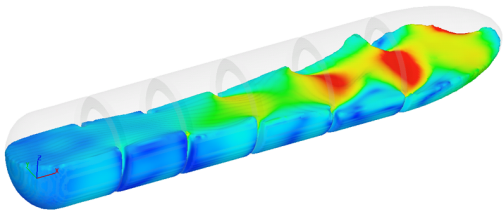


Figure 12: Relationship between τ and $\dot{\gamma}$ depending of the liquid's type.



Pseudoplastic	Newtonian	Dilatant	Bingham
Ketchup	Water	Oobleck	Mayonnaise
Whipped cream	Mineral oil	Quicksand	Toothpaste
Blood	Gasoline	Silly putty	Sludge
Paint	Alcohol		melted chocolate
Nail polish	Honey		

Table 4: Examples of liquids following the Herschel-Bulkley model. Each column represents a category of Non-Newtonian behavior.

Liquids	k $(Pa.s^n)$	n	τ_0 (Pa)
Ketchup [16]	22.56	0.28	-
Blood [17]	0.017	0.708	-
Mayonnaise [18]	45.40	0.495	98.18
melted chocolate [19]	5.764	0.6973	9.096

Table 5: Rheological parameters for the Herschel-Bulkley model for ketchup, blood, mayonnaise and melted chocolate.

(called Power law model if $\tau_0 = 0$), depending on the values of k, n and τ_0, there are different name given to the fluid behavior:

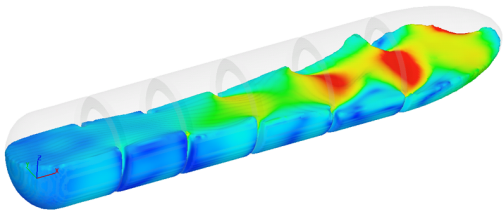
- $\tau_0 > 0$: Bingham fluid
- $n < 1$: Shear thinning fluid
- $n > 1$: Shear thickening fluid
- $n = 1$ and $\tau_0 = 0$: Newtonian fluid

If k isn't constant through time (not in my case):

- $t \mapsto k'(t) > 0$: Rheopectic fluid
- $t \mapsto k'(t) < 0$: Thixotropic fluid

Some example of common liquids are shown in table 4 and sort by their parameters for the Herschel-Bulkley model.
The parameters of the Herschel-Bulkley model for the Non-Newtonian liquids from table 2 are given in table 5.
Because I am using Abaqus/Explicit the Herschel-Bulkley is not available in the interface from the viscosity toolbox (it is available with Abaqus/CFD). So I had to implement a subroutine to add this viscoelastic behavior to the material. The easiest subroutine to use turned out to be a VUVISCOSITY subroutine that allows to give the explicit relationship between the shearing stress (τ) and the shearing strain ($\dot{\gamma}$) by changing the definition of η given in (8) with:

$$\eta(\dot{\gamma}) = \tau_0.1_{\{\dot{\gamma}=0\}} + (\frac{\tau_0}{\dot{\gamma}} + k\dot{\gamma}^{n-1}).1_{\{\dot{\gamma}>0\}} \tag{11}$$



The subroutine is written in Fortran and is available in the appendix B. In order to use it, the line corresponding for the viscosity in the inp file has to be switched with:

*Viscosity, definition=User, Properties =3
$\eta, n, \tau_0,$

and in order for ABAQUS to run the analyse, the abaqus' line command should be:

»abaqus job=inpfile interactive user=VUVISCOSITYsubroutine.f

To make sure that the subroutine works perfectly, I tested the subroutine on a Newtonian liquid that can be modeled by the viscosity toolbox as already introduced. In the figure 13, there are screenshots of water simulated with and without the subroutine with the same other parameters. It shows that the subroutine is working because the results are almost perfectly identical between the two ways to implement the viscosity.

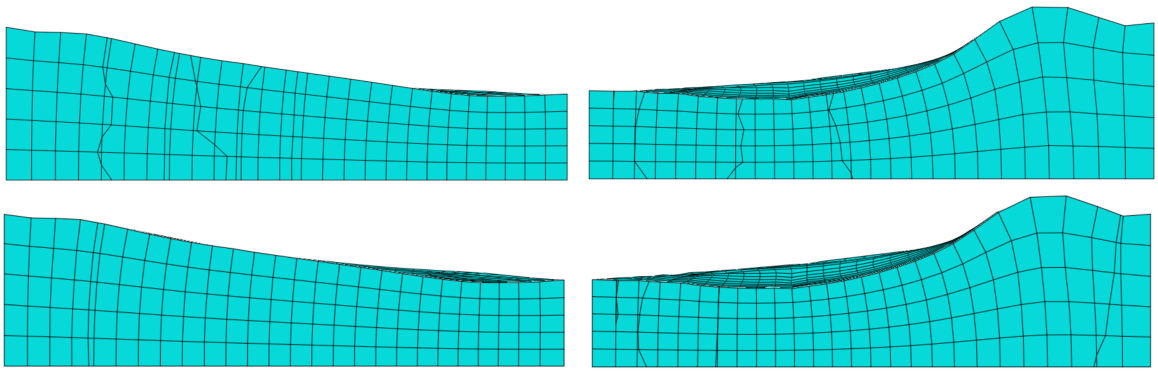
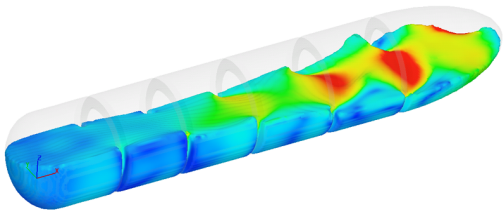


Figure 13: Screenshots from xy plan of the water facing a initial velocity $v = 0.10m/s$. On the left column $t = 0.1s$ and on the right column $t = 0.25s$. The first line correspond to water simulated via the abaqus viscosity toolbox, the second line correspond to water simulated with the use of the VUVISCOSITY subroutine.

3.2.4 Edit mesh

The mesh that I used is made with C3D8R elements (An 8-node linear brick, reduced integration, hourglass control). It is not possible to use C3D8 elements with the ALE mesh, even if using C3D8R implies to have hourglass effects (a pattern of "boom and bust"). Because of the geometry of the liquid, I chose to add a sweep control in order to minimize the mesh transition. The shape of the mesh is shown in figure 14.



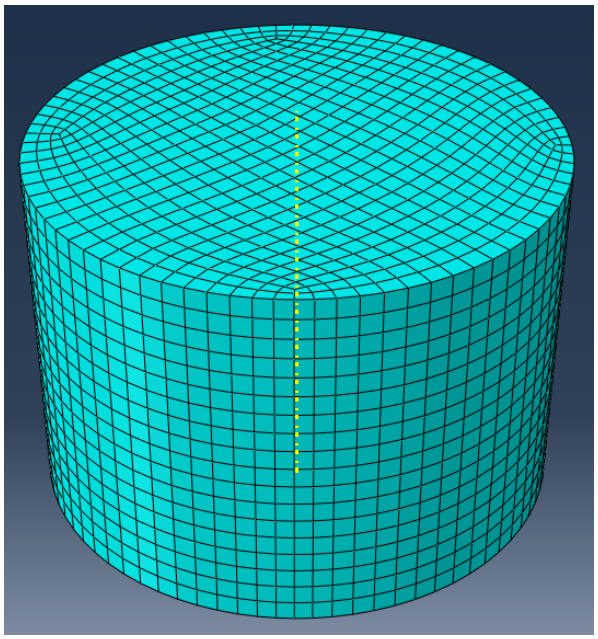
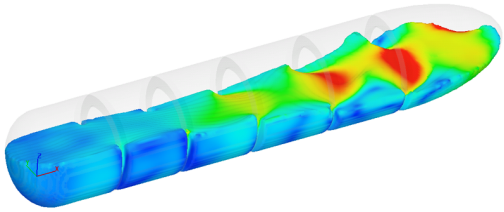


Figure 14: Screenshot from abaqus showing the mesh designed with C3D8R elements. Sweep control is used to minimize the mesh transition. The global seed is $h \approx 0.004$.

To use the ALE mesh, I had to create it in:

»Step »Other »ALE Adaptative Mesh Domain

A lot of parameters can be used to control the ALE mesh, all described in [20]. Because of the nature of the problem, I have used the ALE Adaptative Mesh Controls toolbox with the parameters showed in figure 15. The frequency of using re-meshing sweeps is every 1 iterations and the number of re-meshing sweeps is 5 per increment.



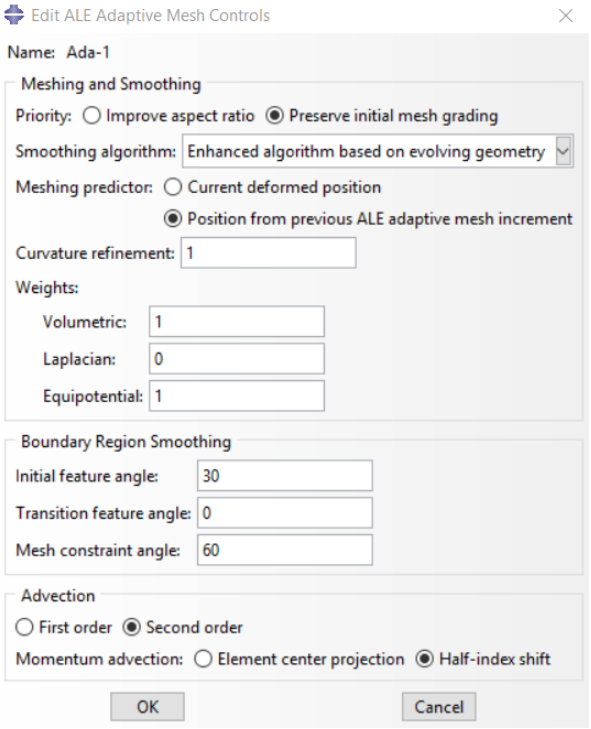


Figure 15: Screenshot from abaqus showing how to control the ALE adaptative mesh.

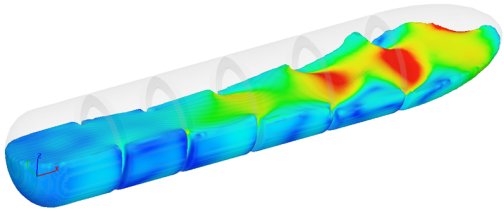
3.2.5 Edit boundary conditions

Still following the work that has been made in [1], the simulation follows two steps:

- Step 1: The glass follows a straight line along an horizontal axis with a constant velocity for $0.1s$
- Step 2: The glass stops moving, the liquid sloshes and the simulation stops when the liquid reach equilibrium.

I only considered velocities that were not too fast for the liquid, in the sense that the liquid would not be divided in several parts during the simulation.
Assuming that the surfaces of the liquid in contact physically with the glass would always remain in contact during the experiment, the boundary conditions can be applied directly to the surfaces in contact with the glass. It allows mainly, as already said before, to not use a contact interaction between the container and the liquid, and so to only have to implement the liquid and not the container.
I have imposed then boundary conditions of type Velocity/Angular velocity. The vertical axis here is along the y coordinate. The translation of the glass is along the x coordinate.
For the external surface on the side, the boundary conditions are:

$$\begin{cases} V_x &= v_0 \text{ in Step 1, 0 in Step 2.} \\ V_z &= 0 \\ \Omega_x &= 0 \\ \Omega_y &= 0 \\ \Omega_z &= 0 \end{cases} \tag{12}$$



For the bottom surface, the boundary conditions are:

$$\begin{cases} V_x & = v_0 \text{ in Step 1, 0 in Step 2.} \\ V_y & = 0 \\ V_z & = 0 \\ \Omega_x & = 0 \\ \Omega_y & = 0 \\ \Omega_z & = 0 \end{cases} \tag{13}$$

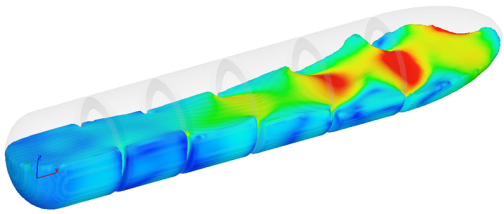
To complete the implementation, I just needed to add a load force to have the gravity applied on the liquid. Here I assumed that $\vec{g} = (0, -9.81, 0)_{\vec{x},\vec{y},\vec{z}}$.

3.3 Simulations

The work from [1] had simulated water with the SPH method. In my work, I have simulated the 6 liquids introduced in table 6 using the ALE mesh. But it is easy to simulate a new liquid when an input file using the ALE mesh is given.
Concretely to simulate a new liquid, given a previous input file (.inp file), you need to change the viscosity parameters k, n and τ_0 from the Herschel-Bulkley model with the ones corresponding to the new liquid. You can change the time-increment, the initial velocity given to the liquid and the time of simulation. The ALE-mesh can be changed too by changing the frequency and the number of mesh sweeps per frequency.
It is possible to simulate every liquid with the ALE mesh based on my computation, however with a high accuracy level, simulations can take a really long time to compute. For example, on my computer, 10 days are not enough to compute 10 seconds of water with 12141 nodes.
So, because of time consumption, I have simulated the less viscous liquids with less mesh accuracy rather than high viscous fluid. As described in table 6, blood has then less nodes than high viscosity liquids (chocolate, honey and mayonnaise). Water and butter have also a height of 2cm (instead of 7cm). With those changes, all simulations can run within 24 hours.
In table 6, the initial velocities were chosen depending on the viscosity of the liquid to make sure that the velocity was high enough to see sloshing with enough magnitude but not too high to don't have extreme deformations.
Right now, using Abaqus/Explicit, there is no direct way to impose incompressibility to a material. This appears to become an issue when simulating low viscous liquids (water, butter and blood), because the liquids compress on themself. One way to change that outcome is to play with the bulk viscosity. However as explained in [21], increasing the bulk viscosity parameter will make the liquid closer to incompressibility but excessively small time-increment will be needed to get the solution. So it is not sure that incompressibility can be solved this way.
Videos of those simulations can be found on Youtube[1].
To extract the data of those simulations and work on them later on, I have to extract variables from abaqus and to convert them to be readable on Matlab.
For each nodes, I have extracted the values of those variables every $10^{-3}s$:

- position, in $\mathbb{R}^3$
- velocity, in $\mathbb{R}^3$
- the internal energy, in $\mathbb{R}$
- the stress components, in $\mathbb{R}^6$

[1] https://youtu.be/8iUi5cn7Ups

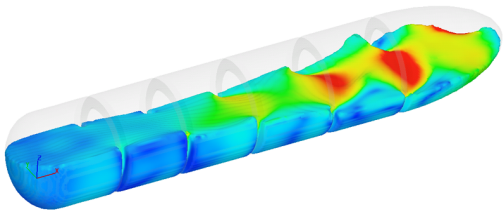


Liquids	velocities $v_0(m/s)$				Time of simulation (s)	Number of nodes
Blood	0.05	0.1	0.15	0.2	5.1	1790
Butter	0.05	0.1	0.15	0.2	5.1	716
Honey	0.25	0.5	0.75	0.9	0.6	12141
Mayonnaise	0.25	0.5	0.75	0.9	0.6	12141
melted chocolate	0.25	0.5	0.75	0.9	0.6	12141
Water	0.05	0.1	0.15	0.2	5.1	716

Table 6: Simulations done and the parameters corresponding to them. Videos of the simulations are available on Youtube.

The manifold of the liquid at a time given is assumed to be the set of the variables from all the nodes of the liquid. Positions, velocities and internal energies come from the known manifold of Newtonian liquid but to add a principle of time-history there is also the information given by the stress.
To sum up, every simulation of a liquid can be represented with a data matrix where every column correspond to a time step. The dimensions are:

- Blood: $\mathbb{R}^{23270\times5101}$
- Butter: $\mathbb{R}^{9308\times5101}$
- Honey: $\mathbb{R}^{157833\times601}$
- Mayonnaise: $\mathbb{R}^{157833\times601}$
- melted chocolate: $\mathbb{R}^{157833\times601}$
- Water: $\mathbb{R}^{9308\times5101}$



4 GENERIC STRUCTURE

The thermodynamically consistent integrator that has been used in [1] is based on the GENERIC structure, following the work in [2]. The next sections will explain how the integrator is elaborated.

4.1 GENERIC FORMALISM

The manifold of a problem represents the true dimension of the problem. Concretely, a manifold is a set of variables that entirely describe the behavior of a problem. For example a particle in a Newtonian fluid is completely described by it's position, velocity and internal energy, so the dimension of a particle's manifold can be 7. For N particles, the dimension would be $D = 7^N$. Following the geometry from figure 9, the corresponding number of particles of water would be $N \approx 10^{25}$.
Let's $\Gamma_t \in \mathscr{C}^1(\mathscr{I})$ be the manifold of a problem:

$$\Gamma_t = \Gamma(t) : \mathscr{I} = [0, T] \to \mathscr{S} \subset \mathbb{R}^D \tag{14}$$

The dynamic of the system is assumed to be given by:

$$\frac{d}{dt}\Gamma_t = L(\Gamma_t)\nabla E(\Gamma_t) + M(\Gamma_t)\nabla S(\Gamma_t) \tag{15}$$

In this formalism, L is the Poisson-matrix which represents the conservative part of the evolution and M is the friction-matrix which represents the irreversible part of the evolution. It should follow then: $L(\Gamma_t).\nabla S(\Gamma_t) = 0$ and $M(\Gamma_t).\nabla E(\Gamma_t) = 0$.
To ensure the conservation of energy in the system, L has to be skew-symmetric.
To ensure the second-law of thermodynamics, M has to be symmetric and semi-positive.
Finally with the structures of L and M, and the two previous degeneracy conditions, the first and second principles of Thermodynamics are ensured:

$$\begin{cases} \frac{d}{dt}E(\Gamma_t) & = 0 \\ \frac{d}{dt}S(\Gamma_t) & \geq 0 \end{cases}$$

4.2 GENERIC RESOLUTION

The discretisatoin in time of (15) gives the explicit scheme:

$$\frac{\Gamma_{n+1} - \Gamma_n}{\Delta t} = L(\Gamma_n)\nabla E(\Gamma_n) + M(\Gamma_n)\nabla S(\Gamma_n) \tag{16}$$

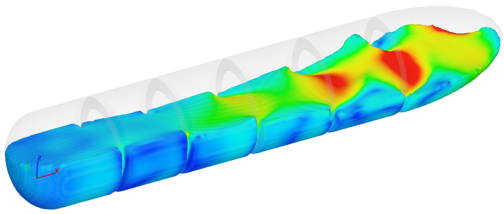
With FEM methods, it is possible to get approximations for $\nabla E(\Gamma_n)$ and $\nabla S(\Gamma_n)$:

$$\begin{cases} \nabla E(\Gamma_n) & \approx A_n.\Gamma_n \\ \nabla S(\Gamma_n) & \approx B_n.\Gamma_n \end{cases}$$

So the numerical scheme (16) becomes:

$$\frac{\Gamma_{n+1} - \Gamma_n}{\Delta t} = L(\Gamma_n).A_n.\Gamma_n + M(\Gamma_n).B_n.\Gamma_n \tag{17}$$

The next step here is to find the matrices L, A, M and B. With the access to Z, a set of $(\Gamma_t)_{t\in\mathscr{I}^z}$, we can then resolve (17) to get the values of $L(\Gamma_t)$, $A(\Gamma_t)$, $M(\Gamma_t)$ and $B(\Gamma_t)$ for $t \in \mathscr{I}^Z$:
Finding $L(\Gamma_t)$, $A(\Gamma_t)$, $M(\Gamma_t)$ and $B(\Gamma_t)$ for $t \in \mathscr{I}^Z$ such that:



$$\begin{cases} \frac{\Gamma_{t+\delta t}-\Gamma_t}{\delta t} = L(\Gamma_t).A_t.\Gamma_t + M(\Gamma_t).B_t.\Gamma_t, \forall t \in \mathscr{I}^Z \\ L \text{ skew-symmetric, and } M \text{ symmetric, semi-positive} \end{cases}$$

It is possible to solve the problem with $L(\Gamma)$ known (partial) or not (unpartial). It is also possible to add the constraints $L.B.\Gamma = 0$ and $M.A.\Gamma = 0$ (problem constrained) or not (problem unconstrained). Now with the set $Z = \{\Gamma_t, t \in \mathscr{I}^Z\}$ it is possible to get an approximation of the values of $L(\Gamma)$, $A(\Gamma)$, $M(\Gamma)$ and $B(\Gamma)$, $\forall \Gamma \in \mathscr{S}$ with interpolating methods. Giving a weight-matrix W the interpolation of L_n, A_n, M_n and B_n are:

$$L_n^{\mathrm{f}} = \sum_{k \in \mathscr{I}^Z} W_{n,k} L(\Gamma_k) \tag{18}$$

$$A_n^{\mathrm{f}} = \sum_{k \in \mathscr{I}^Z} W_{n,k} A(\Gamma_k) \tag{19}$$

$$M_n^{\mathrm{f}} = \sum_{k \in \mathscr{I}^Z} W_{n,k} M(\Gamma_k) \tag{20}$$

$$B_n^{\mathrm{f}} = \sum_{k \in \mathscr{I}^Z} W_{n,k} B(\Gamma_k) \tag{21}$$

Knowing the interpolating function f, the numerical scheme (17) can be used again to get the values of $\tilde{\Gamma}_t$:

$$\begin{cases} \frac{\tilde{\Gamma}_{n+1}-\tilde{\Gamma}_n}{\Delta t} = L_n^{\mathrm{f}}.A_n^{\mathrm{f}}.\tilde{\Gamma}_n + M_n^{\mathrm{f}}.B_n^{\mathrm{f}}.\tilde{\Gamma}_n \\ \tilde{\Gamma}_0 = \Gamma_0 \end{cases} \tag{22}$$

The weight matrix W_n is determined by the neighborhood of the point $\tilde{\Gamma}_n$ and the topological space defined.

4.3 GENERIC NUMERICAL RESOLUTION ON MATLAB

In my work, the problem (17) is partial and constrained.
To find the values of L, A, M and B in (17) knowing Z, for my work, I have assumed that the matrix L was known (problem partial):

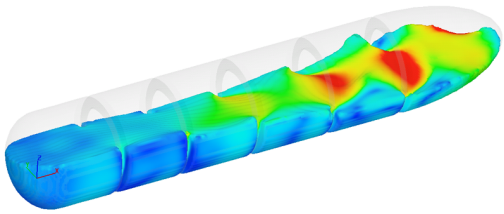
$$L = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \text{in } \mathbb{R}^2 \tag{23}$$

L is skew-symmetric.
Then with the function *fmincon* in Matlab, the matrix A, B and M is determined solving:

$$\underset{A,B,M}{argmin} f, \text{ such that } \forall t \in [t_0, T_f], c(A,t) = 0 \text{ and } d(B,t) = 0 \tag{24}$$

with:

$$\begin{cases} f(M,A,B,t_0,T_f) & = \sum_{i=t_0}^{T_f} \frac{1}{N} \sqrt{||\frac{\Gamma_i - \Gamma_{i-1}}{\delta t} - L.A.\Gamma_{i-1} - M.B.\Gamma_{i-1}||^2} \\ c(A,t) & = M.A_t.\Gamma_t \\ d(B,t) & = L.B_t.\Gamma_t \end{cases}$$



4.4 GENERIC complexity

As previously said, the manifold dimension is too high ($D = 7^N$ of a Newtonian fluid). This implies that many parameters have to be actualised during the numerical resolution (22), the interpolating function f is complex and the resolution is too time-consuming.
For those reasons, it is better to previously use reduction method (Υ) on Z and then use the GENERIC structure on the new data $\Upsilon(Z)$. This method allows to decrease the dimension of the data ($D \xrightarrow{\Upsilon} M, M \ll D$), to have easier interpolating function f and to handle real-time calculus. The choice of Υ will be discussed in an next section of the report.

4.5 GENERIC correction

Following the work of [22], it is possible to match more experimental data with models.Assuming to have a GENERIC model with $L, \nabla E, M$ and ∇S known, we want to the model to match new data obtained in $Z_{\Gamma_{exp}}$.
Following the scheme (22), I will obtain a new set of values $Z_{\Gamma_{mod}}$. Presumably, I don't have $Z_{\Gamma_{exp}} = Z_{\Gamma_{mod}}$. Yet I want to keep the values of $L, \nabla E, M$ and ∇S because they are hypothetically correct for the model. So I am going to add a correction term in the model in order to get:

$$\Gamma_{exp} = \Gamma_{mod} + \Gamma_{corr} \tag{25}$$

The associated set to correction is:

$$Z_{\Gamma_{corr}} = Z_{\Gamma_{exp}} - Z_{\Gamma_{mod}}$$

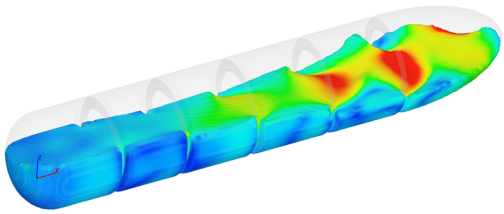
Since I still want that the correction data to follow the GENERIC structure, I can apply the GENERIC resolution with $Z_{\Gamma_{corr}}$ solving the partial/unpartial, constrained/unconstrained system:

$$\begin{cases} \frac{\Gamma^{corr}_{t+\delta t} - \Gamma^{corr}_t}{\delta t} = L^{corr}(\Gamma^{corr}_t).A^{corr}_t.\Gamma^{corr}_t + M^{corr}(\Gamma^{corr}_t).B^{corr}_t.\Gamma^{corr}_t, \forall t \in \mathscr{I}^{Z_{\Gamma_{corr}}} \\ L^{corr} \text{ skew-symmetric, and } M^{corr} \text{ symmetric, semi-positive} \end{cases}$$

Then experimental data are following the new GENERIC structure:

$$\begin{aligned} \frac{d}{dt}\Gamma_t = & L(\Gamma_t)\nabla E(\Gamma_t) + M(\Gamma_t)\nabla S(\Gamma_t) \\ & + L^{corr}(\Gamma_t)\nabla E^{corr}(\Gamma_t) + M^{corr}(\Gamma_t)\nabla S^{corr}(\Gamma_t) \end{aligned}$$

Which is based on the model built before, so using $L, \nabla E, M$ and ∇S. It means that once I have a data-set from abaqus' simulations, I can use the corrected model on a different liquid that the one used in the simulations.



5 Reduction methods

In this section, I will explain some reduction model methods. In order to illustrate clearly their efficiency, I have used every reduction model on a concrete example: reducing the Klein bottle (3D objet) in a 2D space (dimension of its manifold). The shape of the Klein bottle is represented in the figure 16.

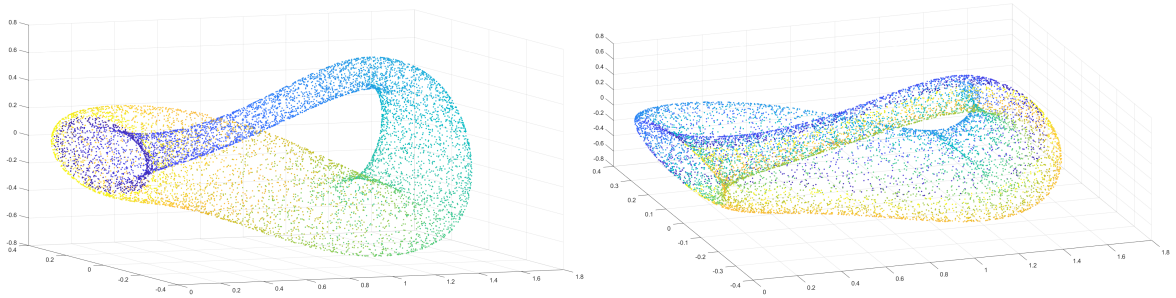


Figure 16: *10000 points uniformly chosen of the Klein bottle, colored along the u parameter on the left and colored along the v parameter on the right.*

In my example, the Klein bottle is a non-oriented surface that can be implemented easily because $\{(x,y,z)(u,v) : u,v \in [0,2\pi[\}$ is the Klein bottle. A point (x,y,z) is defined as showed in (26).

$$\begin{cases} x(u,v) &= \frac{\sqrt{2}(20u^3-65\pi u^2+50\pi^2 u-16\pi^3)\cos(v)(\cos(u)(3\cos^2(u)-1)-2\cos(2u))}{80\pi^3\sqrt{8\cos^2(2u)-\cos(2u)(24\cos^3(u)-8\cos(u)+15)+6\cos^4(u)(1-3\sin^2(u))+17}} - \frac{3\cos(u)-3}{4} \\ y(u,v) &= -\frac{(20u^3-65\pi u^2+50\pi^2 u-16\pi^3)\sin(v)}{60\pi^3} \\ z(u,v) &= -\frac{\sqrt{2}(20u^3-65\pi u^2+50\pi^2 u-16\pi^3)\cos(v)\sin(u)}{15\pi^3\sqrt{8\cos^2(2u)-\cos(2u)(24\cos^3(u)-8\cos(u)+15)+6\cos^4(u)(1-3\sin^2(u))+17}} \\ &\quad + \frac{\sin(u)\cos^2(u)+\sin(u)}{4} - \frac{\sin(u)\cos(u)}{2} \end{cases} \tag{26}$$

Giving the fact that the reduced space has the same dimension than the manifold space, a good reduction model method on this example should be able in 2D to completely describe the shape of the object as does the parameters u and v in (26).
I will use the notation $Z \in \mathbb{R}^{n\times p}$ to represent the data-set that has to be embedded in a reduced space (n is the number of points and p is the dimension of a point).

5.1 POD: Proper Orthogonal Decomposition

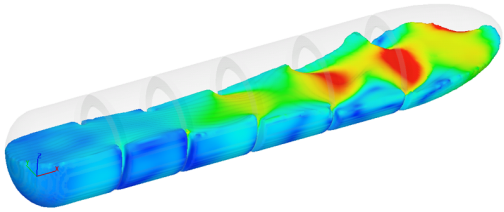
The POD aims to look for the eigenvalues λ_j and eigenvectors V_j of the matrix $Q = Z^T.Z \in \mathbb{R}^{n\times n}$. By sorting $(\lambda_j, V_j)_{j\in 1,...,n}$ in the ascending order based on the value of λ. Then the new basis is:

$$\Xi = \{u_1, ..., u_M\}$$

Where $u_j = \frac{1}{\sqrt{\lambda_j}} Z.V_j$
Given a new point z, its embedding $(\check{z})$in the reduced space is:

$$\check{z} = \Upsilon(z) = \sum_{j=1}^{M} (z^T.u_j)u_j \tag{27}$$

The embedded Klein bottle following this method is showed in figure 17. The geometric shape has not really be changed by POD, it induces that points coming from two different parts of the surface are mixed together.



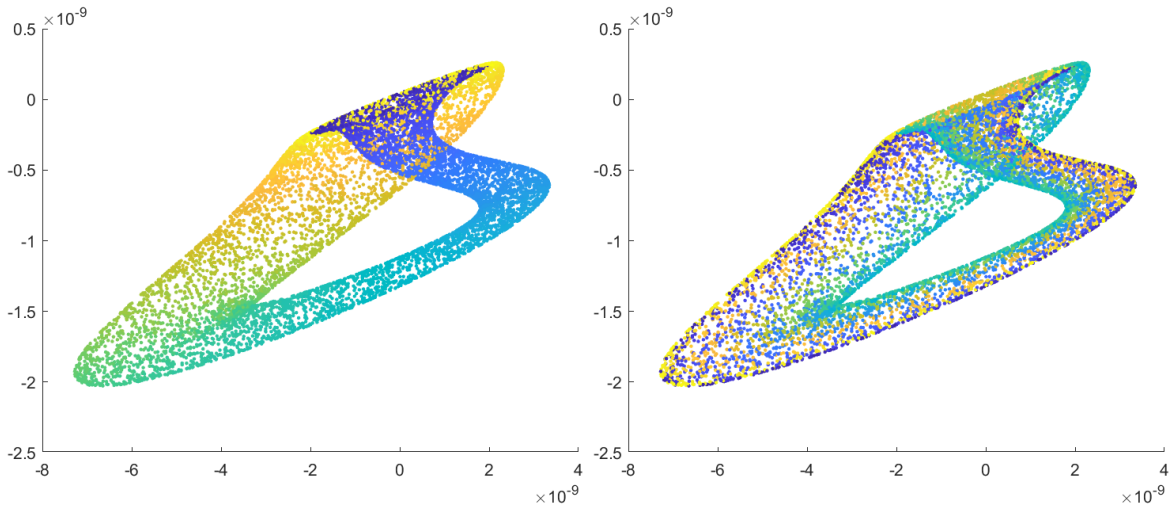


Figure 17: Klein bottle after POD, colored along the u parameter on the left and along the v parameter on the right

5.2 LLE: Local Linear Embedding

The main idea is to keep the neighbours for each point in the embedded space. Choosing K, the number of closest neighbours, the LLE tries to find W solution of:

$$\begin{cases} W = \epsilon(W) = \sum_{i=1}^{n} |z_i - \sum_{j=1}^{n} W_{ij} z_j|^2 \\ W_{ij} = 0 \text{ if } z_j \text{ isn't in the K nearest neighbours of } z_i \\ \sum_j W_{ij} = 1, W_{ii} = 0 \end{cases} \tag{28}$$

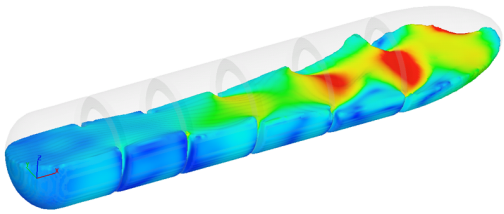
Then, to find the new points $\Xi = \{\xi_1, ..., \xi_n\}$ solving:

$$\begin{cases} \Xi = (\sum_{i=1}^{n} |\xi_i - \sum_{j=1}^{n} W_{ij} \xi_j|^2) \\ \sum_i \xi_i = 0 \\ \xi.\xi^T = I_M \end{cases} \tag{29}$$

Given a new point z, its embedding ($\check{z}$)in the reduced space is:

$$\begin{cases} \check{z} = \Upsilon(z) = \sum_{j=1}^{K} \alpha_j \xi_{\sigma(j)}, \ \sigma(j) \text{ represents the K nearest neighbours} \\ \alpha = |z - \sum_{j=1}^{K} \alpha_j z_j|^2 \end{cases} \tag{30}$$

The embedded Klein bottle following this method is showed in figure 18. The geometric shape is not changed like with the POD even though the Klein bottle is less distorted here. But the problem that points coming from two different parts of the surface are mixed together, still remain.



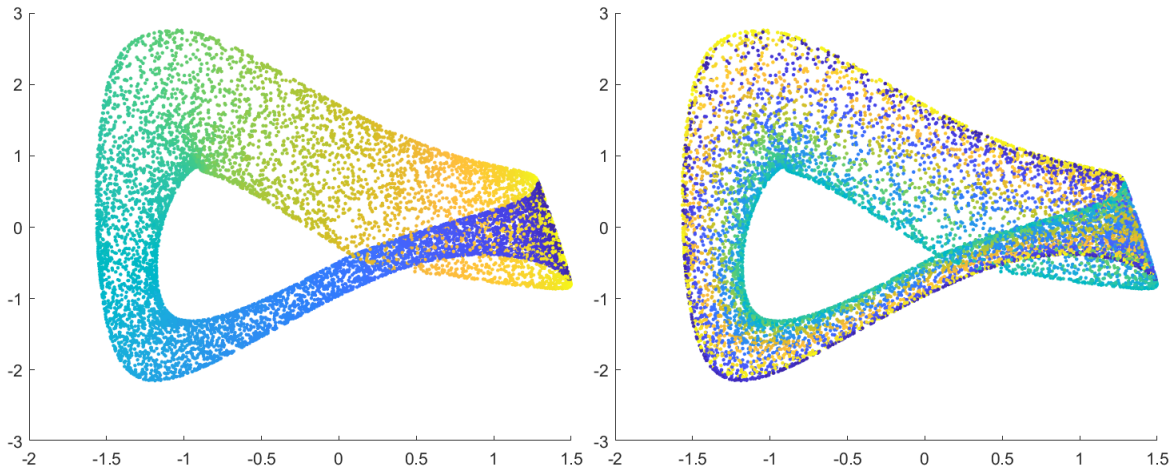


Figure 18: Klein bottle after LLE,colored along the u parameter on the left and the v parameter on the right

5.3 t-SNE: t-Distributed Stochastic Neighbor Embedding

The previous methods being linear, they are not adapted to non-linear behavior observed in fluid mechanics. The two next methods show way better results and so, are more complex mathematically. For the t-SNE [23], a conditional similarity is defined between each snapshot of the data-set, given a distance matrix D it follows:

$$\begin{cases} p_{(i|j)} = \frac{\exp(-\frac{D_{ij}^2}{2\sigma_i^2})}{\sum_{k\neq i}\exp(-\frac{D_{ik}^2}{2\sigma_i^2})} \\ p_{(i|i)} = 0 \end{cases} \tag{31}$$

The similarity matrix is then:

$$P_{i,j} = \frac{p_{(i|j)} + p_{(j|i)}}{2n} \tag{32}$$

The parameters σ_i is chosen in order to respect a good perplexity parameter of the data set:

$$\begin{cases} Perp(P_i) & = 2^{H(P_i)} \\ H(P_i) & = -\sum_j p_{(j|i)} \log_2 p_{(j|i)} \end{cases} \tag{33}$$

Usually perplexity is fixed between 5 and 50. A conditional similarity is then defined for the new points:

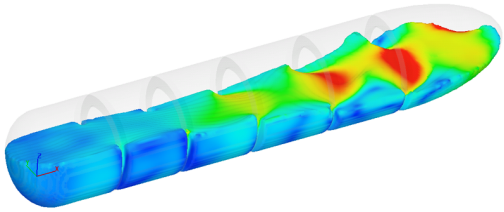
$$q_{(i|j)} = \frac{\exp(-d(\xi_i,\xi_j)^2)}{\sum_{k\neq i}\exp(-d(\xi_i,\xi_k)^2)} \tag{34}$$

To find the new points $\Xi = \{\xi_1, ..., \xi_n\}$, the algorithm minimizes the Kullbrack-Leiber:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{(j|i)} \log(\frac{p_{(j|i)}}{q_{(j|i)}}) \tag{35}$$

The minimization is done with a gradient descent of C based on:

$$\frac{\delta C}{\delta \xi_i} = 2\sum_j (p_{(j|i)} + p_{(i|j)} - (q_{(j|i)} + q_{(i|j)}))(\xi_i - \xi_j) \tag{36}$$



Given a new point z, its embedding ($\check{z}$) in the reduced space is:

$$\begin{cases} \check{z} & = \Upsilon(z) = \underset{\xi}{\sum_{j=1}} p_j \log(\frac{p_j}{q_j}) \\ p_j & = \frac{\exp(-\frac{d(z,z_j)^2}{2\sigma})}{\sum_k \exp(-\frac{d(z,z_k)^2}{2\sigma})} \\ q_j & = \frac{\exp(-\frac{d(,\xi_j)^2}{2\sigma})}{\sum_k \exp(-\frac{d(,\xi_k)^2}{2\sigma})} \\ H(z) & = H(P_i) \end{cases} \tag{37}$$

The embedded Klein bottle following this method is showed in figure 19. The shape of the Klein bottle has really changed, enough to don't be recognized. However, through the colors, it is possible to see that the reduction kept the neighbourhoods and so kept the information of the data-set. Clusters are also present, meaning that the algorithm detected different behavior around the surface which will be interesting if a cluster would correspond to a parameter known.

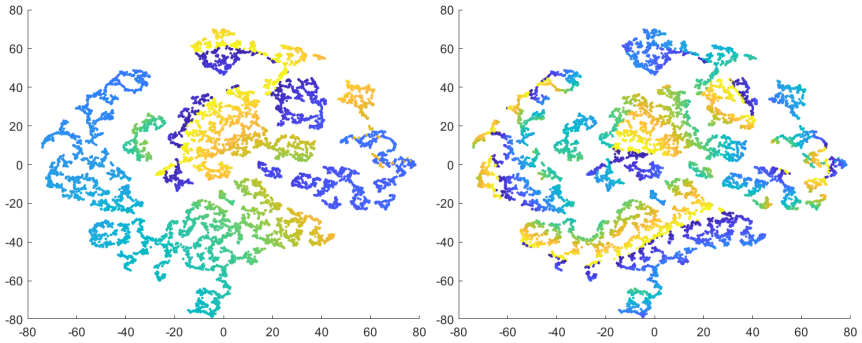


Figure 19: Klein bottle after t-SNE, colored along the u parameter on the left and the v parameter on the right

5.4 UMAP: Uniform Manifold Approximation and Projection

The UMAP method is a new model reduction method [5] that can directly compete with the results of a t-SNE. It allows to conserve the geodesic distance in a data-set
For every point X_j in the ball centered in X_i, the geodesic distance between X_i and X_j is approximated by:

$$d_{\mathscr{M}}(X_i, X_j) = \frac{1}{r_i} d_{\mathbb{R}^n}(X_i, X_j) \tag{38}$$

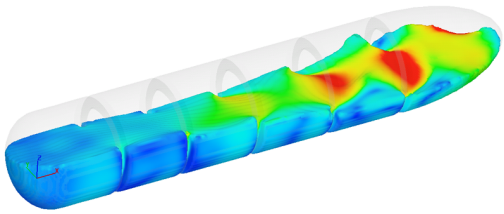
Considering that every point X_i has k-nearest neighboors that are the points in the ball of center X_i (giving r_i), a parameter ρ_i can be defined for every point:

$$\rho_i = \min(d_{\mathscr{M}}(x_i, x_{i_j}) | 1 \leq j \leq k : d_{\mathscr{M}}(x_i, x_{i_j}) > 0) \tag{39}$$

and a parameter σ_i in sort that:

$$\sum_{j=1}^{k} \exp(-\frac{\max(0, d_{\mathscr{M}}(x_i, x_{i_j}) - \rho_i)}{\sigma_i}) = \log_2(k) \tag{40}$$

A weighted directed graph $\bar{G} = (V, E, w)$ can by then defined.Where the vertices V are the points $(X_i)_{1 \leq i \leq N}$, the oriented edges E are $\{(x_i, x_{i_j}) | 1 \leq j \leq k, 1 \leq i \leq N\}$ and the weights w are $w(x_i, x_{i_j}) = \exp(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i})$ We note A the weight matrix.



A being non-symmetric, it means that the weight can not be, yet, interpreted as a distance. But the symmetric matrix:

$$B = A + A^T - A \circ A^T \tag{41}$$

Where $\circ$ is the Hadamard product, can be defined instead. Then using B as the weight matrix, a new weighted non-oriented graph $G = (V, E, w_n)$ is defined.
Then in order to get the position of the points y_i in the embedded space, each point position is actualised by the cost function:

$$C((E, w_n), (E, w_d)) = \sum_{e \in E} w_n(e) \log(\frac{w_n(e)}{w_d(e)}) + (1 - w_n(e)) \log(\frac{1 - w_n(e)}{1 - w_d(e)}) \tag{42}$$

Where $w_d(e) = \exp(-\frac{\max(0, d_{\mathbb{R}^d}(x_i, x_{i_j}) - \rho_i)}{\sigma_i})$ is the weight in the embedded space of the edge $e \in E$.
The first part tends to respect the edges with high weights and the second one tends to associate a low weight when an edge has a small weight in the manifold.
Given a new point z, its embedding ($\check{z}$)in the reduced space is:

$$\begin{cases} \check{z} & = \underset{\check{z} \in \mathbb{R}^d}{argmin} C((E_z, w_n), (E_z, w_d)) \\ w(z, x_{z_j}) & = \exp(-\frac{\max(0, d_{\mathscr{M}}(z, y_{z_j}) - \rho_z)}{\sigma_z}) \\ \sigma_z & \mid \sum_{j=1}^{k} \exp(-\frac{\max(0, d_{\mathscr{M}}(z, x_{z_j}) - \rho_z)}{\sigma_z}) = \log_2(k) \\ \rho_z & = \min(d_{\mathscr{M}}(z, x_{z_j}) \mid 1 \leq j \leq k : d_{\mathscr{M}}(z, x_{z_j}) > 0) \end{cases} \tag{43}$$

E_z has the edges connected from z to its neighbours.
The embedded Klein bottle following this method is showed in figure 20. This is the best reduction for the Klein bottle because there is no crossing-over of the points not like in the POD and the LLE, the initial shape is conserved not like in the t-SNE. An other reason to use UMAP is that it runs quicker than the t-SNE.

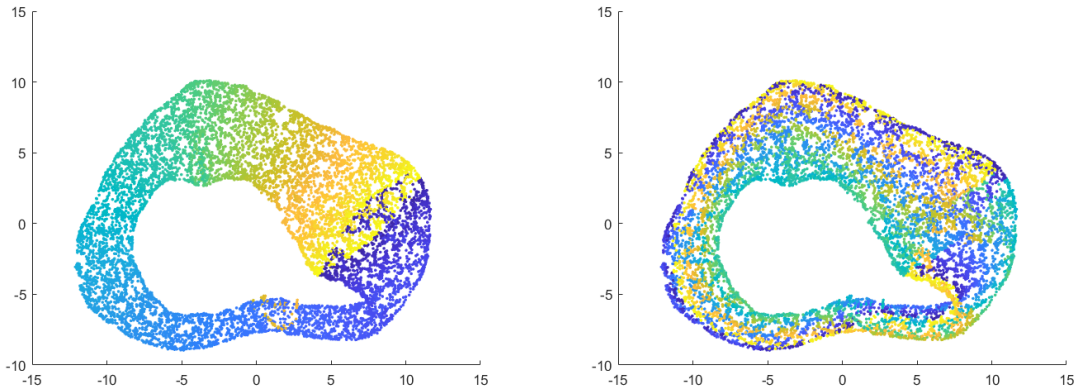
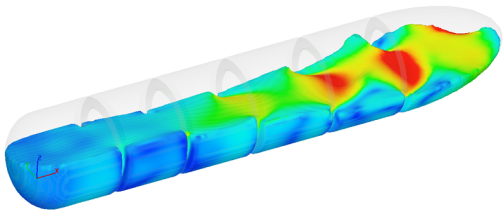


Figure 20: Klein bottle after UMAP, colored along the u parameter on the left and the v parameter on the right

5.5 Interpolation

In (18), (19), (20) and (21) I have defined the way to interpolate the matrix L, A, M and B based on the values of them already known and a weight-matrix W. I am going to explain how to get W.



The weight-matrix W has to be found at each time-step, the weights depends of the current point $\tilde{\Gamma}_t$ and its neighborhood.
Because I have used mainly the UMAP reduction-method, the weight-matrix corresponds to the weights coming from that method:

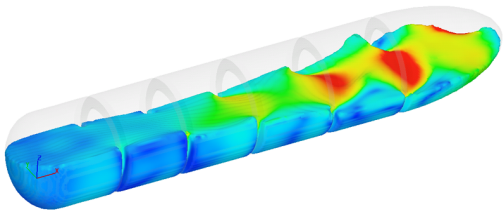
$$\begin{cases} W(\tilde{\Gamma}_t, z) & = \exp(-\frac{\max(0, d_{\mathbb{R}^d}(\tilde{\Gamma}_t, z) - \rho_{\tilde{\Gamma}_t})}{\sigma_{\tilde{\Gamma}_t}}) \text{ if z is in the neighborhood of } \tilde{\Gamma}_t \\ W(\tilde{\Gamma}_t, z) & = 0 \text{ otherwise} \\ \sigma_{\tilde{\Gamma}_t} & | \sum_z \exp(-\frac{\max(0, d_{\mathbb{R}^d}(\tilde{\Gamma}_t, z) - \rho_{\tilde{\Gamma}_t})}{\sigma_{\tilde{\Gamma}_t}}) = \log_2(N) \\ \rho_{\tilde{\Gamma}_t} & = \min(d_{\mathbb{R}^d}(\tilde{\Gamma}_t, z) : d_{\mathbb{R}^d}(\tilde{\Gamma}_t, z) > 0) \end{cases} \tag{44}$$

Where the neighborhood of a point corresponds to its N closest neighbors.
The number of neighbors N to choose can be different from the one used in the UMAP method.

5.6 From the embedded space to the initial space

Now with the GENERIC scheme and the interpolation method, it is possible obtain, starting from an initial, new points associated with the weight-matrix used $(\tilde{\Gamma}_n, W_n)_n$.
The question now is knowing $(\tilde{\Gamma}_n, W_n)_n$, how can I find the corresponding points of $(\tilde{\Gamma}_n)_n$ in the initial space?
The fact is that of the UMAP projection is constructed in order to make sure that the weights from $(W_n)_n$ are the same in the initial and embedded space. So I can assumed that a could approximation of $\tilde{\Gamma}_n$ in the initial space is:

$$\hat{\Gamma}_n = \sum_k W_{k,n} \Gamma_k \tag{45}$$



6 Results

6.1 Reduction method applied to simulations

From the simulations obtained with Abaqus, I had to check if data could be projected in an embedded space using one of the reduction methods detailed in the previous section.
I have decided to embed all the trajectories in 3D using t-SNE. Concretely because all the liquids doesn't have the same dimension of representation (different accuracy in the mesh), all data don't have the same manifold dimension: $\mathbb{R}^{157833}$ for chocolate, mayonnaise and honey, $\mathbb{R}^{23270}$ for blood, $\mathbb{R}^{9308}$ for butter and water.
So in order to use a reduction method on all the liquid, I have decided to take for each liquid a manifold a dimension $\mathbb{R}^{9308}$.
Finally starting from the initial space $\mathbb{R}^{55848\times66024}$, I have obtained the projected trajectories in $\mathbb{R}^{3\times66024}$ showed in the figure 21.

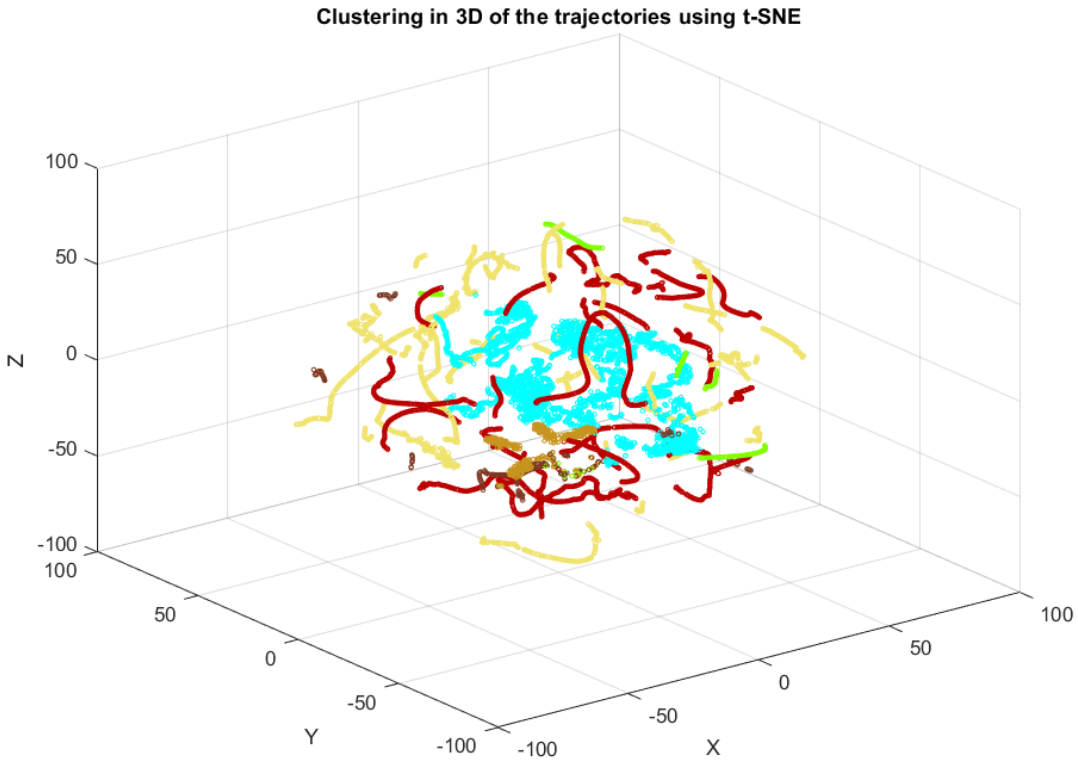
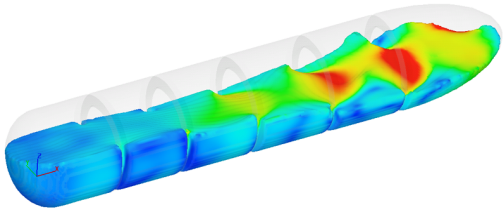
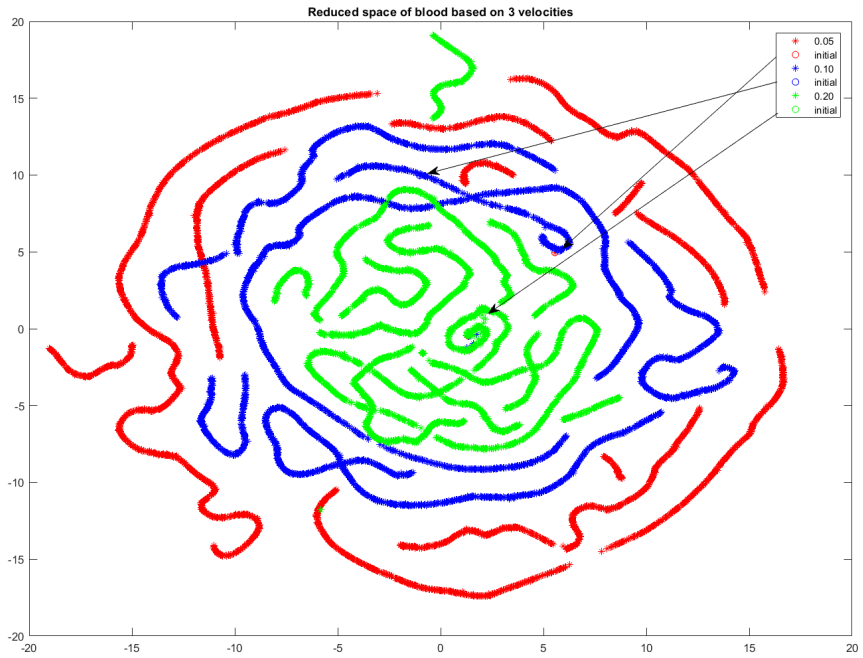


Figure 21: Clustering of the different liquids using t-SNE. Every color correspond to a liquid, blue (water), butter (yellow), blood (red), chocolate (dark brown), honey (brown) and mayonnaise (green).

In the figure 21, every color corresponds to a liquid: blue for water, yellow for butter, red for blood, dark for chocolate, brown for honey and green for mayonnaise.





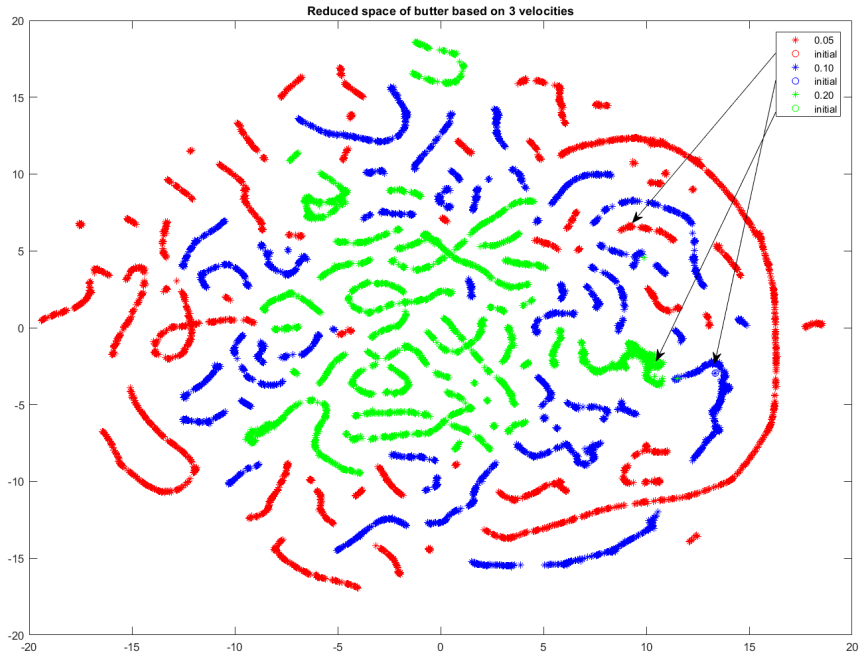
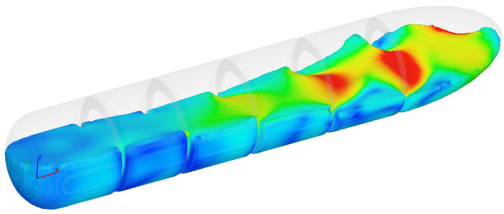
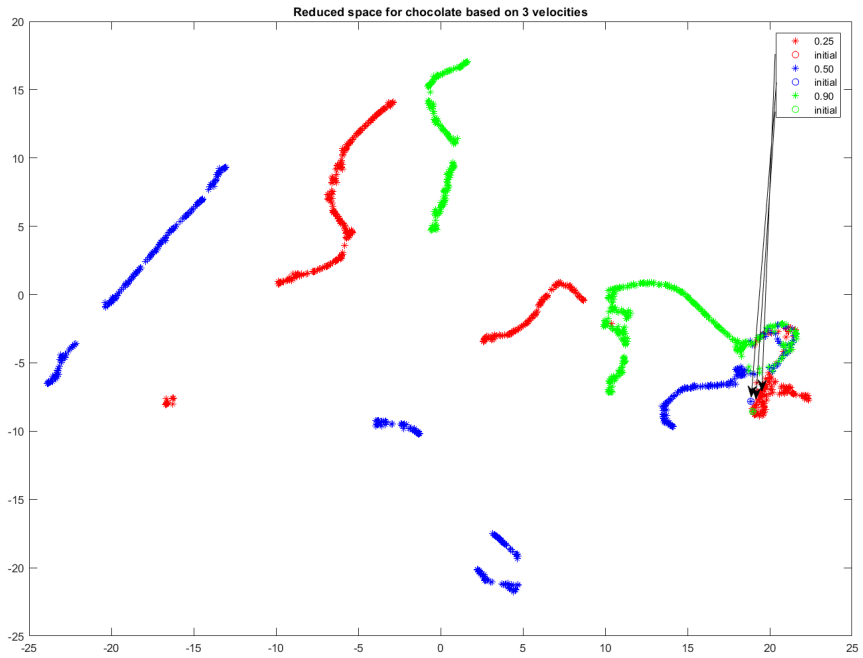


Figure 22: 2D embedded spaces of blood and butter using UMAP. Each color is a different trajectory (velocity). Arrows point to the first point of each trajectory.





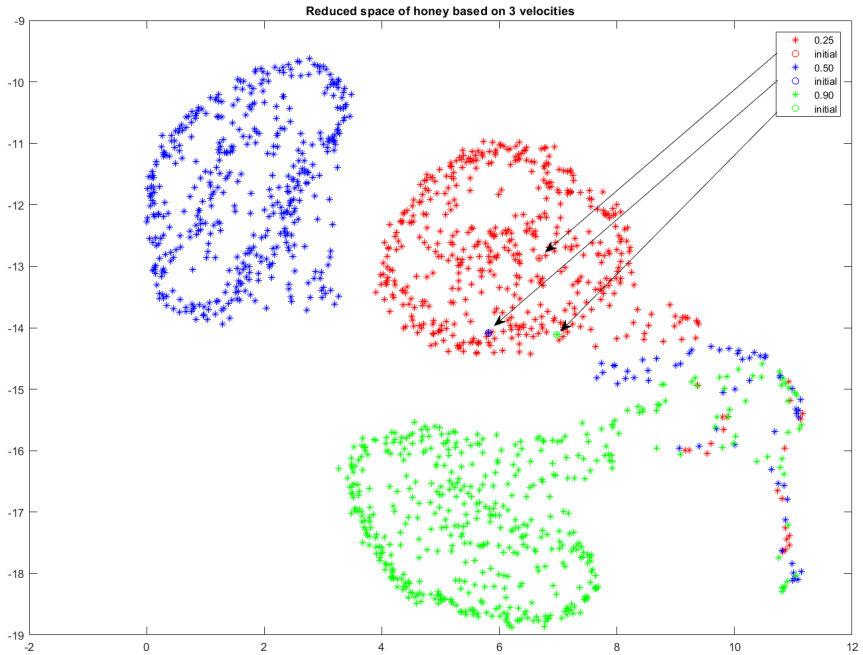
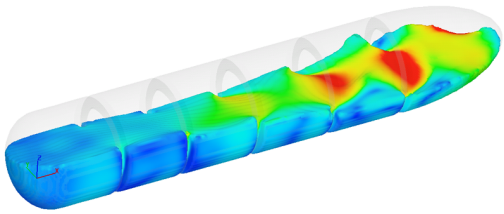
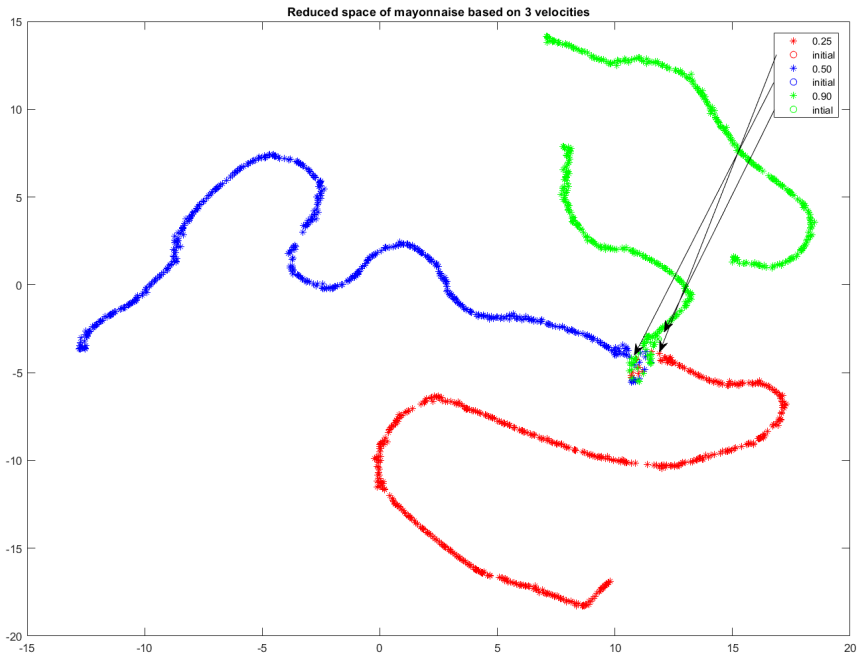


Figure 23: *2D embedded spaces of chocolate and honey using UMAP. Each color is a different trajectory (velocity). Arrows point to the first point of each trajectory.*





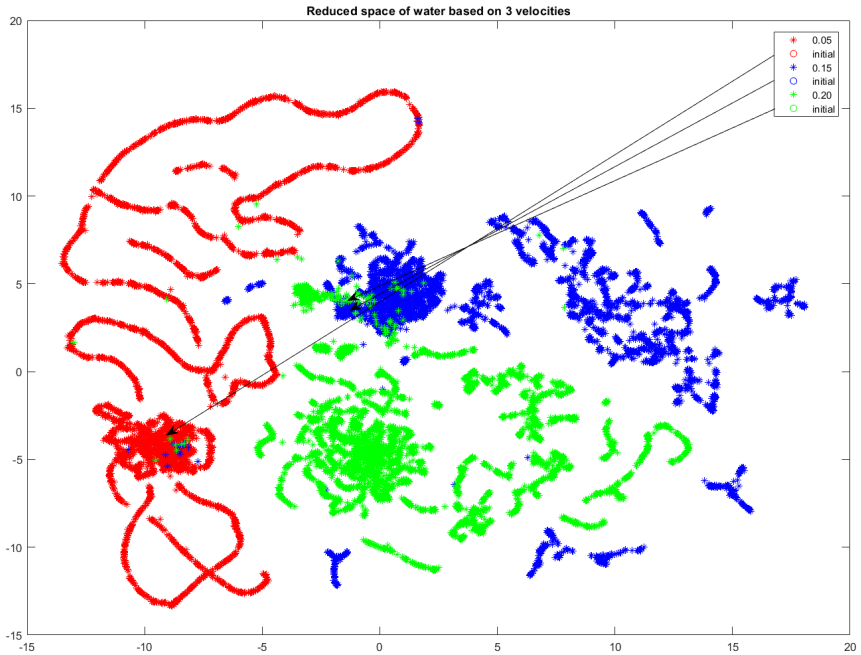
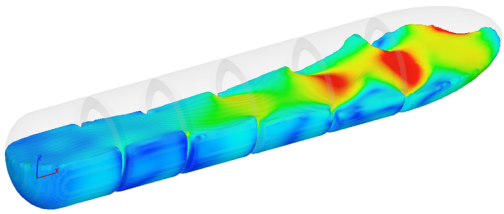


Figure 24: 2D embedded spaces of mayonnaise and water using UMAP. Each color is a different trajectory (velocity). Arrows point to the first point of each trajectory.



It is remarkable to see that all the trajectories are separated by the t-SNE. Indeed using a binary tree build on 75% data to find the liquid, cross-validated test on the 25% remaining showed to obtain the right classification more than 99% of the time.
So this means that it is possible to classify a new trajectory to a liquid if the trajectory is close (initial velocity) to an existing one. Knowing that, I have applied the GENERIC scheme on each liquids separately.
Indeed for each liquid, I have used the UMAP reduction method to build for each liquid an embedded space in $2D$ where to apply the GENERIC scheme. For each liquid, I have selected only 3 (out of 4) trajectories to build the embedded space. To build the UMAP manifold, the connectivity parameter is 10 and the number of neighbors is 15.
All those embedded space are represented in the figures 22, 23 and 24. For every liquid, the trajectories are separated in the embedded space.

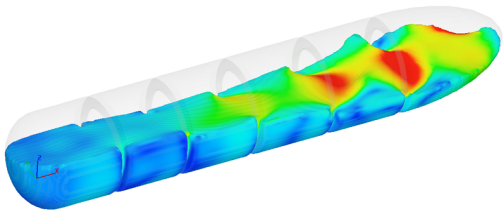
Liquids	$v_0(m/s)$	$\max(e_2^r)$	$\max(E_\infty)$	$\max(E_\infty^r)$	$mean(e_2^r)$	$mean(E_\infty)$	$mean(E_\infty^r)$
Blood	0.05	$5{,}387.10^{-6}$	$9{,}92.10^{-5}$	$1{,}1681.10^{-3}$	$1{,}712.10^{-6}$	$8{,}983.10^{-6}$	$1{,}324.10^{-4}$
Blood	0.10	$7{,}202.10^{-8}$	$2{,}497.10^{-4}$	$3{,}308.10^{-3}$	$3{,}928.10^{-8}$	$3{,}501.10^{-5}$	$5{,}099.10^{-4}$
Blood	0.20	$1{,}266.10^{-7}$	$4{,}906.10^{-4}$	$6{,}486.10^{-3}$	$7{,}604.10^{-8}$	$6{,}886.10^{-5}$	$9{,}389.10^{-4}$
Butter	0.05	$3{,}495.10^{-7}$	$8{,}566.10^{-5}$	$4{,}181.10^{-3}$	$2{,}143.10^{-7}$	$3{,}478.10^{-6}$	$1{,}775.10^{-4}$
Butter	0.10	$9{,}552.10^{-8}$	$1{,}65.10^{-4}$	$7{,}896.10^{-3}$	$5{,}013.10^{-8}$	$5{,}89.10^{-6}$	$2{,}962.10^{-4}$
Butter	0.20	$4{,}672.10^{-7}$	$3{,}236.10^{-4}$	$1{,}521.10^{-2}$	$1{,}638.10^{-7}$	$1{,}106.10^{-5}$	$5{,}301.10^{-4}$
Liquids	$v_0(m/s)$	$\max(e_2^r)$	$\max(E_\infty)$	$\max(E_\infty^r)$	$mean(e_2^r)$	$mean(E_\infty)$	$mean(E_\infty^r)$
Chocolate	0.25	$2{,}623.10^{-8}$	$4{,}937.10^{-4}$	$6{,}372.10^{-3}$	$1{,}405.10^{-8}$	$1{,}832.10^{-5}$	$2{,}477.10^{-4}$
Chocolate	0.50	$2{,}5.10^{-8}$	$9{,}704.10^{-4}$	$1{,}097.10^{-2}$	$1{,}099.10^{-8}$	$7{,}243.10^{-5}$	$8{,}785.10^{-4}$
Chocolate	0.75	$1{,}177.10^{-8}$	$1{,}198.10^{-3}$	$1{,}354.10^{-2}$	$4{,}837.10^{-9}$	$1{,}797.10^{-4}$	$1{,}78.10^{-3}$
Honey	0.25	$4{,}612.10^{-9}$	$4{,}445.10^{-4}$	$6{,}332.10^{-3}$	$2{,}063.10^{-9}$	$1{,}407.10^{-5}$	$1{,}961.10^{-4}$
Honey	0.50	$2{,}944.10^{-8}$	$8{,}858.10^{-4}$	$1{,}252.10^{-2}$	$1{,}799.10^{-8}$	$2{,}93.10^{-5}$	$3{,}989.10^{-4}$
Honey	0.75	$3{,}689.10^{-9}$	$1{,}316.10^{-3}$	$1{,}839.10^{-2}$	$1{,}397.10^{-9}$	$4{,}665.10^{-5}$	$6{,}204.10^{-4}$
Liquids	$v_0(m/s)$	$\max(e_2^r)$	$\max(E_\infty)$	$\max(E_\infty^r)$	$mean(e_2^r)$	$mean(E_\infty)$	$mean(E_\infty^r)$
Mayonnaise	0.25	$5{,}52.10^{-9}$	$4{,}209.10^{-4}$	$5{,}863.10^{-3}$	$2{,}872.10^{-9}$	$3{,}71.10^{-6}$	$5{,}205.10^{-5}$
Mayonnaise	0.50	$3{,}233.10^{-8}$	$8{,}91.10^{-4}$	$1{,}177.10^{-2}$	$1{,}837.10^{-8}$	$1{,}099.10^{-5}$	$1{,}491.10^{-4}$
Mayonnaise	0.90	$6{,}461.10^{-9}$	$1{,}403.10^{-3}$	$1{,}629.10^{-2}$	$2{,}866.10^{-9}$	$2{,}583.10^{-5}$	$3{,}201.10^{-4}$
Water	0.05	$1{,}631.10^{-6}$	$7{,}987.10^{-5}$	$3{,}62.10^{-3}$	$3{,}381.10^{-7}$	$4{,}606.10^{-6}$	$2{,}462.10^{-4}$
Water	0.15	$5{,}899.10^{-8}$	$2{,}561.10^{-4}$	$9{,}849.10^{-3}$	$3{,}014.10^{-8}$	$1{,}482.10^{-5}$	$6{,}955.10^{-4}$
Water	0.20	$3{,}135.10^{-7}$	$3{,}167.10^{-4}$	$1{,}169.10^{-2}$	$9{,}487.10^{-8}$	$1{,}828.10^{-5}$	$7{,}976.10^{-4}$

Table 7: Mean and maximum of e_2^r,, E_{inf} and E_∞^r for each trajectory.

6.2 GENERIC NUMERICAL SCHEME

I have then build the GENERIC scheme on each trajectory and tested the reconstruction error of the numerical scheme starting from every initial points. The interpolation is done each time with $N = 5$ neighbors.
To evaluate the different errors of reconstruction, I have define 3 values (46), (47) and (48).

$$e_2^r = \frac{||\tilde{\Gamma}_t^{emb} - \Gamma_t^{emb}||_2}{||\Gamma_t^{emb}||_2} \tag{46}$$



e_2^r represents the relative error in the reconstruction on the embedded manifold. Γ_t^{emb} is the embedded manifold at the time t and $\tilde{\Gamma}_t^{emb}$ is its approximation using the GENERIC scheme.

$$E_\infty = |h(\tilde{\Gamma}_t) - h(\Gamma_t)| \tag{47}$$

E_∞ is the difference between the maximum height of the liquid at a time t $h(\Gamma_t)$ and the one from its approximation $h(\tilde{\Gamma}_t)$

$$E_\infty^r = \frac{|h(\tilde{\Gamma}_t) - h(\Gamma_t)|}{h(\Gamma_t)} \tag{48}$$

E_∞^r is the relative error assimilated at E_∞.
In the figure 25, 26 and 27, there are the values of e_2^r,, E_{inf} and E_∞^r and in the table 7, their mean and maximum for each trajectory.

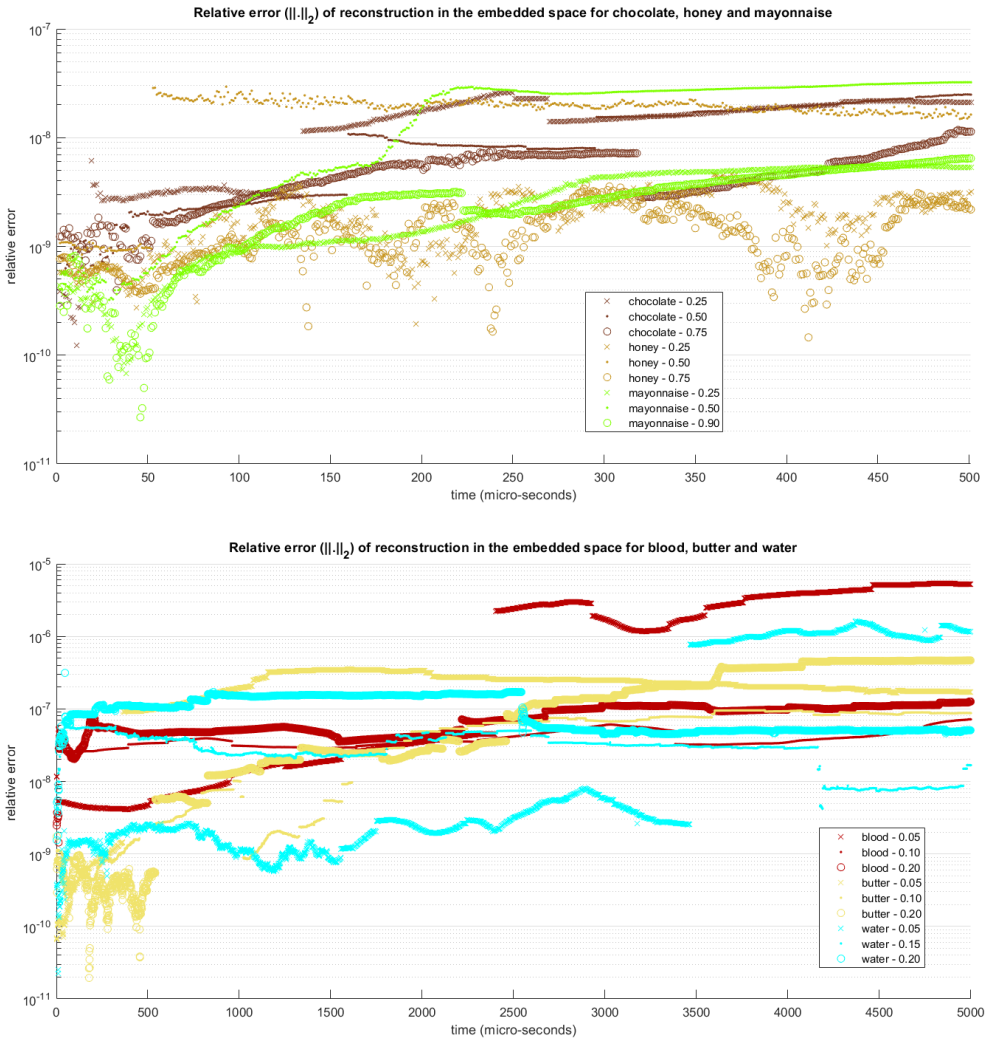


Figure 25: Values of the error e_2^r for every trajectory.

For the error of reconstruction in the embedded space, represented by e_2^r, it is really small because it is always smaller than 10^{-5} for every trajectory. So the GENERIC numerical scheme works very well to reconstruct trajectories.
E_∞ gives the information that the shape of the liquids is respected because the maximum height is well respected. Indeed the height different is always smaller than 2mm and with the relative error E_∞^r, it is always smaller than 2% which is a really low relative error.

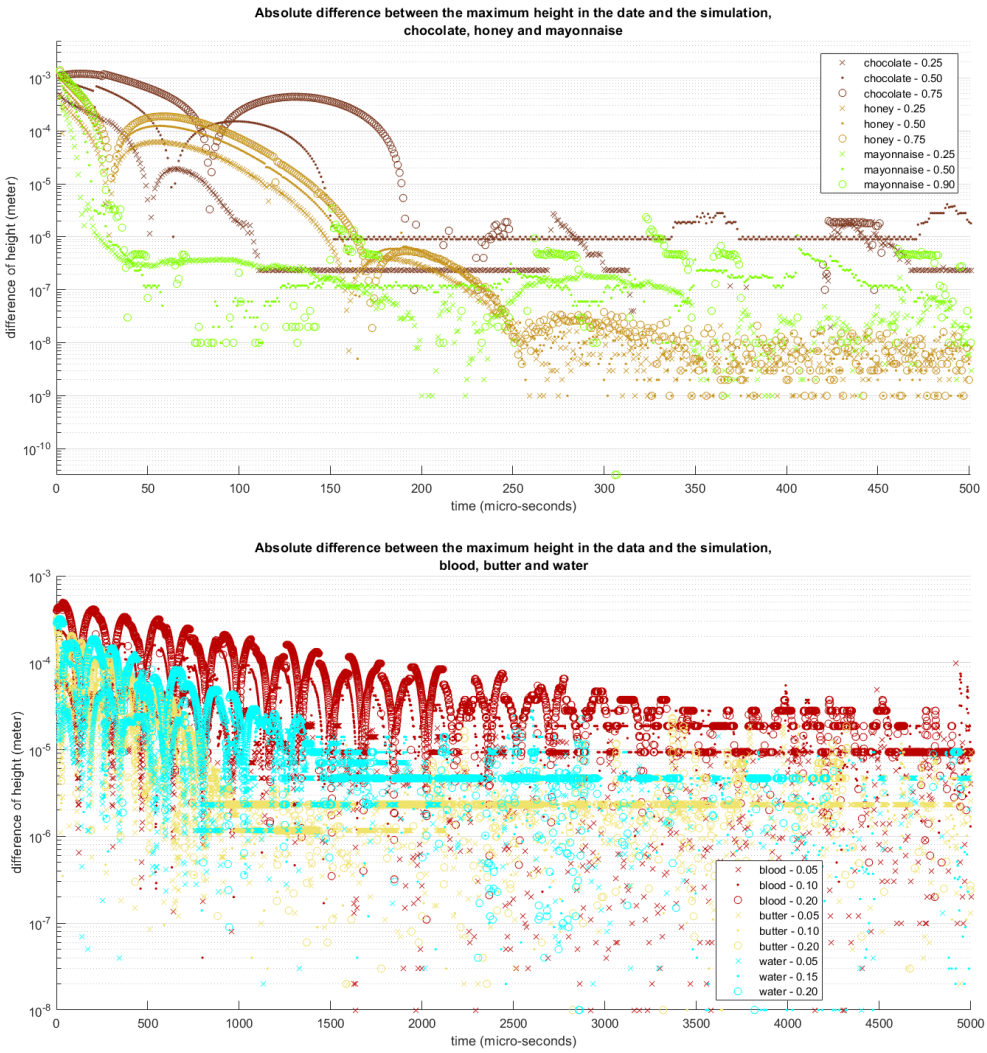
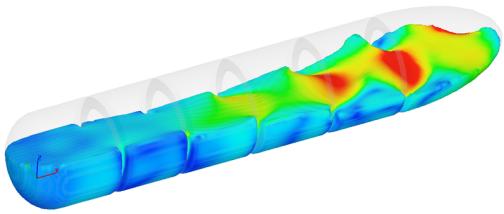


Figure 26: Values of the error E_∞ for every trajectory.

However at the current time, the GENERIC scheme is not able to build correctly a new trajectory from scratch. Basically, because of the morphology of the embedded space and the interpolating method, starting from a new point the new trajectory will either stay really located around the initial point because it is too far from the other trajectories or it will completely follow a trajectory that already exists.
The trajectories being separated in the embedded space, it is difficult to have an interpolating method



that will really use the information of each trajectory to build a new one.

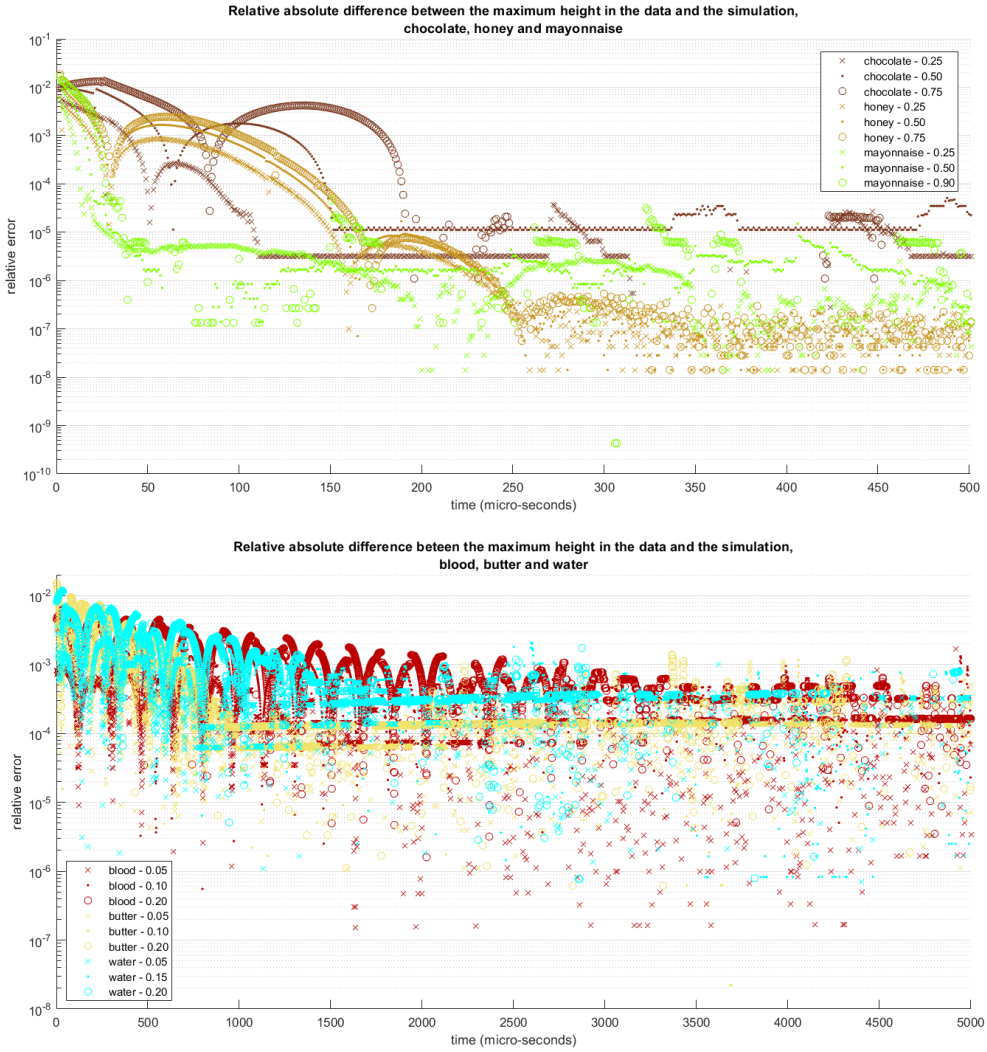
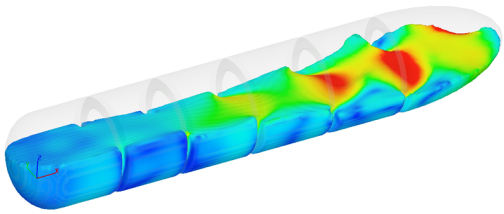


Figure 27: *Values of the error E^r_∞ for every trajectory.*

7 Conclusions

Simulating liquids on Abaqus with the ALE-mesh is very difficult and can still show problems with the incompressibility of the liquids. However if this problem does not occur, simulations of the liquids show a realistic behavior. In that sense, simulating sloshing behavior is possible and realistic for very viscous liquids using the Abaqus/Explicit toolbox with the ALE-mesh. A way to make sure of the incompressibility of the liquid seems to be playing with the bulk viscosity parameter.
The new reduction-method UMAP shows incredible results to resume a very high and complex data-set in a 2D space. Which allows visual interpretation of liquids behavior. UMAP also has the principle to develop a topological space associated to the initial space. One way to improve even more the embedded space would be to give the same order of magnitude to every physical parameter. For example the internal energy does not have the same order of magnitude than the velocity which implies that in the building of the topological space, that uses at some point the Euclidean distance, the variations in the values of internal energy are less important than the one of velocity. So adding a scaling parameter to each physical parameter could help improving the distinction of different behaviors.
The GENERIC numerical scheme showed great results when applied to 2D trajectories. The reconstruction of the trajectories was very accurate, showing really small relative error in the reconstruction. However, the way to interpolate does not work with building new trajectories because of the shape of the trajectories in the embedded space. One way to be able to create new trajectories could be to improve the UMAP algorithm as previously said and to add more data coming from other velocities. In a way, to discretize more the data on the initial velocity. The aim of the work in progress is also to be able to run the GENERIC numerical scheme in real-time, which is not true right now: it takes for example 100 seconds to get 5 seconds of blood behavior. But this time can be improved by computing a better way to find the neighbors in the embedded space. The function that I personally used came from the UMAP package but the research could be done faster knowing the shape of the data. For example, I think that giving a tree architecture to the research of neighbors could really improve the time spent to locate the closest neighbors.
Another way to build the manifolds could be to use the statistical analysis done in the second section, and so, to simulate liquids with density and viscosity according to the distributions given. This way liquids are identified by their characteristics.



References

[1] Beatriz Moya et al. "Learning slosh dynamics by means of data". In: *Computational Mechanics* (2019).

[2] M. Grmela and H.C Oettinger. "General Equation for Non-Equilibrium Reversible-Irreversible Coupling, GENERIC". In: *Phys. Rev. E. 56* 6 (1997), pp. 6620–6632.

[3] J.J. Monaghan. "An introduction to SPH". In: *Computer Physics Communications* 48 (1988), pp. 89–96.

[4] Erwin Stein, René de Borst, and Thomas J.R. Hughes. *Encyclopedia of computational mechanics*. John Wiler Sons, 2004.

[5] Leland McInnes, John Healy, and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: (2018).

[6] U. Ruth Charrondiere, David Haytowitz, and Barbare Stadlmayr. *FAO/INFOODS Density Database*. URL: http://www.fao.org/3/ap815e/ap815e.pdf.

[7] Michael Smith. *Approximate viscosities of some common liquids*. URL: https://www.michael-smith-engineers.co.uk/mse/uploads/resources/useful-info/General-Info/Viscosities-of-Common-Liquids.pdf.

[8] Charles Zaiontz. *Kolmogorov-Smirnov Test for Normality*. URL: http://www.real-statistics.com/tests-normality-and-symmetry/statistical-tests-normality-symmetry/kolmogorov-smirnov-test/.

[9] Rey-Yie Fong. "A Comparison Study between ALE and SPH for Yacht Structure Design under Slamming Impact Loads". In: Aug. 2011.

[10] Geneviève Toussaint, Amal Bouamoul, and DRDC Valcartier. "Comparison of ALE and SPH methods for simulating mine blast effects on structures". In: *Defence RD Canada - Valcartier* (2010). Technical report.

[11] Roberts et al. *Heat and thermodynamics (Vol. 4)*. Interscience Publishers, 1954.

[12] Glenn Elert. *The Physics Hypertextbook*. URL: https://physics.info/viscosity/.

[13] SVM. *Viscosity of flower honey (blended)*. URL: https://wiki.anton-paar.com/en/flower-honey-blended/.

[14] ABAQUS Inc. *Water sloshing in a pitching tank*. URL: http://130.149.89.49:2080/v6.9/books/bmk/default.htm?startat=ch01s12ach89.html.

[15] Herschel et al. "Konsistenzmessungen von Gummi-Benzollösungen". In: *Kolloid Zeitschrift* 39 (1926). DOI: 10.1007/BF01432034, pp. 291–300.

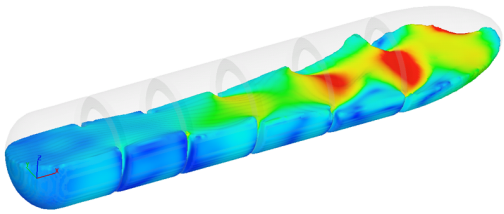
[16] P.Bottiglieri et al. "Rheological characterization of ketchup". In: *Journal of food quality* 14 (1991). DOI: 10.1111/j.1745-4557.1991.tb00089.x, pp. 497–512.

[17] Shewaferaw S. Shibeshi and William E. Collins. "The rheology of blood flow in a branched arterial system". In: *National Institute of Health* (2006).

[18] Dayane Izidoro et al. "Sensory evualation and rheological behavior of commercial mayonnaise". In: *International Journal of Food Engineering* 3 (2007). DOI: 10.2202/1556-3758.1094.

[19] Vojtech Kumbar et al. "Rheological behaviour of chocolate at different temperatures". In: *Slovak journal of food sciences* 12 (21 March 2018). DOI: 10.5219/876, pp. 123–128.

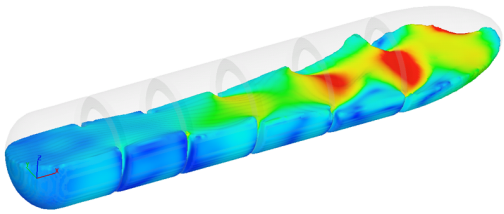
[20] Abaqus Analysis User's Manual. *ALE adaptative meshing*. URL: https://www.sharcnet.ca/Software/Abaqus610/Documentation/docs/v6.10/books/usb/default.htm?startat=pt04ch12s02aus81.html.



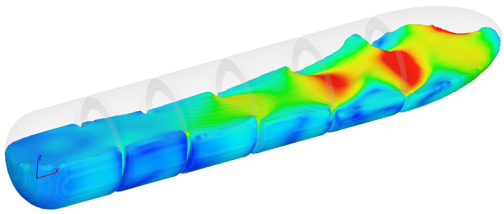
[21] Abaqus Analysis User's Manual. *Hyperelastic behavior of rubberlike materials*. URL: https://classes.engineering.wustl.edu/2009/spring/mase5513/abaqus/docs/v6.6/books/usb/default.htm?startat=pt05ch17s05abm07.html.

[22] David Gonzalez, Francisco Chinesta, and Elias Cueto. "Learning corrections for hyperelastic models from data". In: ().

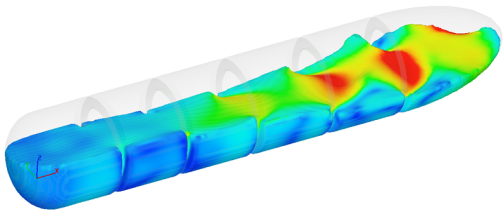
[23] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.



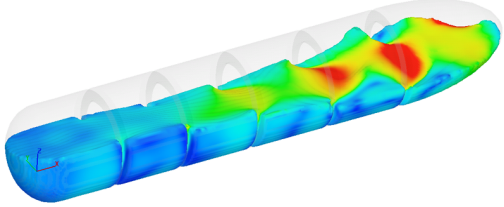
A Data-set of the common liquids



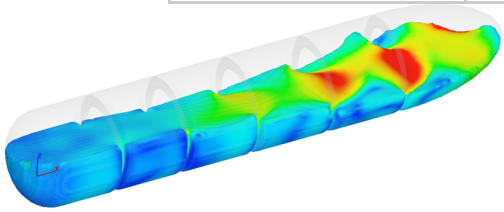
	absolute viscosity			Temperature		Viscosity type	density
	mPA.s	Pa.s	log(Pa.s)	°C	K		kg/m3
Acetate glue	1300	1,3	1,1E-01	20	293	T	1190
Baby food	1400	1,4	1,5E-01	93	366	T	
Batter	29500	29,5	1,5E+00	30	303	T	550
Beet sauce	1950	1,95	2,9E-01	76	349	T	
Biscuit cream premix	29200	29,2	1,5E+00	18	291	T	
Blood	3500	3,5	5,4E-01	37	310	T	1060
Bone oil	48	0,048	-1,3E+00	54	327	N	
Brewers Yeast	368	0,368	-4,3E-01	18	291	T	1100
Broth mix	430	0,43	-3,7E-01	18	291	T	
Butter fat	42	0,042	-1,4E+00	43	316	N	911
Butter fat	20	0,02	-1,7E+00	65	338	N	911
Butter deodorised	45	0,045	-1,3E+00	50	323	N	911
Carob bean sauce	1500	1,5	1,8E-01	30	303	T	
Castor oil	580	0,58	-2,4E-01	27	300	N	961
Castor oil	36	0,036	-1,4E+00	80	353	N	961
Chinawood oil	300	0,3	-5,2E-01	21	294	N	937
Chocolate	280	0,28	-5,5E-01	49	322	T	1325
Citrus fruit pulp	600	0,6	-2,2E-01	20	293	T	
Coconut oil	55	0,055	-1,3E+00	24	297	N	924
Coconut oil	30	0,03	-1,5E+00	38	311	N	924
Cod oil	32	0,032	-1,5E+00	38	311	N	930
Coffee liquor 30-40%	50	0,05	-1,3E+00	20	293	T	1130
Corn oil	28	0,028	-1,6E+00	57	330	N	922
Cottage cheese	30000	30	1,5E+00	18	291	T	1040
Cotton seed oil	62	0,062	-1,2E+00	24	297	N	920
Cotton seed oil	24	0,024	-1,6E+00	52	325	N	920
Cocoa butter	50	0,05	-1,3E+00	60	333	N	919
Cocoa butter	0,5	0,0005	-3,3E+00	100	373	N	919
Condensed milk	60	0,06	-1,2E+00	45	318	N	1293
Condensed milk 75% solid	2160	2,16	3,3E-01	20	293	T	1293



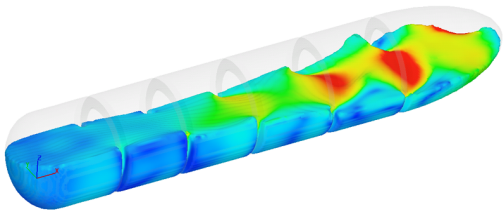
Cream 30% fat	14	0,014	-1,9E+00	16	289	N	1012
Cream 40% fat	48	0,048	-1,3E+00	16	289	N	1000
Cream 50% fat	112	0,112	-9,5E-01	16	289	N	994
Cream 50% fat	55	0,055	-1,3E+00	32	305	N	978
Cresol crystals	10	0,01	-2,0E+00	18	291	T	1020
Custard	1500	1,5	1,8E-01	85	358	T	1070
Detergent	1470	1,47	1,7E-01	70	343	T	1400
Diethylene	32	0,032	-1,5E+00	21	294	N	1120
Edible oil	65	0,065	-1,2E+00	20	293	N	920
Ethylene	18	0,018	-1,7E+00	21	294	N	1180
Gelatine 37% solid	1190	1,19	7,6E-02	43	316	T	1270
Glucose	6000	6	7,8E-01	25	298	T	1560
Glycerine 100%	648	0,648	-1,9E-01	20	293	N	1260
Glycerine 100%	176	0,176	-7,5E-01	38	311	N	1260
Gravy slurry	110	0,11	-9,6E-01	80	353	T	
Hand cream	780	0,78	-1,1E-01	18	291	T	900
Honey	14095	14,095	1,1E+00	20	293	N	1420
Fruit juice	65	0,065	-1,2E+00	18	291	N	1060
Isopropyl alcohol	1,9	0,0019	-2,7E+00	85	358	N	786
Jam garnish	8440	8,44	9,3E-01	16	289	T	
Lacquer 25% solids	3000	3	4,8E-01	18	291	T	837
Lard	62	0,062	-1,2E+00	38	311	N	919
Lard oil	45	0,045	-1,3E+00	38	311	N	917
Latex emulsion	200	0,2	-7,0E-01	24	297	T	
Latex emulsion	48	0,048	-1,3E+00	65	338	T	
Linseed oil raw	29	0,029	-1,5E+00	38	311	N	930
Malt extract 80%	9500	9,5	9,8E-01	18	291	T	336
Malt extract	3000	3	4,8E-01	60	333	T	336
Mayonnaise	20000	20	1,3E+00	20	293	T	910
Milk	2	0,002	-2,7E+00	18	291	N	1000
Milk	1	0,001	-3,0E+00	52	325	N	1000
Milk whey 48% sugar	1100	1,1	4,1E-02	40	313	T	1000
Mincemeat	100000	100	2,0E+00	30	303	T	1020
Mousse mix	1200	1,2	7,9E-02	5	278	T	



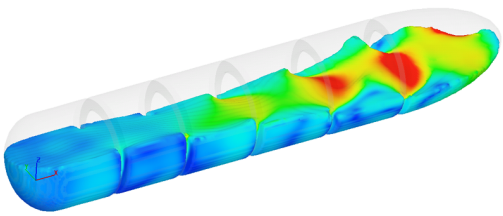
NaOH 20%	1	0,001	-3,0E+00	18	291	N	2130
NaOH 30%	1	0,001	-3,0E+00	18	291	N	2130
NaOH 40%	20	0,02	-1,7E+00	18	291	N	2130
Olive oil	40	0,04	-1,4E+00	38	311	N	918
Orange juice 30 brix	630	0,63	-2,0E-01	20	293	N	1014
Orange juice 30 brix	91	0,091	-1,0E+00	80	353	N	1014
Orange juice 50 brix	2410	2,41	3,8E-01	20	293	N	1014
Orange juice 50 brix	330	0,33	-4,8E-01	80	353	N	1014
Palm oil	43	0,043	-1,4E+00	38	311	N	915
Parafin emulsion	3000	3	4,8E-01	18	291	T	800
Peanut oil	38	0,038	-1,4E+00	38	311	N	910
Pectin	300	0,3	-5,2E-01	38	311	N	700
Pectin	345	0,345	-4,6E-01	27	300	N	700
Polyester	3000	3	4,8E-01	30	303	T	1390
Polypropylene	240000	240	2,4E+00	50	323	T	946
Polyisobutylene	12500	12,5	1,1E+00	85	358	T	920
Plastisol	28000	28	1,4E+00	18	291	T	
Printer ink	1000	1	0,0E+00	38	311	T	1064
Printer ink	400	0,4	-4,0E-01	54	327	T	1058
Process cheese	6500	6,5	8,1E-01	80	353	T	1100
Process cheese	30000	30	1,5E+00	18	291	T	1100
Propylene	52	0,052	-1,3E+00	21	294	N	1740
Resin solution	880	0,88	-5,6E-02	24	297	T	1000
Resin solution	975	0,975	-1,1E-02	21	294	T	1000
Resin solution	7140	7,14	8,5E-01	18	291	T	1000
Rice pudding	10000	10	1,0E+00	100	373	T	1069
Salad cream	2000	2	3,0E-01	18	291	T	994
Sauce apple	500	0,5	-3,0E-01	80	353	T	
Shampoos	3000	3	4,8E-01	36	309	T	960
Sperm oil	24	0,024	-1,6E+00	38	311	N	
Soap arylan	3000	3	4,8E-01	36	309	T	932
Soap solution	630	0,63	-2,0E-01	60	333	T	932
Sorbitol	200	0,2	-7,0E-01	20	293	N	1490
Soya bean oil	60	0,06	-1,2E+00	24	297	N	927



Soya bean oil	12	0,012	-1,9E+00	80	353	N	927
Soya bean slurry	10000	10	1,0E+00	90	363	T	
Sulphonic acid	125	0,125	-9,0E-01	30	303	T	
Tomato ketchup	1000	1	0,0E+00	30	303	T	1149
Tomato paste 30%	195	0,195	-7,1E-01	18	291	T	1040
Toothpaste	85000	85	1,9E+00	18	291	T	1330
Tricetate dope	50000	50	1,7E+00	40	313	T	1290
Triethylene	40	0,04	-1,4E+00	21	294	N	1100
Turpentine	2	0,002	-2,7E+00	16	289	N	865
Vinegar	15	0,015	-1,8E+00	20	293	N	1050
Water	1,3	0,0013	-2,9E+00	18	291	N	983
Wax	500	0,5	-3,0E-01	93	366	T	900
Whale oil	30	0,03	-1,5E+00	38	311	N	
Whole egg	150	0,15	-8,2E-01	4,5	277,5	T	1031
Yeast surry	20	0,02	-1,7E+00	18	291	T	
Yoghurt	152	0,152	-8,2E-01	40	313	T	1060



B VUVISCOSITY SUBROUTINE



```
      subroutine vuviscosity (
C Read only -
     *     nblock,
     *     jElem, kIntPt, kLayer, kSecPt,
     *     stepTime, totalTime, dt, cmname,
     *     nstatev, nfieldv, nprops,
     *     props, tempOld, tempNew, fieldOld, fieldNew,
     *     stateOld,
     *     shrRate,
C Write only -
     *     viscosity,
     *     stateNew )
C
C
      dimension props(nprops),
     *  tempOld(nblock),
     *  fieldOld(nblock,nfieldv),
     *  stateOld(nblock,nstatev),
     *  shrRate(nblock),
     *  tempNew(nblock),
     *  fieldNew(nblock,nfieldv),
     *  viscosity(nblock),
     *  stateNew(nblock,nstatev)
C
      character*80 cmname
C
              parameter ( one = 1.d0 )
C
C     Cross viscosity
C
      eta     = props(1)
      rn      = props(2)
      tau     = props(3)
C
      do k = 1, nblock
         if ( shrRate(k) .eq. 0 ) then
            viscosity(k) = tau
         else
            viscosity(k) = tau / shrRate(k) +
     *       eta * (shrRate(k))**(rn-1)
         end if
      end do
C     print*,'work',viscosity(1)
C     write(*,*) 'work ', viscosity(1)
C
      return
      end
```