Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

Implementation and validation of a Herschel-Bulkley PFEM model in Kratos Multiphysics

MASTER FINAL THESIS

Final Thesis developed by:
Tomas Pozo, Timur

Directed by:
Franci, Alessandro
De Pouplana Sardà, Ignasi

Master in:
Civil Engineering

Barcelona, February 2022

Department of DECA – Department of Civil and Environmental Engineering

Acknowledgments

I would like to thank my tutors D. Alessandro Franci and D. Ignasi de Pouplana, for their guidance and patience throughout these long months in the preparation of this work.

I would like to thank my friends, my fellow students and co-workers, because they have always given me the possibility to see beyond my vision and to know other points of view.

Finally, and most especially, to my family, because without them I would not be who I am and I would not have reached where I have reached.

Abstract

The objective of this master's thesis is the implementation and validation, by means of literature examples, of the Herschel-Bulkley constitutive law in fluid dynamics in a particle finite element method (PFEM).

The basis of the work is the PFEM formulation for Bingham fluids implemented in the open-source platform Kratos Multiphysics.

The Herschel-Bulkley model relates the shear stress tensor to the strain rate tensor taking into account the yield stress, which limits the beginning or the end of the fluid motion; the fluid viscosity, which is responsible for the fluid's resistance to motion; and the fluid index, a parameter representing the level of nonlinearity of the motion.

The Herschel-Bulkley law is validated with benchmark problems from the literature. Convergence in space and time, sensitivity analysis of the model variables and comparisons with the standard Bingham model are carried out.

In addition, the so-called Papanastasiou regularization is applied to avoid numerical drawbacks arising from the Herschel-Bulkley model.

All results are plotted throughout the paper and compared to both numerical and experimental references found in scientific articles.

Throughout the numerical calculations it is possible to calibrate a series of fundamental parameters for simulations with the model.

Resumen

El objetivo de este trabajo de final de máster es la implementación y validación, mediante ejemplos en la literatura, de la ley constitutiva de Herschel-Bulkley de la dinámica de fluidos mediante el método de elementos finitos de partículas (PFEM).

La base del trabajo es la formulación PFEM para fluidos Bingham implementada en la plataforma de código abierto Kratos Multiphysics.

El modelo de Herschel-Bulkley relaciona el tensor de tensiones cortante con el tensor de velocidad de deformación teniendo en cuenta el límite elástico, que limita el inicio o el final del movimiento del fluido; la viscosidad del fluido, que es responsable de la resistencia del fluido al movimiento; y el índice del fluido, un parámetro que representa el nivel de no linealidad del movimiento.

La ley de Herschel-Bulkley se valida con problemas de referencia de la literatura. Se lleva a cabo la convergencia en el espacio y el tiempo, el análisis de sensibilidad de las variables del modelo y las comparaciones con el modelo estándar de Bingham.

Además, se aplica la llamada regularización de Papanastasiou para evitar los inconvenientes numéricos del modelo de Herschel-Bulkley.

Todos los resultados se grafican a lo largo del trabajo y se comparan con referencias numéricas y experimentales encontradas en artículos científicos.

A lo largo de los cálculos numéricos es posible calibrar una serie de parámetros fundamentales para las simulaciones con el modelo.

0. Contents

List of Figures

List of Tables

1. Introduction

1.1. Objectives

The main purpose of the work consists in the implementation and validation of a Herschel-Bulkley constitutive law with Papanastasiou regulating parameter. The regularization of the Herschel-Bulkley law is required in order to circumvent the numerical drawbacks arising from the piecewise function of the model and its singularity. For this purpose, this model is solved numerically by means of the Particle Finite Element Method, a method that combines the fundamentals of particle methods and the Finite Element Method (FEM).

The model is implemented in the Kratos Multiphysics program (from here on Kratos) that is a framework for building parallel multi-disciplinary simulation software. Kratos has BSD license and is written in C++ with extensive Python interface.

The implementation of the law is validated and contrasted with examples found in the literature. The validation has multiple parts where convergence studies are highlighted by varying the mesh size among others. Through this, the sensibility of the model to crucial parameters is studied. Particular attention is devoted to study the effect of the Papanastasiou regulation parameter.

1.2. Motivation

Laboratory testing on non-Newtonian fluids is a complex task. It is particularly complex to recreate real-world situations on a reduced and control scale (called reduced physical model). Numerical methods, after being calibrated and validated, offer the possibility of perform varied analyses for different fluid characteristics and on different scales and geometries in a quite straightforward fashion. Nevertheless, in order to consider the numerical results reliable, it is necessary a preliminary phase of validation of the numerical tool where simple experimental tests are used as the reference.

The Finite Element Method is one of the most known numerical methods. In fluid mechanics, the FEM is usually employed using an Eulerian description of the motion and mesh. An Eulerian mesh is fixed and this is useful for large deformations analysis (that are typical in fluid dynamics) because it does not get distorted during the fluid motion. However, this description is linked to convective terms in the development of the time derivatives, which need a special treatment to avoid numerical drawbacks, such as instability of the numerical results. Furthermore, Eulerian FEM requires ad hoc strategy to capture the fluid free surface. The PFEM uses a Lagrangian description of the medium, usually used for solid mechanics due to its low deformations. This can be done thanks to the use of a continuous remeshing algorithm This has the great advantage of getting rid of the convective derivative and equating the material derivative with the spatial derivative- Furthermore, in such a Lagrangian method, the fluid free surface is automatically detected by the mesh nodes position. For these reasons, the PFEM has been used here for modelling the non-Newtonian fluids

The use of numerical simulation has significant advantages with respect to laboratory tests Just knowing the necessary material parameters and geometry, a good numerical simulation can reach accurate results. With this, someone can reduce the use of physical laboratory tests and use them to contrast the numerical simulation results under specific situations and geometries.

Figure 1.1 shows the scheme that computational engineering follows when facing a problem like the present one (like any other). A more detailed outline of the computational engineering approach to the problems posed to it is presented in Annex C.

The innovation of this work is not the treatment of the fluid flow from a Lagrangian description approach (that has already been done successfully before), but the implementation and careful validation of a Herschel-Bulkley law in this numerical framework.

The fact of considering a non-linearity in the rheological law of the fluids increase the complexity of the numerical model.

The Herschel-Bulkley law represent a general framework of non-Newtonian fluid law. For example, the Bingham law can be view as a particular case of the Herschel-Bulkley model.

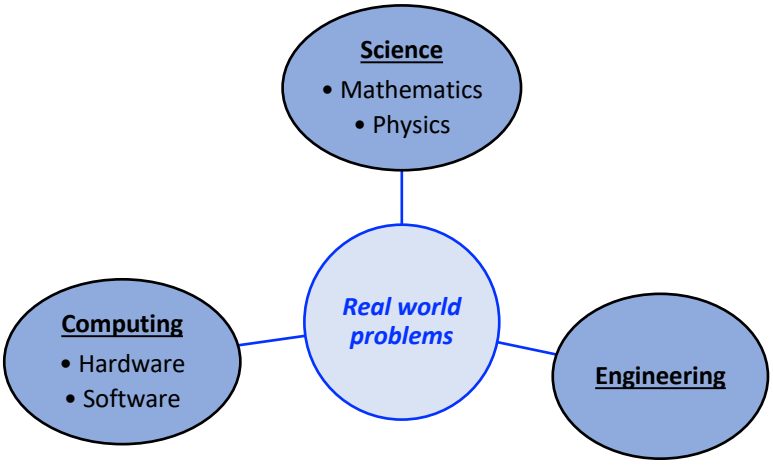


Figure 1.1: Computer engineering scheme

1.3. State of the art

This part presents the existing advances on the subject in question.

Throughout the paper, different approaches to the description of fluid dynamics are presented. It is not only the description that differs but also the way of solving it by numerical methods.

Due to the nonlinear relationship between stresses and kinematic quantities, the numerical simulation of non-Newtonian fluid mechanics is challenging and still in active development. There are different numerical procedures in the description of non-Newtonian materials. The most known methods are the finite difference method and finite element method.

In the Table 1.1 are examples of the application of different numerical methodologies for the resolution of viscoplastic fluids type:

Numerical method	Examples
Finite Elements	[1] & [2]
Finite Volumes	[3]
Boundary Elements	[4]

Table 1.1: Examples of application of numerical methods

In [5], a detailed description of advantages and disadvantages of the use of differences and finite elements as applied to fluids:

1) Finite difference techniques are relatively easy to understand and to implement for a newcomer to the field, while the development of a finite element code requires a non-negligible amount of programming.

2) The finite difference method will usually be cheaper on the computer than the finite element method.

3) The finite element method has a tremendous advantage over finite differences for solving flows in a complex geometry, which, usually, cannot even be approached with the latter.

In spite of the different numerical resolution methodologies applied, there is also, in each of these methods, a different approach to the description of the motion. Many of the models used in the literature characterize the motion with the Eulerian description (spatial description). The physical properties are described in terms of the spatial coordinates and time. In a Eulerian description, the focus is on what is occurring at a fixed point in space (a spatial point labeled by its spatial coordinates) as time progresses. It is used, normally, in fluid mechanics [6]. However, the method presented here presents a totally different approach and uses the Lagrangian description of the motion (material description). The physical properties are described in terms of the material coordinates and time. It focusses on what is occurring at a moving material point (a particle, labeled by its material coordinates) as time progresses [6].

It is not only the numerical method and the description of the motion, but also the constitutive law model that is used for the simulation. The Bingham model (a particular case of Herschel-Bulkley) is one of the most studied and used non-Newtonian laws. The model has been used in the application of engineering numerical simulations of many types, such as, landslides, snow avalanches, mud flows and debris flows [7]. All of the above cases have one thing in common and that is the fact that they comply with the free surface flow condition. A free surface is understood to be that surface of a fluid that is subjected to a zero parallel shear tensor field.

The PFEM has been applied to numerous examples of physical problems and has been able to solve them correctly with great efficiency. It has been applied to geotechnics [10] , fluid-structure interaction [11], and free surface fluid dynamics flows [12], among others. Debris flows have been investigated also using a Herschel Bulkley model in [8]. This article deals with the model using the Smooth Particle Hydrodynamics (SPH) method, which is a mesh-free particle based numerical method (using a Lagrangian motion description).

An extra addition to the numerical calculation of viscoplastic fluids as presented in this work is the evaluation of the variables that characterize them. The parameters of the Herschel Bulkley model are difficult to calculate. One of the ways to do it is by the method proposed and developed by [9] that consists in measuring the pressure gradients while the fluid flows at different apparent velocities in a circular pipe using a combination of physics-based equations and nonlinear optimization.

Finally, it should be noted that the fact that the fluid boundaries are constantly evolving makes the simulation of these very complex. When using the Eulerian description, the models must be completed with specific techniques capable of following the free surface of the fluid in each step of the simulation itself (for instance, Level Set Method [13] and Volume Of Fluid [14]). However, in the Lagrangian description, where the position of the particles (the nodes of the discretization mesh) is constantly being updated, the free surface of the fluid is automatically located.

1.4. Methodology

After a review of non-Newtonian fluids and their multiple numerical and variant methods of resolution, the methodology used is explained.

The fluid dynamics is described with the constitutive law of the model proposed by Herschel-Bulkley. In order to circumvent the numerical drawbacks arising from the piecewise function of the model and its singularity, the model is regularized with the Papanastasiou parameter.

Once it is understood, it is implemented in the Kratos Multiphysics program, which has a user-friendly graphical interface in the GiD software, a universal and adaptive and pre and post processor for numerical simulations in science and engineering [15]). The model was implemented using the C++ programming language.

Prior to model testing, several checks of the parameters involved in the numerical calculations must be performed. This is achieved by convergence analysis of these parameters and their calibration.

As a previous step to the validation of the implemented model, the Bingham model is validated in order to be able to perform a behavioral analysis of the current state of the software. Once this is done, the Herschel-Bulkley validation must be performed. For this, several examples must be simulated contrasting the results with the existing literature.

1.5. Structure

The structure followed consists of the following path through the sections of the report:

- First, in the Introduction (Chapter 1), the motivation is detailed and the objectives present in this final work are extended.
- Then, in the State of the Art (Chapter 2), the current state of the art is summarized in terms of the model to be used and the numerical methods employed.
- Then, a detailed description of the PFEM (Chapter 3) program is given. For a better understanding of the method, a summary of numerical methods is attached (Annex C). For a better understanding of the model to be simulated, Annex B (and A as language of B) is added.
- Once the method is implemented, it is time to contrast its operation with examples. This is developed in Chapter 4 (Validation).
- In the Conclusions and future lines of research, a summary of the results is provided and future work to be carried out such as the coupling of the thermo-mechanical analysis using PFEM is proposed.
- In Annex D, it is briefly explained what the program used consists of, as a basic tool for solving the mathematics.

2. PFEM. Theory & formulation

2.1. Non-Newtonian fluids

The continuum, as can be seen in Figure 2.1., is classified into very large sets according to its mechanical characteristics. Much of the theory of fluid mechanics focuses on Newtonian fluids because of their peculiarity in their rheology, a linear relationship between strain rates and shear stresses. However, not all fluids can be modeled by this simple constitutive law.

That is why solid alternatives to that theory must be studied. Many types of constitutive laws arise and all of them can be applied in different fields depending on the characteristics of the fluid. In the case of this work, the focus is on non-Newtonian viscoplastic time-independent fluids.

These types of fluids are constantly present on a daily basis. Examples are shower gel and blood itself. An example applied to civil engineering is the fresh cement. Examples in nature are lava and mud.

To model such fluids, one must apply Newton's second law to obtain the linear momentum balance.

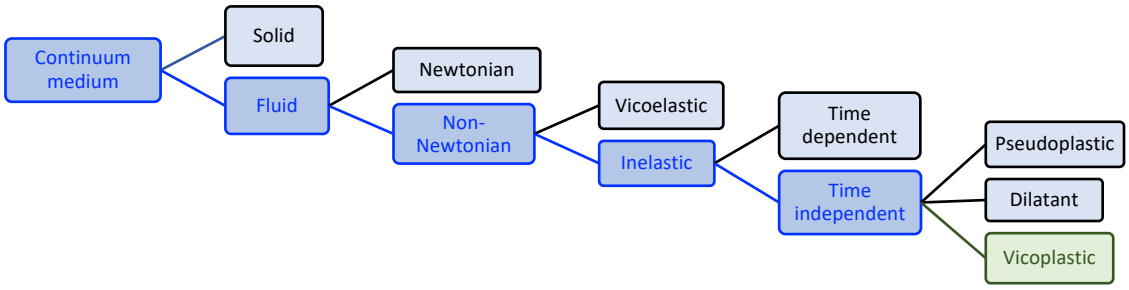


Figure 2.1: General classification of fluids

The Cauchy stress tensor is usually expressed in fluid mechanics by (3.1), where "p" is the pressure (there are several definitions, listed in Annex B) and "$\underline{\underline{\tau}}$" is the viscous stress tensor. Note that the first equation is written in a tensorial way, while the second one is provided with indicial notation. This is done also for the following equations.

$$\begin{cases} \underline{\underline{\sigma}} = -p\,\underline{\underline{\delta}} + \underline{\underline{\tau}} \\ \sigma_{ij} = -p\,\delta_{ij} + \tau_{ij} \end{cases} \tag{3.1}$$

The viscosity stress tensor is formulated for both compressible (3.2) and incompressible (3.3) media.

$$\begin{cases} \underline{\underline{\tau}} = \kappa\left(\underline{\underline{d}} : \underline{\underline{\delta}}\right)\underline{\underline{\delta}} + 2\,\mu\,\underline{\underline{d'}} \\ \tau_{ij} = \kappa\,d_{kk}\,\delta_{ij} + 2\,\mu\,d_{ij}{}' \end{cases} \tag{3.2}$$

$$\begin{cases} \underline{\underline{\tau}} = 2\,\mu\,\underline{\underline{d}} = 2\,\mu\,\underline{\underline{d'}} \\ \tau_{ij} = 2\,\mu\,d_{ij} = 2\,\mu\,d_{ij}{}' \end{cases} \tag{3.3}$$

Where "σ_{ij}" is the Cauchy stress tensor measured in [*Pa*], "p" is the thermodynamic pressure measured in [*Pa*], "κ" is the bulk viscosity measured in [*Pa·s*], "d_{ij}" is the strain rate tensor measured in [*1/s*], "δ_{ij}" is the Kronecker's delta, "μ" is the dynamic viscosity measured in [*Pa·s*].

It is highlighted that the dynamic viscosity in the Herschel Bulkley model is represented by the so-called consistency modulus *k*. For further clarification and to avoid misinterpretation in the notation, the Table 2.1 shows the different meaning of indices represented with a similar letter.

Figure 2.2 shows the formulation used in the model implemented in the simulations.

$$\left.\begin{array}{l} \dot{p} + \kappa\left(\underline{\nabla} \cdot \underline{v}\right) = 0 \\ \underline{\nabla} \cdot \underline{\sigma} + \rho\, \underline{b} = \rho\, \underline{\dot{v}} \end{array}\right\} \rightarrow \textit{Mass Conservation + Linear Momentum Balance}$$

$$\uparrow \quad \textit{Stress tensor decomposition}$$

$$\underline{\sigma} = -p\, \underline{\delta} + \underline{\tau}$$

$$\uparrow \quad \textit{Shear stress description}$$

$$\underline{\tau} = 2\, \mu(\dot{\gamma})\, \underline{d}$$

$$\uparrow \quad \textit{Herschel-Bulkley-Papanastasiou proposal}$$

$$\mu(\dot{\gamma}) = \frac{\tau_o}{\dot{\gamma}}\left(1 - e^{-m\,\dot{\gamma}}\right) + k\, \dot{\gamma}^{n-1}$$

$$\uparrow \quad \textit{Herschel-Bulkley model}$$

$$\tau = \tau_o + k\, \dot{\gamma}^n$$

Figure 2.2: Outline of the formulation to be followed

$Bulk\ Modulus$	$\rightarrow$	κ
$Bulk\ Viscosity$	$\rightarrow$	$\mathcal{K}$
$Consistency\ Modulus = Dynamic\ viscosity$	$\rightarrow$	k

Table 2.1: Distinction of the different K's

A general way to express the general dynamic behavior of any fluid is:

$$\mu = f\left(\underline{d}, \rho, \theta\right) \tag{3.4}$$

Where "$\underline{d}$" is the strain rate tensor, "ρ" is the fluid density and "θ" is the fluid temperature.

Some of non-Newtonian fluids require minimal shear stress "τ_o" to start moving or deforming (to keep the material flowing). This minimum stress is called *yield shear* stress.

As it can be seen in Table 2.2, there exits some different theorical fluid behaviors, also illustrated in the Figure 2.3.

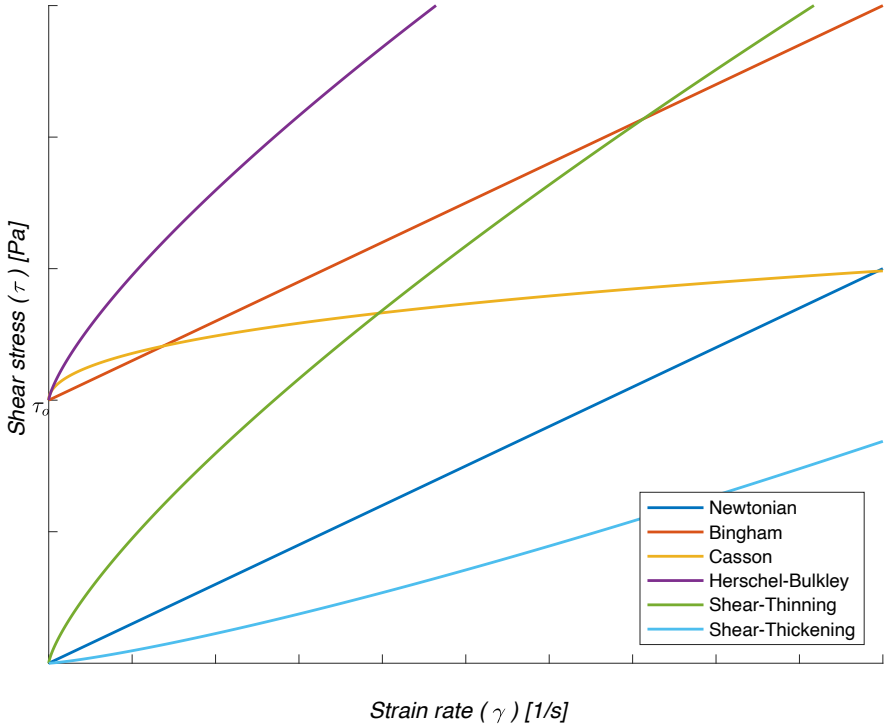


Figure 2.3: Rheological diagrams showing different models

	Rheological Models			
	$k\ [Pa \cdot s^n]$	$n\ [-]$	$\tau_o\ [Pa]$	Law
Newtonian	$\mu > 0$	1	0	$\tau = \mu\,\dot{\gamma}$
Bingham	> 0	1	> 0	$\tau = \tau_o + k\,\dot{\gamma}$
Casson	$\mu > 0$	$\frac{1}{2}$	> 0	$\sqrt{\tau} = \sqrt{\tau_o} + \sqrt{\mu\,\dot{\gamma}}$
Herschel-Bulkley	> 0	$0 < n < \infty$	> 0	$\tau = \tau_o + k\,(\dot{\gamma})^n$
Pseudoplastic / Shear-Thinning	> 0	$0 < n < 1$	0	$\tau = k\,(\dot{\gamma})^n$
Dilatant / Shear-Thickening	> 0	$1 < n < \infty$	0	$\tau = k\,(\dot{\gamma})^n$

Table 2.2: Fluid rheological models

2.1.1. Herschel-Bulkley

The Herschel-Bulkley model (1926) is a general expression so far for modelling the dynamic behavior of a Newtonian and non-Newtonian fluid.

In [16], it is stated that the plastic Herschel-Bulkley model is for non-Newtonian and time independent fluids (the yield stress and the potential law are combined), it is only dependent on the deformation speed (strain rate tensor), so the stress can be expressed as (3.5)

$$\underline{\tau} = f\left(\underline{d}\right) = f\left(\underline{\dot{\gamma}}\right) \quad \rightarrow \quad \tau_{ij} = 2\,\mu(\dot{\gamma})\,d_{ij} = \mu(\dot{\gamma})\,\dot{\gamma}_{ij} \tag{3.5}$$

There is an empirical law called potential law (proposed by *Waele*) that describes non-Newtonian fluid motion (3.6):

$$\tau = \tau_o + k\,\dot{\gamma}^n \quad \rightarrow \quad \frac{d\tau}{d\dot{\gamma}} = n\,k\,\dot{\gamma}^{n-1} \tag{3.6}$$

Where "k" and "n" are constants for a particular fluid. The value of "k" is a measure of the consistency of the fluid; the higher the "k" value, the more "viscous" the fluid. The value of "n" is a measure of the degree of non-Newtonian behavior; the farther away the value of "n" is from unity, the more pronounced the non-Newtonian properties will be of the fluid. To better understand the behavior described by (3.6), Figure 2.4 displays the stress as a function of strain rate.

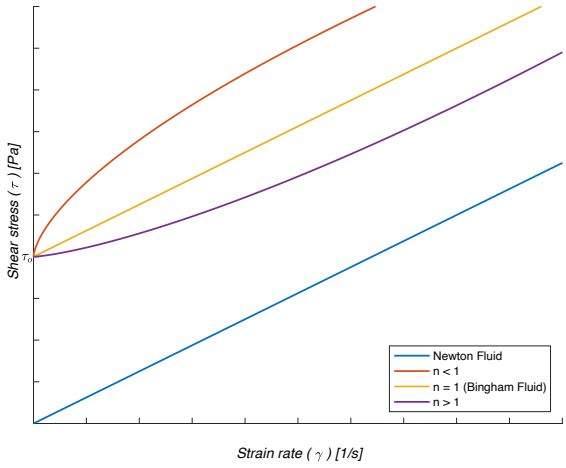


Figure 2.4: Visualization of rheological laws varying "n"

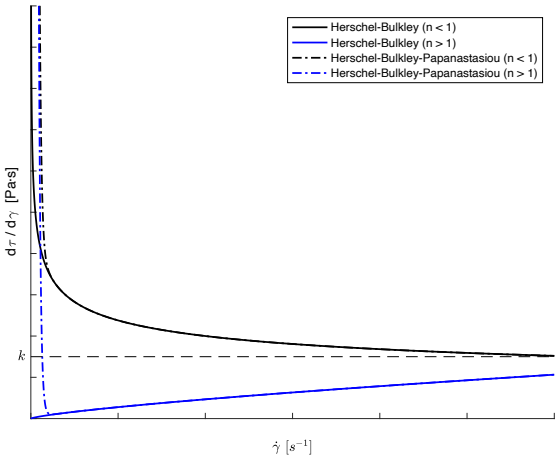


Figure 2.5: Law of rheology derivatives with respect to strain rate

Herschel-Bulkley proposed the following expression to model the viscosity (apparent/dynamic viscosity) as a function of the strain rate tensor norm (3.7).

$$\begin{cases} \mu(\dot{\gamma}) = k\,\dot{\gamma}^{n-1} + \dfrac{\tau_o}{\dot{\gamma}} & for\ \ \tau > \tau_o \\ \dot{\gamma} = 0 & for\ \ \tau \leq \tau_o \end{cases} \tag{3.7}$$

The expression of the dynamic viscosity arises from the combination of the Herschel-Bulkley Law and the proposal of the tangential stresses of the stress tensor (3.8):

$$\left.\begin{array}{l}\tau = 2\,\mu(\dot{\gamma})\,d = \mu(\dot{\gamma})\,\dot{\gamma} \\ \\ \tau = \tau_o + k\,\dot{\gamma}^n\end{array}\right\} \rightarrow \mu(\dot{\gamma})\,\dot{\gamma} = \tau_o + k\,\dot{\gamma}^n \quad \rightarrow \quad \mu(\dot{\gamma}) = \frac{\tau_o}{\dot{\gamma}} + k\,\dot{\gamma}^{n-1} \tag{3.8}$$

The "$\dot{\gamma}$" is the equivalent strain rate, computed as the positive part of the square root of the second invariant J (3.9) (see annex A):

$$\dot{\gamma} = \sqrt{J_2\left(\underline{\underline{\dot{\gamma}}}\right)} = \sqrt{\frac{1}{2}\left(\underline{\underline{\dot{\gamma}}} : \underline{\underline{\dot{\gamma}}}\right)} = \sqrt{2\left(\underline{\underline{d}} : \underline{\underline{d}}\right)} \tag{3.9}$$

The "τ" is the equivalent strain rate, computed as the positive part of the square root of the second invariant J (3.10) (see annex A):

$$\tau = \sqrt{J_2\left(\underline{\underline{\tau}}\right)} = \sqrt{\frac{1}{2}\left(\underline{\underline{\tau}} : \underline{\underline{\tau}}\right)} \tag{3.10}$$

The flow rate can be determined by laboratory tests using the logarithmic (3.11) expression:

$$n = \frac{ln(\tau - \tau_o) - ln(k)}{ln(\dot{\gamma})} \qquad \forall \tau > \tau_o \tag{3.11}$$

By means of this equation, a linear regression can be performed and observe which value predominates when fixed parameters are established in the fluid (density, bulk modulus, yield stress, boundary and initial conditions).

2.1.2. Papanastasiou model

The drawback of the Herschel-Bulkley model is its singularity in the viscosity when "$\dot{\gamma} \rightarrow 0$". In 1987, Papanastasiou proposed a regularization valid both for the unyielded and the yielded zone models. The reason for the use of this regularization parameter is its immediate application and implementation in viscoplastic discontinuous models in numerical solvers [16] [1].

This model is one of the most widely used in solving problems for viscoplastic fluids in numerical calculations due to its ease of implementation (*Mitsoulis and Zisis*, 2001), (*Frey et al.*, 2010), (*Perić* and *Slijecpčević*, 2001), among others.

Its main advantage is the description in a single equation, both in creep ($\tau > \tau_o$) and non-creep ($\tau \leq \tau_o$) zones. It achieves this by means of a smoothed function of the viscosity which is a function of the strain rate "$\dot{\gamma}$" and a regularization parameter "m" described with (3.12) expression. The effect of "m" can be visualized at Figure 2.6.

$$Regularization\ function \quad \rightarrow \quad \left(1 - e^{-m\,\dot{\gamma}}\right) \tag{3.12}$$

This parameter is applied as a product in the yield stress term (3.13) in the Herschel-Bulkley expression, thus modeling, as a continuous function, the law.

$$\tau = \tau_o\left(1 - e^{-m\,\dot{\gamma}}\right) + k\,\dot{\gamma}^n = [Pa] \tag{3.13}$$

Where "e" is the Euler's number.

In (3.13) the derivative with respect to the strain rate of the (3.14) law is formulated, where it can be observed in the Figure 2.5.

$$\frac{d\tau}{d\dot{\gamma}} = \tau_o\left(m\, e^{-m\,\dot{\gamma}}\right) + n\, k\, \dot{\gamma}^{n-1} = [Pa \cdot s] \tag{3.14}$$

The regularization parameter has a key role for the approximation of the Herschel-Bulkley curve for small values of the shear strain rate. The higher is "m", the better is the approximation of the rigid behavior of the original Herschel-Bulkley model. Note that "m" has the dimension of the time [s].

In [16], states that the viscosity is bounded when the strain rate gradient tends to zero. For very high values of the regularization parameter "m" this limit value of the viscosity may cause numerical problems.

$$\mu_{max} = \lim_{\dot{\gamma}\to o} \mu(\dot{\gamma}) \tag{3.15}$$

The limiting value of the viscosity when the strain rate tends to zero varies according to the value "n", as can be seen in Table 2.3 (limits solved by L'Hôpital's Rule).

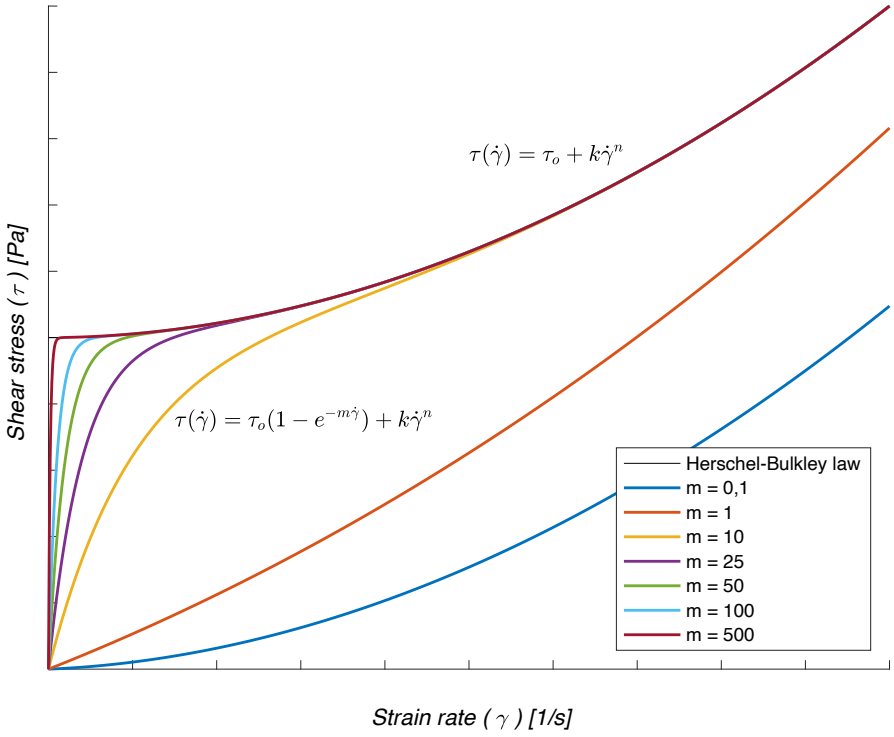


Figure 2.6: Effects of the Papanastasiou regularizing parameter on the Herschel-Bulkley law.

It is observed that for pseudoplastic fluids ($n < 1$) the viscosity is not bounded. In these cases, it is essential to apply the truncation procedure express in (3.16) and visualized in Figure 2.7.

Exponent "n"	*Viscosity "μ_{max}"*
$n < 1$	∞
$n = 1$	$k + \tau_o\, m$
$n > 1$	$\tau_o\, m$

Table 2.3: Results of the calculation of the (3.15) limit for different fluid indexes

$$Truncation\ value \quad \rightarrow \quad \mu_t < \mu_{max} \ when \ \dot{\gamma} < \dot{\gamma}_t \tag{3.16}$$

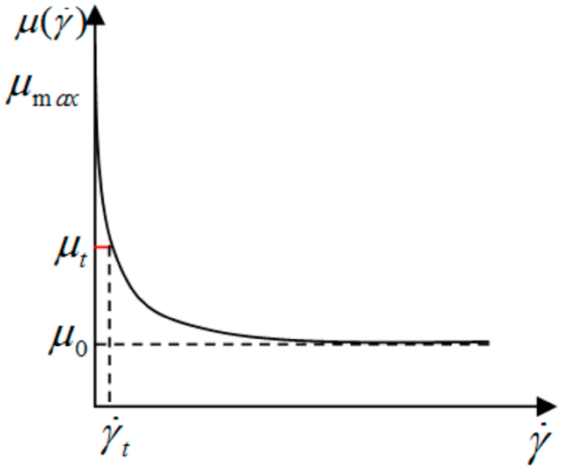


Figure 2.7: Truncation value for extreme cases [16]

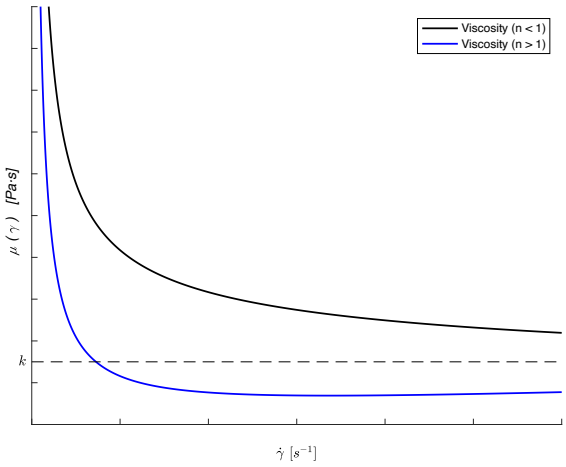


Figure 2.8: Dynamic viscosity as a strain rate function

2.2. Particle Finite Element Method (PFEM)

The PFEM is a numerical technique developed for the solution of multi-physics problems involving large deformations of the domain [17].

The birth of this method can be considered to have occurred in 2004, with the publication of the article "*The particle finite element method: a powerful tool to solve incompressible flows with the free-surfaces and breaking waves (S.R. Idelsohn, E. Oñate and F. Del Pin)*" on which the present chapter is based.

The initial aim was to "Analyze problems in which the interface changes continuously or in fluid-structure interaction with free-surfaces where complicated contact problems are involved" [17].

The key idea of the PFEM is to combine a Lagrangian Finite Element Method (FEM) with an efficient and fast remeshing procedure. In the PFEM, the domain is defined by a set of particles (coinciding with the mesh nodes) that move in a Lagrangian manner according to the calculated nodal variables (e.g., velocity or displacements) and bringing their physical properties.

The PFEM can be seen as both a FEM-based (described in Annex C) and a particle method. A summary of the rationale for both theories is attached at Table 2.4 and Table 2.5.

- Computational domain seen as a collection of material particles.
- All physical properties (density, viscosity, bulk modulus, etc.) and variables (pressure, velocity, temperature, etc.) are assigned to each particle.
- Particles move according to the external and the interaction forces.

Table 2.4: Particle method fundamentals

$$\boldsymbol{PFEM = FEM\ bases \cup Particle\ method}$$

- A mesh is generated over the particles to define shapes functions and solve the governing equations in a Lagrangian way.
- All physical properties (density, viscosity, bulk modulus, etc.) and variables (pressure, velocity, temperature, etc.) are assigned to each particle.
- Particles move according to the external and the interaction forces.

Table 2.5: Finite Element Method (FEM) bases

2.2.1. Description

For continuum mechanics problems, in the Eulerian (Spatial) way to described the medium motion, the finite element mesh is fixed and the material moves across the grid, being the mesh nodes dissociated from physical particles. Due to the relative motion between the material and the grid, convective terms appear in the definition of the time derivatives (view *Annex B*).

Eulerian meshes are particularly suited for large deformation problems in enclosed domains, as those generally considered in standard computational fluid dynamics. On the other hand, they do not provide a natural definition of evolving interfaces (like a free surface in fluid flows).

Lagrangian (Material) way to described the medium motion, the finite element mesh moves along with the continuum body. Consequently, boundaries and interfaces are naturally tracked during the motion allowing for a simpler imposition of boundary conditions. As the material points coincide with the grid nodes, no convective terms appear in the governing equations and material derivatives reduce to time derivatives. Also, the integration points (numerical integration) move with the material, so constitutive laws are evaluated at the same material points for all the duration of the analysis [17].

> The Eulerian description of motion is given to fluid mechanics while the Lagrangian description is given to solid mechanics (annex A).
>
> However (and here is the novelty), PFEM uses the Lagrangian description to solve problems in fluid mechanics.

Since the Lagrangian method allows the mesh to move, the mesh deteriorates with motion. These phenomena constituted for a long time the intrinsic limit of Lagrangian mesh-based solvers. In the literature, there exist two different options to overcome this endemic feature:

1) To introduce a remeshing technique.
2) To abandon completely the concept of mesh, giving rise to the so-called meshless methods.

PFEM deals with the problem of mesh distortion with remeshing. When the mesh becomes too distorted, a new mesh is made with an *ad-hoc* procedure.

Unavoidably, these remeshing operation introduce unwanted numerical diffusion into the numerical solution. That's the reason why PFEM was proposed by *Idelsohn*, *Oñate* and coworkers in 2004.

The PFEM combines the accuracy and robustness of mesh-based techniques with the advantages of particle-based methods. The PFEM discretizes the physical domain with a mesh on which the differential governing equations are solved with a standard finite element approach. To avoid remapping from mesh to mesh, it keeps the nodes of the previous mesh fixed. The new connectivity is built using the Delaunay Triangulation (mesh regeneration algorithm) and a specific technique (alpha-shape method) is used to identify internal and external boundaries. The obtained mesh is then used as the support over which the differential equations are solved in a standard FEM fashion [17].

At the Table 2.6, it presents a summary of the fundamental features:

Fundamental features PFEM
• Lagrangian description of the mechanics problem. • Mesh nodes are treated as physical particles. • All the information is stored at the mesh nodes. • The FEM is used to solve the governing equations. • Mesh connectivity by Delaunay Tessellation. • Boundaries are recovered through *ad-hoc* techniques (alpha-shape method).

Table 2.6: Summary of the fundamental features of the PFEM [17]

2.2.2. Steps

A general solution scheme of the method can be summarized in Table 2.7:

PFEM steps
1) Fill the domain with a set of points referred to as “particles”. 2) Generate a finite element mesh using the particles as nodes. 3) Identify the external and internal body’s boundaries. 4) Solve the Lagrangian form of the governing equations with the FEM. 5) Update the positions of the nodes. 6) Proceed to the next time step. If a remeshing is needed, go to step 2 otherwise, go to step 5.

Table 2.7: Summary of the PFEM steps [17]

In the second step, the mesh can be regenerated with the Delaunay triangulation. The identification of boundaries in the third step, needed to compute the domain integrals and to impose correctly the boundary conditions, is performed using the alpha shape method.

> It is also important to remark that equations of motions solved in fourth step, can be non-linear and so they may require an iterative solution scheme (annex C).

2.2.3. Mesh

In the PFEM, the re-generation of the mesh should be considered a redefinition of the element connectivity rather than a real remeshing because the mesh nodes of the previous mesh are kept in the same position.

The mesh nodes move according to the equation of motion, behaving like particles and the transporting their momentum together with all their physical properties [17].

2.2.3.1. Delaunay triangulation

The Delaunay triangulation is a direct consequence of the Voronoi diagrams (described in Table 2.8).

The Delaunay triangulation can be constructed by joining the points whose Voronoi cells have a common boundary. It is de dual of the Voronoi diagram. The Delaunay tessellation generates a mesh of tetrahedra (in 3D) and triangles (in 2D).

None of its vertices lays inside any tetrahedron’s circumsphere (in 3D) or triangle’s circumcircle (in 2D).

Moreover, the Voronoi cells vertices represent the center of tetrahedron’s circumsphere (in 3D) or triangle’s circumcircle (in 2D) of the Delaunay triangulation. Given a set of points in space, the Voronoi diagram is unique, but may exists different Delaunay triangulations [17].

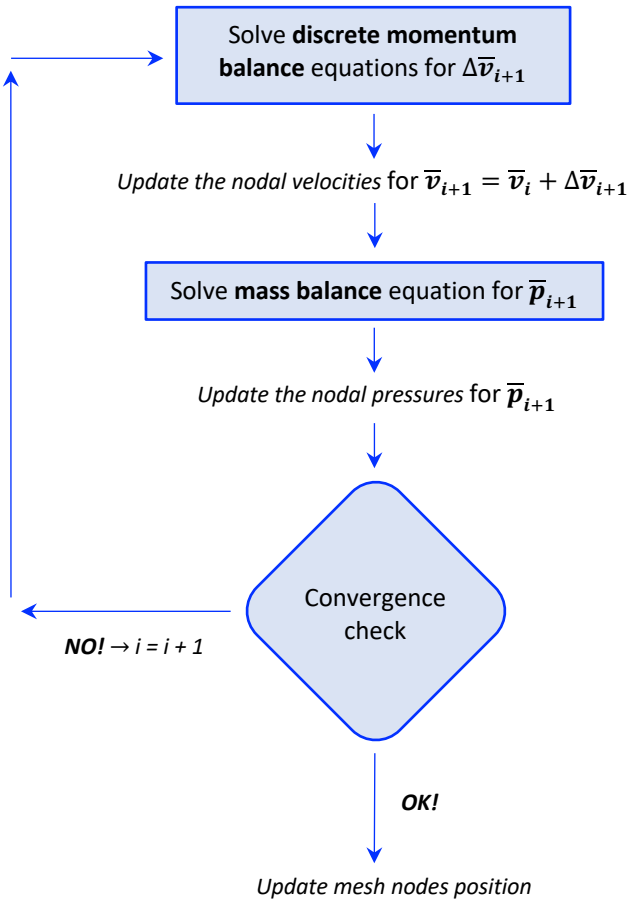


Figure 2.9: Schematic representation of the iterative solution of a generic time step. The subindex "i" represents the iteration number of the implicit solution [7]

Given a set of "*n*" points, the Voronoi diagram is defined as the partition of $\mathbb{R}^3$ in convex regions "T_i", where a node "n_i" is associated to each region "T_i", such that every point of "n_i" is closer to "T_i" than to any other nodes "n_j" with $i \neq j$.

$$Voronoi\ Cell \quad \rightarrow \quad T_i = \left\{ \underline{x} \in \mathbb{R}^3 : \ \left\| \underline{x} - \underline{x}_i \right\|_2 \leq \left\| \underline{x} - \underline{x}_j \right\|_2, \ \forall i \neq j \right\}$$

The region "T_i" is the set of points closest to a given point "$\underline{x}$"

Each Voronoi cell is convex and close if internal and open if placed at the boundary.

$$Voronoi\ Diagrams = Thiessen\ Polygons = Dirichlet\ Tesselation$$

Table 2.8: Description of the Voronoi Diagrams [18]

It is not a must to use Delaunay triangulation for every PFEM problem to be solved. Other methods could be used as long as the mesh obtained was made quickly from an initial distribution of points.

2.2.3.2. Alpha-Shape technique

The basic idea is to remove the unnecessary elements from the mesh using a geometrical criterion based on the mesh distortion. It is a pure geometrical check: it erases from the mesh all those elements that are too large or distorted. For each element "*e*" of the mesh, an index of elemental distortion "α_e" is defined as (3.16):

$$\alpha_e = \frac{R_e}{h_{mean}} \tag{3.16}$$

Where "R_e" is the radius of the circumsphere (3D) or circumcircle (2D) to the considered element and "h_{mean}" is a characteristic mesh size. An example of "h_{mean}" can be the average of the minimum element size among all the elements of the initial mesh.

A threshold value "$\overline{\alpha}$" for the distortion of the mesh can be fixed and, consequently, all the elements that not satisfy the condition "$\alpha_e \leq \overline{\alpha}$" are removed from the mesh. It is allowed to use different values of "$\overline{\alpha}$" for distinct parts of the domain. Different values of "α" may lead to different meshes and computational domains. All particles on an empty sphere with a radius $r(x) > \overline{\alpha} \cdot h(x)$ are considered as boundary particles [17].

One of the main strengths of the PFEM lays in its capability to model separation and reconnection of parts of the computational domain and also single isolated particles. The identification of the parts detaching from the rest of the domain is done automatically by the alpha-shape method.

Suggested values for 2D analyses [19]:

$\alpha \approx 1{,}2 \div 1{,}25$

Numerical simulations of the validation examples considered in this work use an alpha shape of 1.25 in 2D and 1.3 in 3D.

2.2.3.3. Adding and removing nodes

In the PFEM, insertion and removal of mesh nodes can be safely done because the mass is not associated with the nodes but with the elements, as in standard FEM.

Different algorithms to add and remove particle have been proposed. However, the key idea is always based on the following two concepts:

- If a node comes too close to another (or to a boundary) the node should be removed (or moved to another location).
- If an element becomes too large, a new node should be inserted.

It is a must to avoid the "*artificial leakage*" in the removing nodes situation. Adding and removing nodes can also be done with-out altering the total number of nodes. In this case, a new node is added only if a node can be removed from another position [17].

2.2.4. Summary

Table 2.9 summarizes the advantages and disadvantages of this method.

Advantages	Disadvantages
• It allows highly deforming domains: free surface, separation/reconnection of subdomains. • It guarantees good mesh quality at each FEM solution step. • It allows a Lagrangian description of motion: natural modeling of convection. • It allows the detection of fixed or moving solid interfaces.	• It may include artificial changes of topology. • It may induce loss of mass conservation. • It represents an additional computational cost. • It induces the loss of element information stored at Gauss point. • FEM structures must be rebuilt continuously.

Table 2.9: PFEM remeshing advantages and disadvantages

2.3. PFEM for fluid dynamics problems

The PFEM was originally conceived for the solution of free-surface fluid flow problems. Thanks to its Lagrangian description of the motion, the convective terms do not enter in the PFEM governing equations. This is a great advantage as those terms are responsible for non-linearity and non-symmetry, thereby complicating significantly the solution of the governing equations in an Eulerian framework and typically requiring the introduction of stabilization terms to avoid numerical oscillations [17].

The price to pay is the need to continuously remesh the computational domain. This fact has important implications on several aspects of the numerical solver, such as time integration and spatial discretization, the imposition of boundary conditions, or mass conservation.

2.3.1. Problem statement

This section includes the balance equations governing a fluid (defined in Annex B).

2.3.2. Space discretization and stabilization

In the PFEM, a Standard Galerkin approach is used to discretize in space the mass conservation and linear momentum balance equations. In a standard PFEM framework, only linear shape functions are used to approximate the unknown variables. Introducing an isoparametric (same shape/interpolation functions) finite element discretization, the velocity and the pressure can be expanded in terms of the nodal vectors "$\underline{V}$" and "$\underline{P}$", respectively. The semi-discretized motion equations are expressed in (3.17), developed in [17].

$$\begin{cases} \underline{\underline{M}}_v \cdot \dot{\underline{V}} + \underline{\underline{K}} \cdot \underline{V} + \underline{\underline{D}}^T \cdot \underline{P} = \underline{F} & \rightarrow \quad Linear\ momentum \\ \underline{\underline{M}}_p \cdot \dot{\underline{P}} + \kappa\, \underline{\underline{D}} \cdot \underline{V} = \underline{0} & \rightarrow \quad Mass\ conservation \end{cases} \tag{3.17}$$

$Initial\ conditions$	$\rightarrow$	$f = f_o$	$f \in \Gamma_o$
$Boundary\ conditions$	$\rightarrow$	$f = f_{\mathcal{D}}$	$f \in \Gamma_{\mathcal{D}irichlet}$
		$f = f_{\mathcal{N}}$	$f \in \Gamma_{\mathcal{N}eumann}$

Table 2.10: IVC & BVC

Where "$\underline{\underline{M}}_v$" is mass matrix for velocity unknown, "$\underline{\underline{M}}_p$" is mass matrix for pressure unknown, "$\underline{\underline{K}}$" is the fluid matrix emanating from viscosity term, "$\underline{\underline{D}}$" is the discretized divergence differential operator and "$\underline{F}$" is the body forces vector and conditions (Table 2.10).

Those equations must be integrated in time and thus an approximation for the time derivative of pressure and velocities should be provided (3.20), but not before presenting in (3.18) and (3.19) the discretization of the variables to be determined

$$Velocity\ vector\ discretization \quad \rightarrow \quad \underline{v}(\underline{x},t) \approx \underline{v}^h = \sum_i N_i(\underline{x})\, \underline{V}_i(t) \tag{3.18}$$

$$Pressure\ scalar\ discretization \quad \rightarrow \quad p(\underline{x},t) \approx p^h = \sum_i N_i(\underline{x})\, P_i(t) \tag{3.19}$$

Where "$N_i(\underline{x})$" are linear shape functions.

The time derivative can be computed as:

$$\frac{d\underline{V}}{dt} = \frac{\underline{V}^{n+1} - \underline{V}^n}{\Delta t} \qquad \frac{d\underline{P}}{dt} = \frac{\underline{P}^{n+1} - \underline{P}^n}{\Delta t} \tag{3.20}$$

As described in [7], the nodal accelerations can be computed according to the implicit Newmark integration (3.21)

$$\dot{\underline{V}}^{\,n+1} = \frac{2\left(\underline{V}^{\,n+1} - \underline{V}^{\,n}\right)}{\Delta t} - \dot{\underline{V}}^{\,n} \qquad \dot{\underline{P}}^{\,n+1} = \frac{\underline{P}^{\,n+1} - \underline{P}^{\,n}}{\Delta t} \tag{3.21}$$

Using an equal order interpolation for both the pressure and velocity unknowns, the compatibility condition is not fulfilled. Hence, the formulation must be stabilized. In this work it has been used the Finite Increment Calculus (FIC) [17].

3. Validation against literature problems

3.1. Bingham validation

3.1.1. Initial validations

The first validation step is to check and validate the code programmed in Kratos (annex D) using PFEM and the Bingham-Papanastasiou model. This is because the code is the basis for programming the Herschel-Bulkley-Papanastasiou model.

For this purpose, the problems presented in [20] are taken as reference.

The study focuses on the rheological characterization of so-called liquefied sands as Bingham fluid by means of numerical simulation of dam-break. It is important to note that the mentioned article also describes the fluid motion using the PFEM.

In the reference paper, various tests are analyzed for the same initial geometry but varying mechanical properties such as yield stress "τ_o" or dynamic viscosity "μ".

A difference to take into account between the simulations perpetuated by the authors of the article and this work are: mass conservation equation. In the validation article the fluids are stipulated as incompressible, however in this draft the situations were realized by means of a quasi-incompressible formulation (annex B).

Another difference is the density used: the numerical reference uses a density of 1600 kg/m^3, as does the validation in this work, but the laboratory test determined a dry density of 1510 kg/m^3. The same density proposed in the reference work is used here.

Each of the simulations has been performed with the following geometry illustrated in Figure 3.1 (length units are in meters):

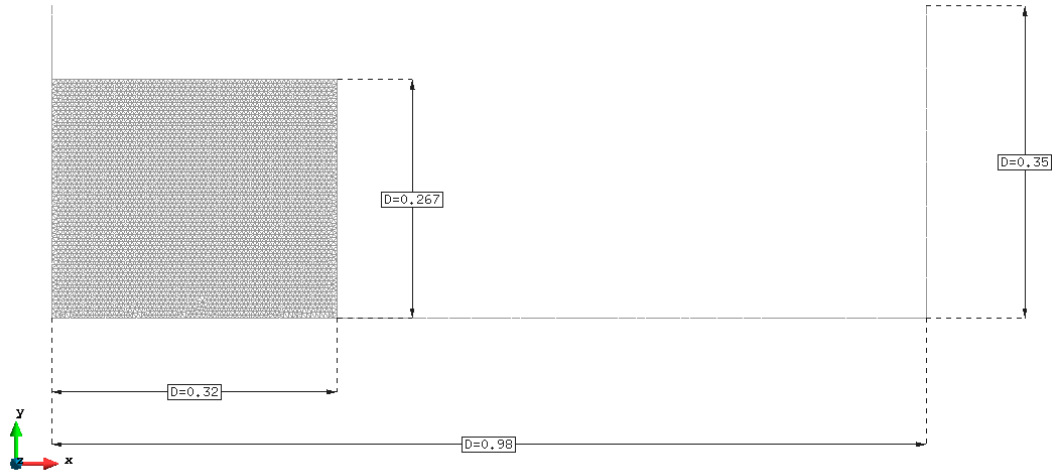


Figure 3.1: Geometry defined for Bingham validation

It is a simple geometry, a rectangle initially at rest (velocity equal to 0 in all the elements that compose it and zero pressure at the top of the fluid). The walls are considered as rigid bodies, that is, they do not suffer deformations and the velocity of the elements that compose it is zero, however, the pressure does vary according to the dynamics of the fluid in motion.

The fluid starts its movement due to the gravity (by its own weight). It is discretized with triangular two-dimensional elements with linear shape functions.

Both simulations have been carried out with more than 3000 finite elements discretizing the fluid (in this work a mesh size about $h^{mesh} = 0{,}006\ m$ has been used to achieve this), using a time step $\Delta t^{step} = 0{,}001\ s$ and a Papanastasiou parameter $m = 1000\ s$.

The properties of the two simulations are described at Table 3.1:

	$t_{end}\ [s]$	$\Delta t^{step}\ [s]$	$h^{mesh}\ [m]$	$\rho\ \left[\frac{kg}{m^3}\right]$	$\kappa\ [Pa]$	$\mu\ [Pa \cdot s]$	$\tau_o\ [Pa]$	$m\ [s]$
Bingham validation 1	6	0,001	0,006	1600	$2{,}1 \cdot 10^9$	200	100	1000
Bingham validation 2						300	50	

Table 3.1: Bingham and simulation parameters

The following experimental results illustrate the horizontal position of the liquefied mass tip as a function of time for a total duration of 6 seconds (Figure 3.2 and Figure 3.4). PFEM simulations with the code proposed by this write-up and the validation comparison code are also included in the results shown in the plots. The difference of results, measured in percentage %, with respect to the numerical reference, has been added beside them.

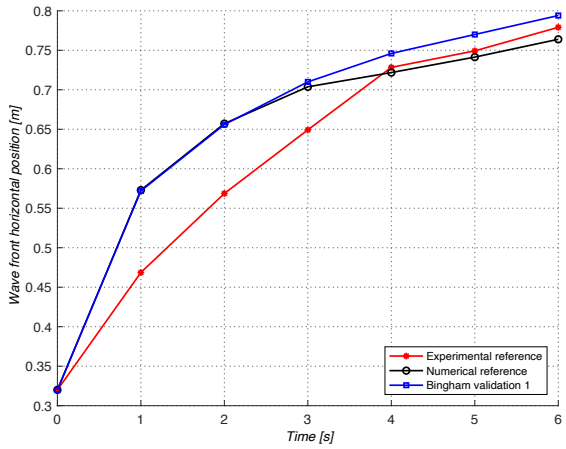


Figure 3.2: Bingham validation 1 results

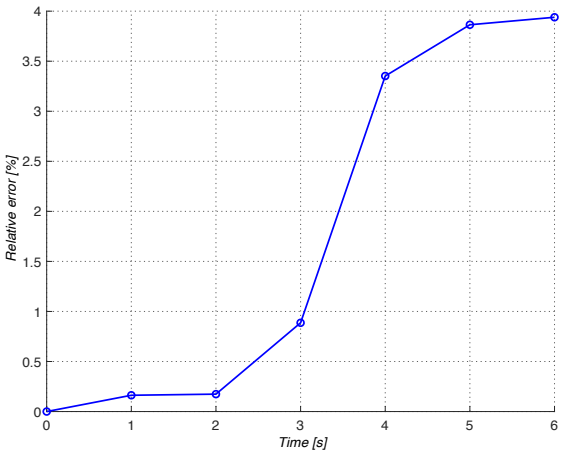


Figure 3.3: Variation of results between Bingham 1 and reference

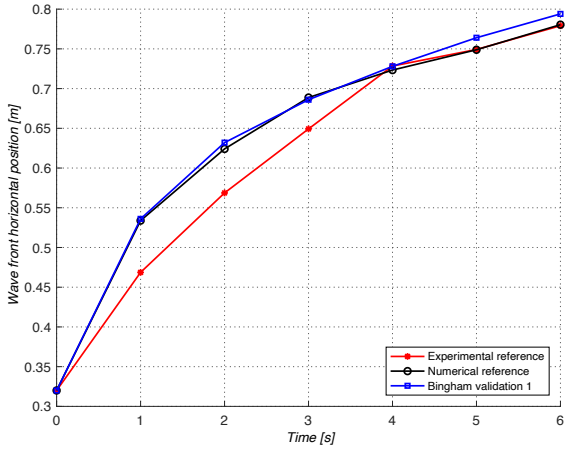


Figure 3.4: Bingham validation 2 results

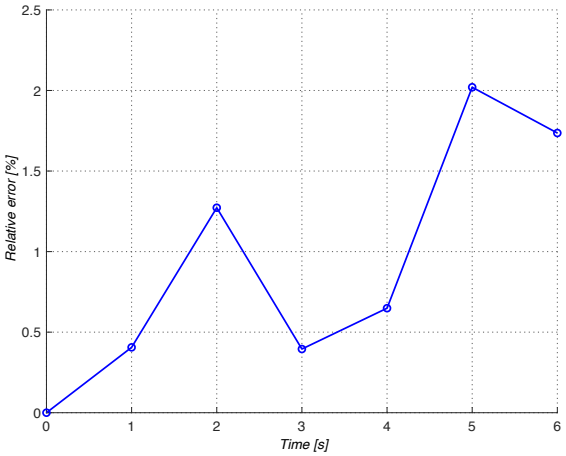


Figure 3.5: Variation of results between Bingham 2 and reference

It can be seen that the first simulation (Figure 3.3) starts out giving similar results to the reference simulation, but as time progresses, it becomes more deformed than the reference simulation, arriving at the end of the calculation with a difference less than 4%.

The second simulation starts as the first one, with values similar to the reference values (Figure 3.5). However, once the calculation time has elapsed, it continues with similar values, ending the calculation with a difference to the reference values of less than 2%.

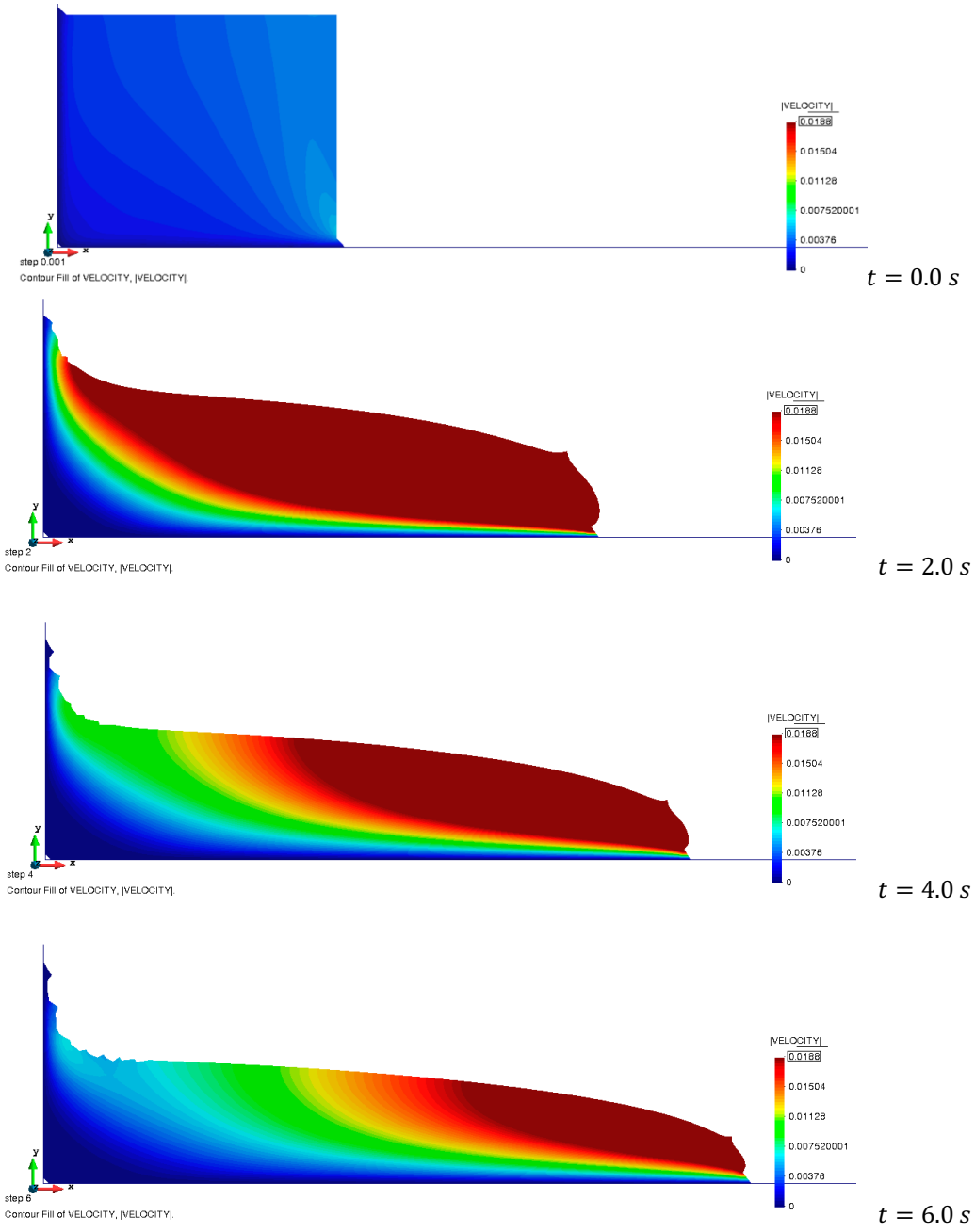


Figure 3.6: Velocity contour plots of Bingham 2 validation

The same plotted contours of the velocity field extracted from [20] are attached in Figure 3.7in order to compare the results calculated using one formulation and the other. The maximum speed of color printing has been limited to facilitate visual analysis.

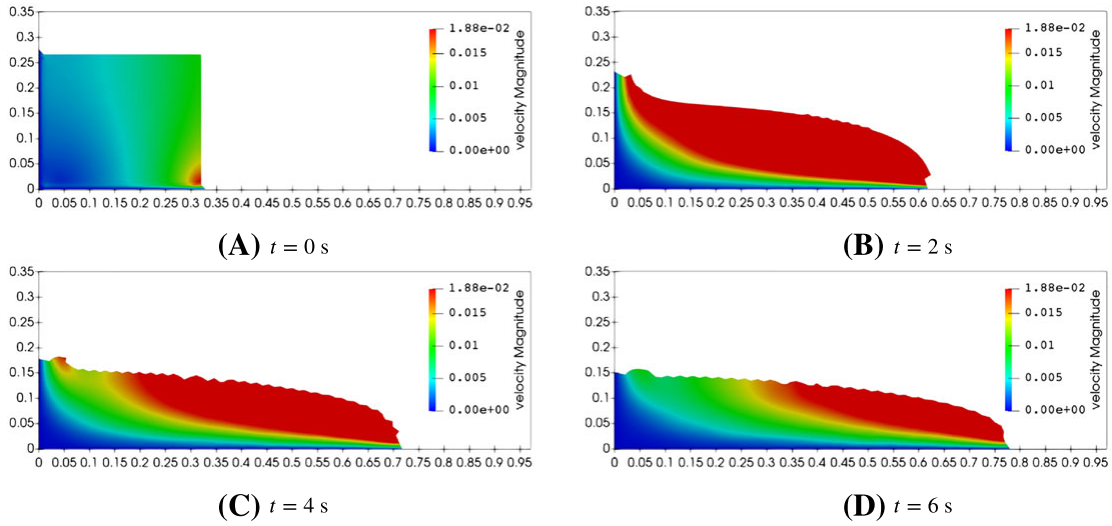


Figure 3.7: Velocity field. Source: [20]

It can be seen that a large region of the fluid exceeds the velocity of 0.0188 from the very beginning. In both simulations the commented region decreases with time in a similar way. A notable difference is the amount of fluid that remains attached to the vertical wall. In this work it is clearly observed that it remains much more clinging than in the reference work.

3.1.2. Varying Bingham parameter

After the two previous simulations, the model continues to be validated with the next step of the same article: comparison of the fluid front advance by varying the two main characteristics of the Bingham model: yield stress and dynamic viscosity.

3.1.2.1. Varying viscosity and constant yield stress

The simulations are performed with the same geometrical characteristics as in section 1.1 of this chapter. The differences are the variations of the mechanical properties of the Bingham model. In this first section, it has varied the dynamic viscosity of the fluid, keeping the yield stress constant, in order to analyze how this affects its movement. It can be clearly seen that there are differences in the motion between one simulation code and another.

3.1.2.2. Varying yield stress and constant viscosity

The simulations are performed with the same geometrical characteristics as in section 1.1 of this chapter. The differences are the variations of the mechanical properties of the Bingham model. In this second section, it has varied the yield stress of the fluid, keeping the dynamic viscosity constant, in order to analyze how this affects its movement. The results of these last three simulations leave open the following hypothesis: the lower the yield stress, the greater the variation of the results with the reference values, and these variations increase with the passage of time. However, when the yield stress is large, then the variation is smaller and even appears to be decreasing over the simulation time.

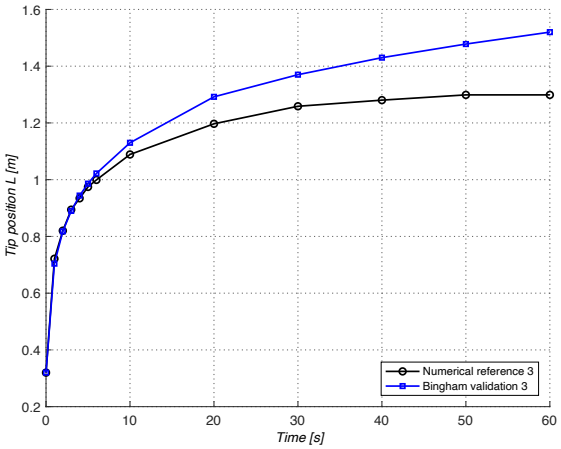


Figure 3.8: Bingham validation 3 (μ=100 Pa·s ,τ_o=20 Pa)

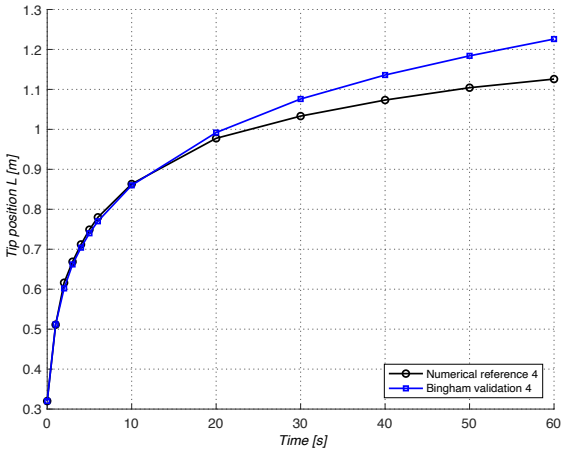


Figure 3.9: Bingham validation 4 (μ=400 Pa·s ,τ_o=20 Pa)

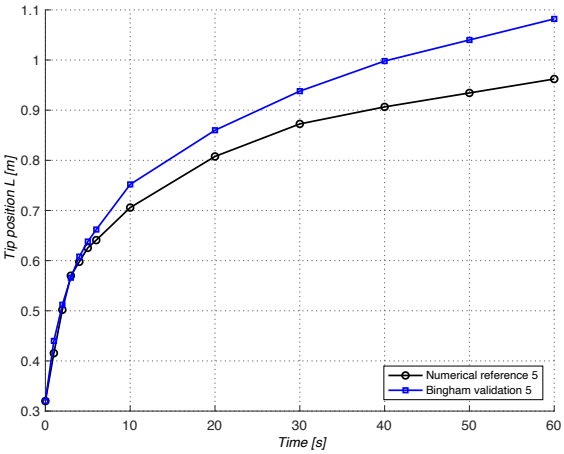


Figure 3.10: Bingham validation 5 (μ=800 Pa·s ,τ_o=20 Pa)

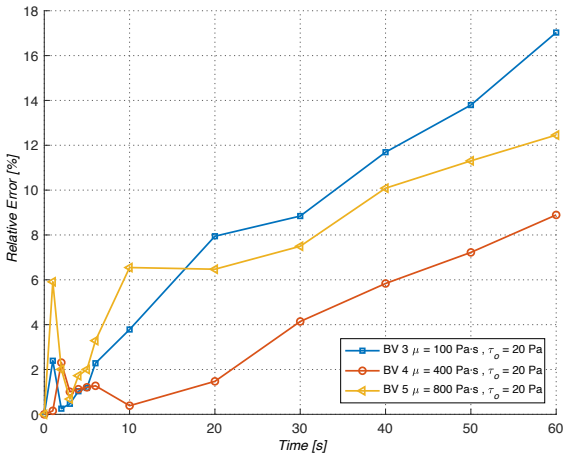


Figure 3.11: Variation of results between Bingham and references

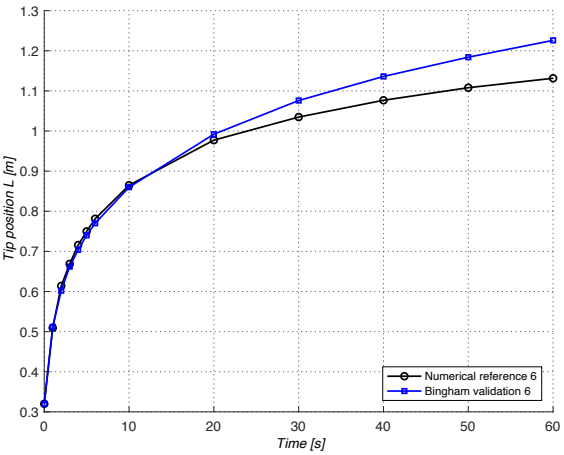


Figure 3.12: Bingham validation 6 (μ=400 Pa·s ,τ_o=20 Pa)

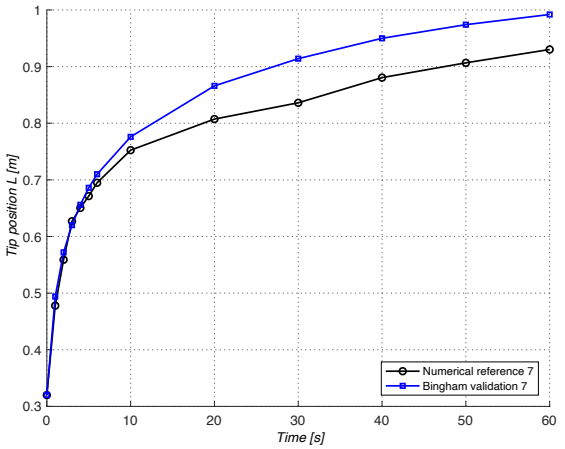


Figure 3.13: Bingham validation 7 (μ=400 Pa·s ,τ_o=100 Pa)

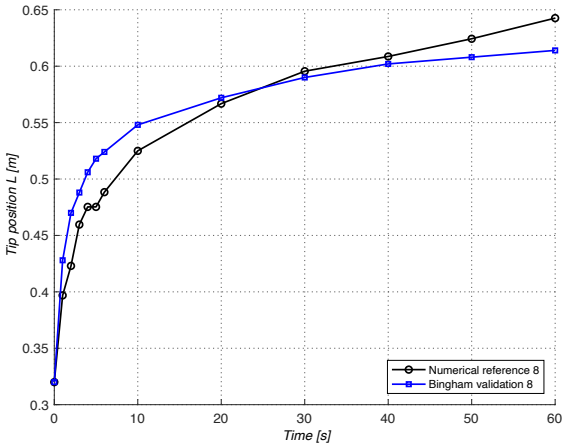


Figure 3.14: Bingham validation 8 (μ=400 Pa·s ,τ_o=400 Pa)

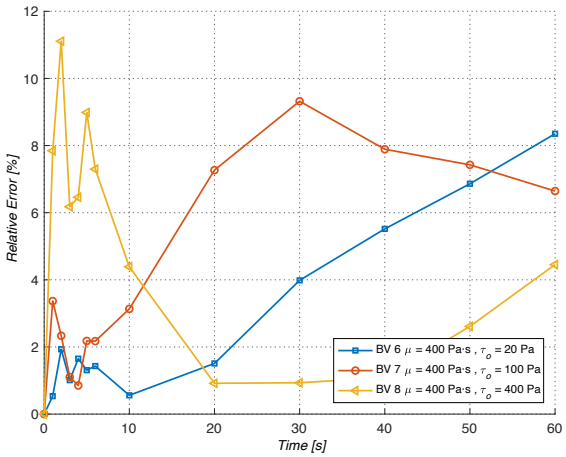


Figure 3.15: Variation of results between Bingham and references

3.2. Herschel-Bulkley validation

This section has been based on and guided by the following articles: [21] and [22].

For the validation of the Herschel-Bulkley-Papanastasiou model, some previous steps had to be taken before testing the model against another article. The drawback was not knowing which time step "Δt^{step}", mesh size "h^{mesh}" and parameter "m" to use in the validations. Due to this challenge, the solution was, based on the characteristics of the first test of the article, to run several simulations in order to compare errors from one simulation to another.

The steps to follow are described in Table 3.2:

$$t_{end} \rightarrow h^{mesh} \rightarrow \Delta t^{step} \rightarrow m \rightarrow Tests\ HB-2D \rightarrow Tests\ HB-3D$$

Table 3.2: Scheme followed in the validation of Herschel-Bulkley

Numerical simulations, in the comparison work, were performed using the VOF (Volume of Fluid Method) in a two-dimensional domain through the numerical package ANSYS CFX [21].

This time, the study reference uses the mass conservation equation in all its expression, treating the fluid to be simulated as compressible, while the PFEM treats the fluid as quasi-incompressible (annex B):

The geometry to be used, in two spatial dimensions, is as follows in Figure 3.16:

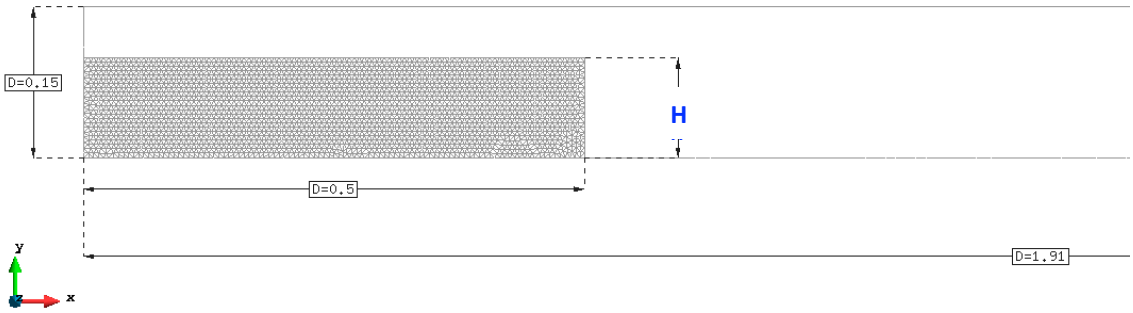


Figure 3.16: Geometry defined for Herschel-Bulkley 2D validation

The geometry to be used, in three spatial dimensions, is identical to the two-dimensional geometry, by extruding in the third dimension the length shown in the Figure 3.17 below:

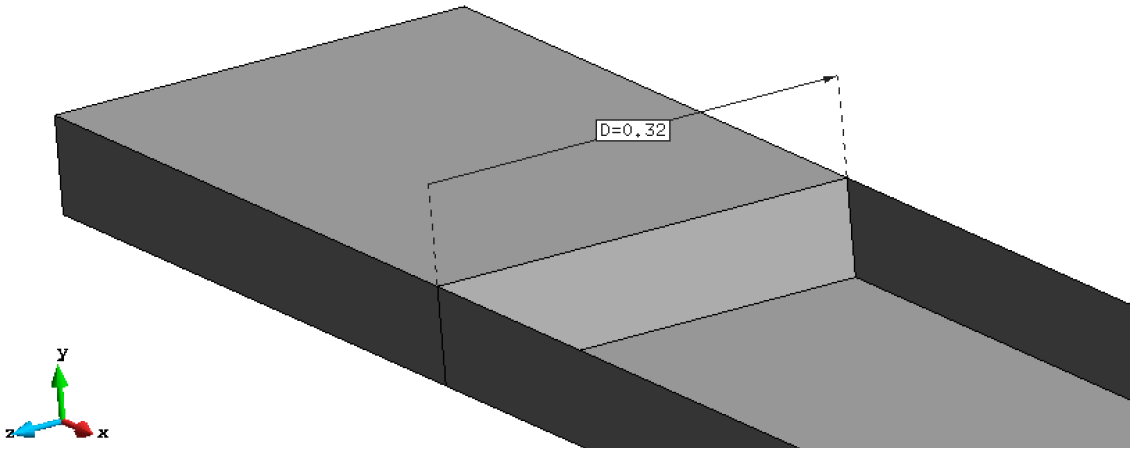


Figure 3.17: Geometry defined for Herschel-Bulkley 3D validation

A zero velocity along the entire contour of the fluid retaining walls has been imposed as a boundary condition.

Necessary condition to be fulfilled for the calculation at the time of the numerical simulations:

$$h^{mesh} > v^{run} \cdot \Delta t^{step}$$

The time step variation Δt^{step} is something that can be set (chosen). The calculation speed is what it is (each processor has its own capacity). The mesh size "h^{mesh}" is also selected.

This condition must be taken into account because otherwise, the "particles" (the mesh nodes) would advance a distance greater than the mesh dimensions and may cause topological inconvenient and consequently errors in the computation (for example, it does not respect the boundary conditions).

3.2.1. Two dimensions validation

3.2.1.1. End time

The first step in validation is to know how much to simulate in order to be able to compare, beyond the values and parameters established by the article.

For this purpose, Test 1 has been modeled, recreating its geometry and properties.

It was decided to set a step time $\Delta t^{step} = 0{,}001$ seconds, a mesh size $h^{mesh} = 0{,}005$ meters and an adaptive exponent (Papanastasiou parameter) $m = 1000$ seconds.

This analysis shown in Figure 3.18 is also used to verify whether both Herschel-Bulkley law and Bingham's law end up at the same point in terms of fluid motions.

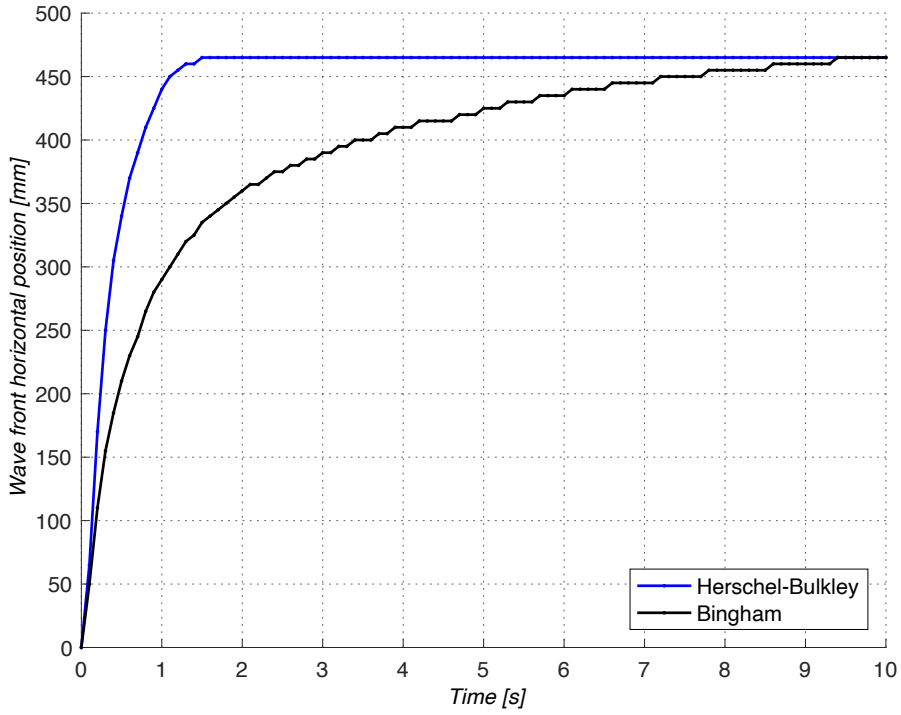


Figure 3.18: End time of movement according to the model used

It can be seen that with both the one and the other, the movement effectively ends at the same location. This is due to the fact that the dynamics is influenced, above all, by the yield stress since, if this stress value is not exceeded, there is no movement.

However, it is observed that to reach the same stopping point in the movement, much more end time t_{end} is required with its resulting computational cost. With Herschel-Bulkley, motion arrest occurs at 1,5 seconds and with Bingham it takes more than 9 seconds. Completion time is reduced by more than 83%.

Figure 3.18 shows what in [20] has already been commented. The stopping of the motion is influenced by the yield stress. If the inertia of the motion of the fluid itself is small, using both the Bingham and Herschel-Bulkley laws, the fluid itself will tend to stop at the same point of displacement. Conversely, if the inertia of the motion is high enough, using one law or the other will cause the motion to stop at a different point. Herschel-Bulkley will stop earlier or later than Bingham depending on the fluid index "*n*" taken as a value.

It can be seen that the movement according to the Bingham model takes longer to occur as time passes.

With this analysis it is possible to proceed to the next one establishing a final time of 1,5 seconds as in the basic article for validation.

$$\boldsymbol{t_{end} = 1,5\ s}$$

3.2.1.2. Mesh size

In this section, the effect of the mesh size on the numerical results is analyzed. The process consists of, starting from the geometrical and mechanical properties of Test 1, comparing different simulations by varying the mesh size, observing the computational cost (run time t^{run}), the displacement suffered by the fluid, with their respective relative error committed.

It has been tested from mesh sizes of 0,1 meters (which did not result in any calculation due to poor discretization of the medium and its properties) to a mesh size of 0,001 meters, having up to 100 elements in the high of the fluid at the beginning of the simulation. The Figure 3.19 shows each of the results of the numerical calculation. There is a clear increase in computational cost when using mesh sizes smaller than 0,01 meters. Even so, it is worth mentioning that from then on, as can be seen in the Figure 3.21, the error is less than 3,2%, leading to the conclusion that a value within that size range is sufficient to continue with the simulations. The variation in movement between a size of 0,01 and 0,001 meters is only 9 millimeters.

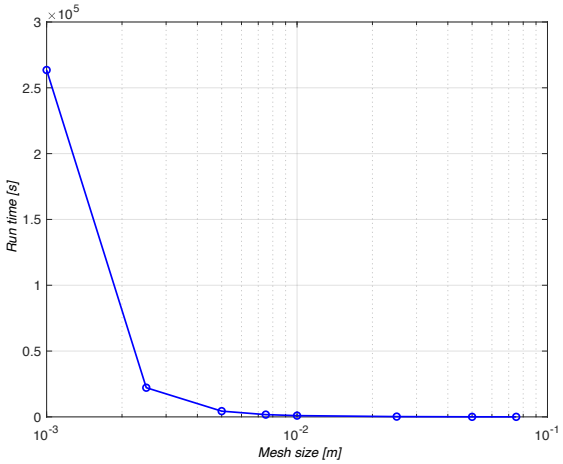


Figure 3.19: Comparison of running time with varying mesh size

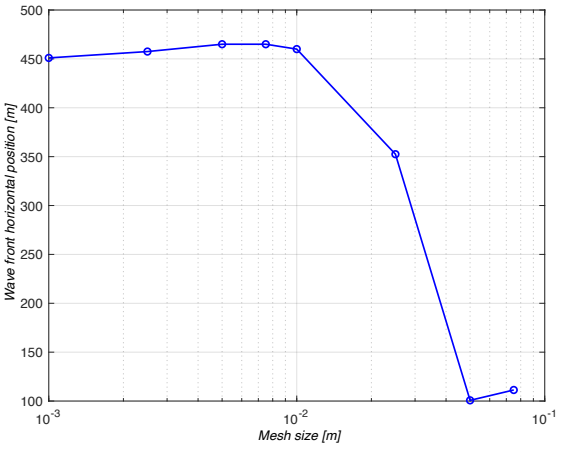


Figure 3.20: Comparison of the fluid front as a mesh size function

In Figure 3.20, a decrease in the value of the final horizontal wave front position is observed as the mesh size decreases.

With this analysis it is possible to proceed to the next one establishing a mesh size of 0,005 meters. This is due to the fact that a decrease in size implies an excessive computational cost for the numerical simulations and since it produces a small relative error, this advantage should be taken advantage of.

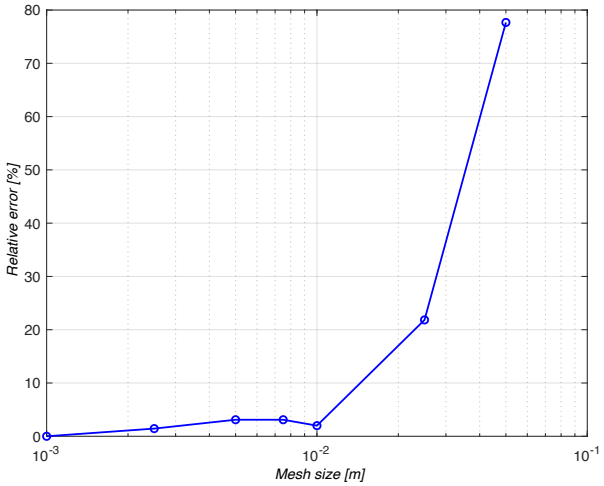


Figure 3.21: Relative error varying the mesh size

$$\boldsymbol{h^{mesh} = 0,005\ m}$$

3.2.1.3. Time step

This section determines the time step to be used in the validation simulations of the model proposed in this document. As in the previous section, where the analysis is performed with the geometrical and mechanical properties of Test 1, this section proceeds in the same way.
It was decided to set a mesh size $h^{mesh} = 0{,}005$ meters and an adaptive exponent (Papanastasiou parameter) $m = 1000$ seconds.
The first attempts at time step are totally inefficient for the calculation, as it is simply not possible for the computer to simulate anything since the time from one step to the next is so long.

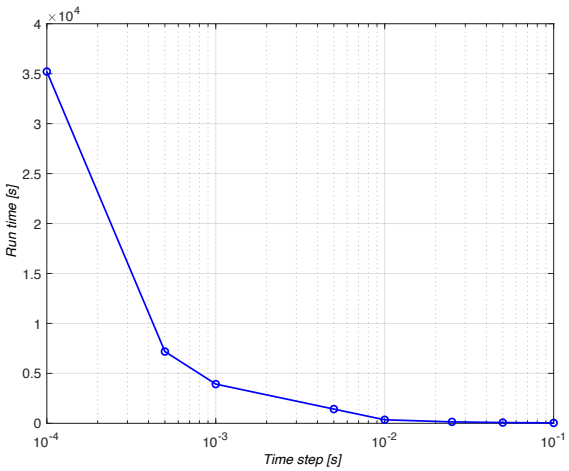


Figure 3.22: Comparison of running time with varying time steps

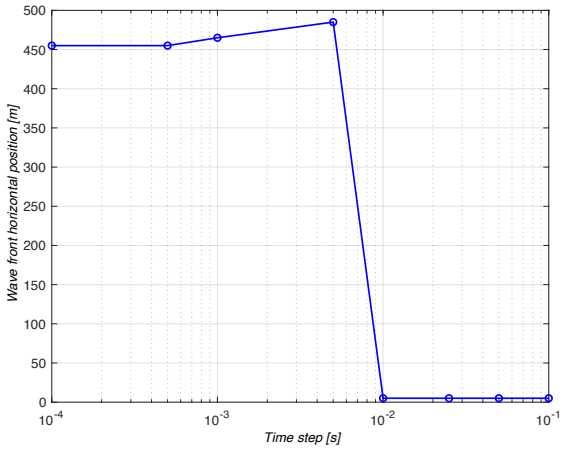


Figure 3.23: Comparison of the fluid front as a time step function

As with the determination of the mesh size to be used, the same applies to the time step analysis. The computational cost increases dramatically once the value of 0,01 seconds is exceeded. However, it has to be lower than this value due to its error when compared to other values.

Based on the results of this analysis, it was decided to use a time step of 0,001 seconds for the model validation tests, since it produces a relative error of less than 3%.

$$\Delta t^{step} = 0,001\ s$$

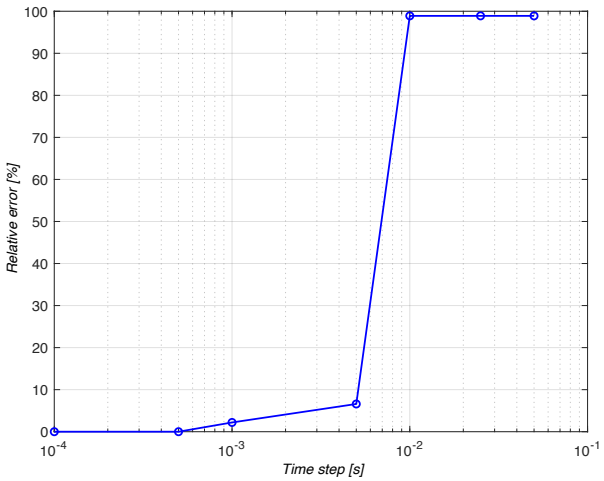


Figure 3.24: Relative error varying the time step

3.2.1.4. Papanastasiou parameter

Once the mesh size and the step time have been established, it is time to determine the adaptive parameter to be used in the numerical simulations Tests for the validation of the model of this writing.

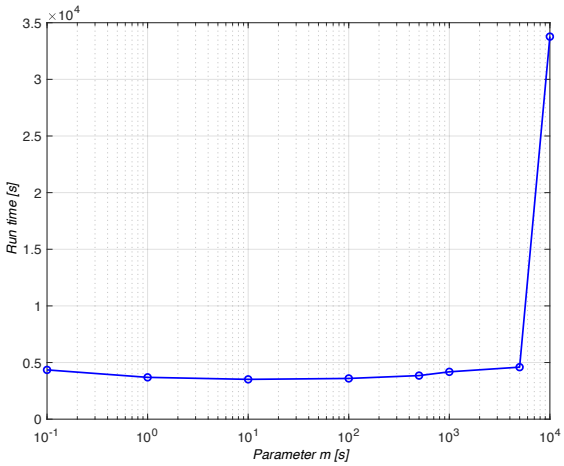


Figure 3.25: Comparison of running time with varying regularization parameter

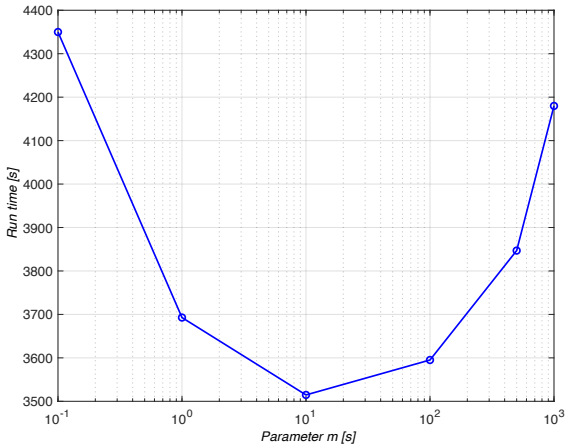


Figure 3.26: Expansion in the area of interest of the Figure 3.25

The two graphs immediately above this paragraph (Figure 3.25 and Figure 3.26) show the same result, however it has been decided to enlarge the starting area of the analysis in order to better appreciate the behavior of the simulations.

It is curious how from a parameter m = 10 and above, the relative error committed is less than 5%.

It is recalled that the higher the value of this parameter, the better it was able to recreate the Herschel-Bulkley law, shown at Figure 2.6.

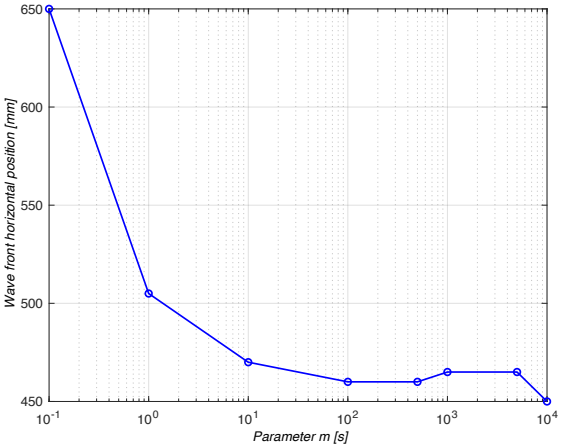


Figure 3.27: Comparison of the fluid front as a regularization parameter function

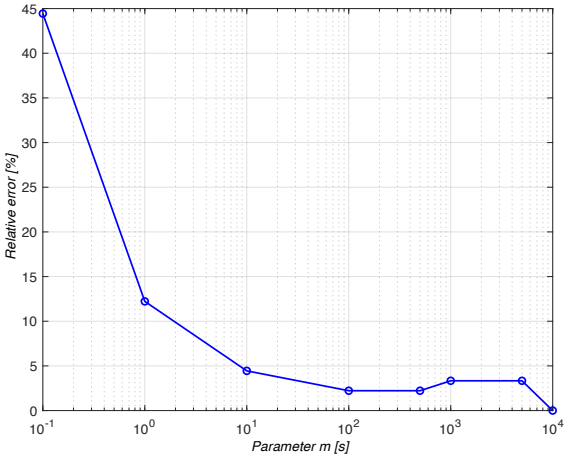


Figure 3.28: Relative error varying the regularization parameter

The simulation of the test was also attempted by imposing an $m = 100000$, however, the computer was unable to perform it. This is due to the presence of very high viscous terms in the governing equations for shear strain rates approaching to zero, thus close to the singularity of the Herschel-Bulkley model. In fact, for excessive high values of parameter "m", the regularization effect is reduced and the drawbacks of the standard Herschel-Bulkley model arise again.

Using high values of "m" and so, high viscous parameter, have a clear effect on the solution of the linear system. In fact, for high value of "m", the solution system may become ill-conditioned and so more difficult to be solved by the iterative linear solver. In other words, for high values of viscosity the solver may need many iterations to reach the exact solution of the linear system. This has a clear consequence on the duration of the analyses, because the solver will last more time to obtain the solution of each time step. In the next analysis, it is analyzed this effect by solving the first steps of the analysis for different values of "m".

The same analyses, but with a better visualization of the comparisons between one value and another, are as follows: Figure 3.29 is the comparison between the various values of the "m" parameter when visualizing the fluid motion and the Figure 3.30 analysis is the comparison of the computational cost required by 10 step times according to the Papanastasiou parameter.

It can be seen in the Figure 3.29, once again, that from m = 10 the fluid motion hardly varies.
The Figure 3.30 determines the computational cost and it can be clearly seen that this, for 10 time steps alone, more than doubles its cost to avoid making an error of less than 5%. If this is extrapolated to the time end, the cost increases exponentially, as seen in the previous parameter determination simulations.

The number of steps to be calculated is determined by (4.1), where both values are pre-set at the time of performing the simulations.

$$\frac{\boldsymbol{t_{end}}}{\boldsymbol{\Delta t^{step}}} = \#\,\boldsymbol{steps} \quad \rightarrow \quad \frac{1{,}5\ [s]}{0{,}005\ \left[\frac{s}{step}\right]} = 300\ steps \tag{4.1}$$

It was decided to continue the numerical calculations with a Papanastasiou parameter value of 1000 seconds since it is one of the cheapest, computationally speaking, compared to others and it respects the behavior of the Herschel-Bulkley law regarding the onset of motion and its yield stress.

$$\boldsymbol{m = 1000\ s}$$

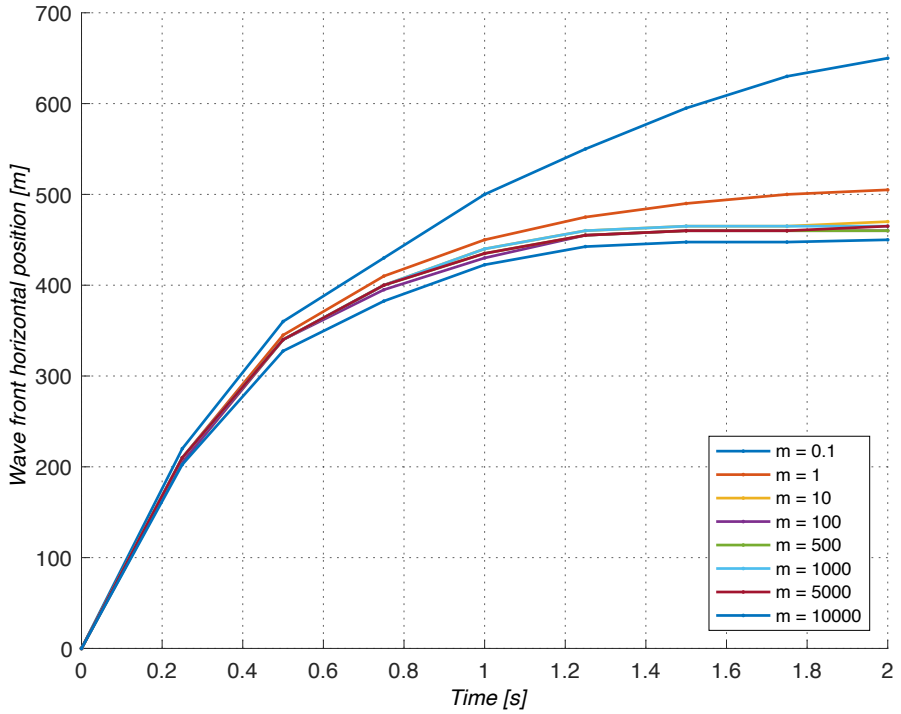


Figure 3.29: Wave front horizontal position varying the parameter "m"

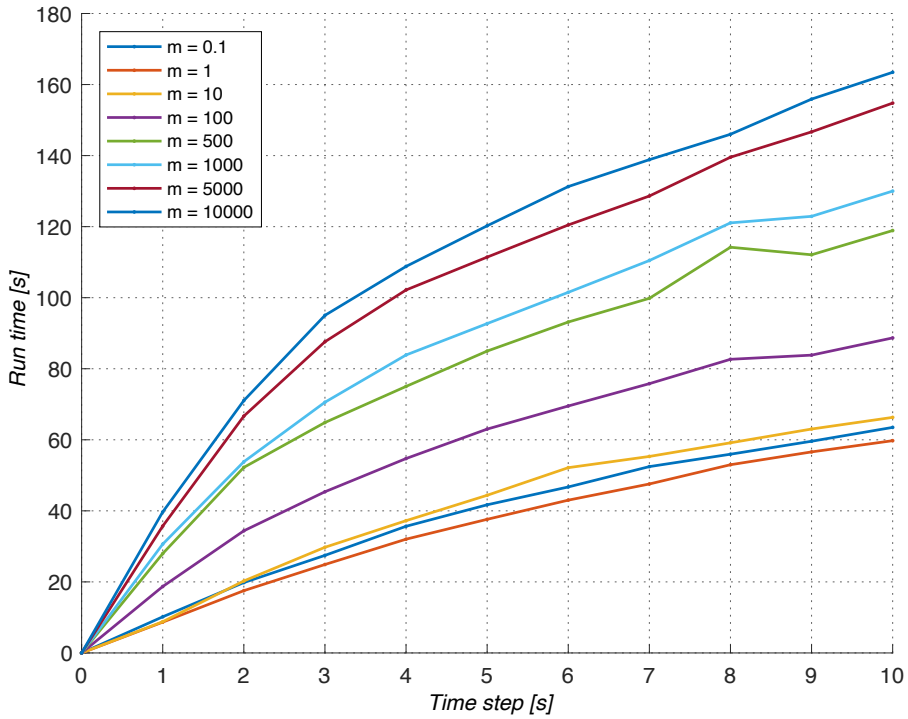


Figure 3.30: Run time after 10 seconds of Time step by varying the parameter "m"

3.2.1.5. Tests

Once the four extrinsic parameters of the Herschel-Bulkley (t_{end}, h^{mesh}, Δt^{step} and m) model have been determined, it is proceeded to its validation. It is now time to test different fluids with different geometry heights within the dam break example.

The validation is divided into 5 fluids of different mechanical properties and three possible starting heights for dam breakage (H=0.07, 0.10 or 0.13 m). The properties of the fluids are detailed in the attached tables.

3.2.1.5.1. Fluid 1

Equal conditions in terms of mechanical and geometrical properties except for the H value, shown in Table 3.3:

	$\boldsymbol{H}\ [m]$	$\rho\ \left[\frac{kg}{m^3}\right]$	$\kappa\ [Pa]$	$k[Pa \cdot s^n]$	$n\ [-]$	$\tau_o\ [Pa]$
Test 1	0,10	1000	$2{,}1 \cdot 10^9$	4,297	0,479	30,002
Test 2	0,13					

Table 3.3: Properties of fluid 1 in Herschel-Bulkley validation simulations

In this test it can be seen that the fact that the height of the fluid in the simulation greatly influences the final results compared to the numerical reference from which the validation examples were taken. The higher the starting height of the fluid, the smaller the difference at the end of the numerical calculation with the reference values (with a final time t_{end} = 1,5 seconds).

The Figure 3.31 shows the advance of the horizontal wave front on the ordinate axis, measured in millimeters, and on the abscissa axis the time advance measured in seconds, where it is limited to 1.5 seconds, is simulated by what was analyzed in section 3.2.1.1. Four types of curves are plotted: the coordinates of the black (numerical reference) and red curves (experimental reference) are taken from [21], while the light blue (Bingham model) and deep blue (Herschel-Bulkley model) coordinates are obtained from simulations using PFEM implemented in Kratos. This figure description is also valid for the: Figure 3.33, Figure 3.35, Figure 3.37, Figure 3.39, Figure 3.45, Figure 3.47, Figure 3.49, Figure 3.55, Figure 3.57, Figure 3.59, Figure 3.61 and Figure 3.63.

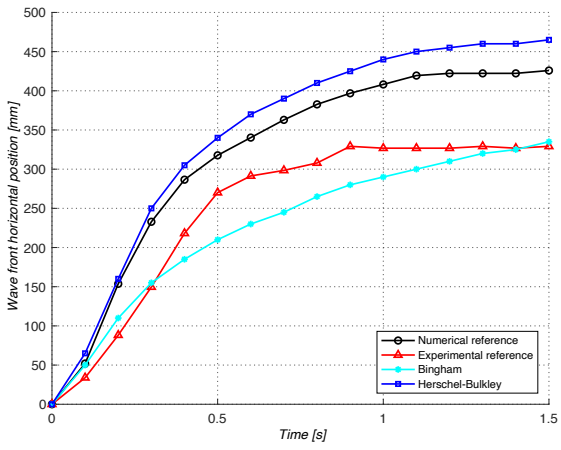


Figure 3.31: Test 1 results

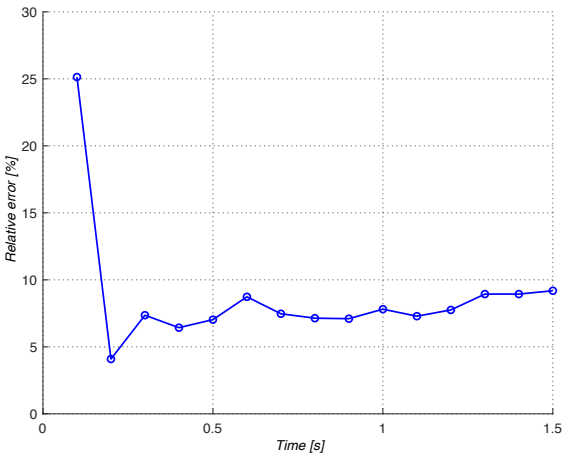


Figure 3.32: Variation of results between Test 1 and reference

The Figure 3.32 shows the progress in time (abscissa axis) of the variation of the results obtained from the PFEM simulation and the numerical reference values. This figure description is also valid for the: Figure 3.34, Figure 3.36, Figure 3.38, Figure 3.40, Figure 3.46, Figure 3.48, Figure 3.50, Figure 3.56, Figure 3.58, Figure 3.60, Figure 3.62 and Figure 3.64.

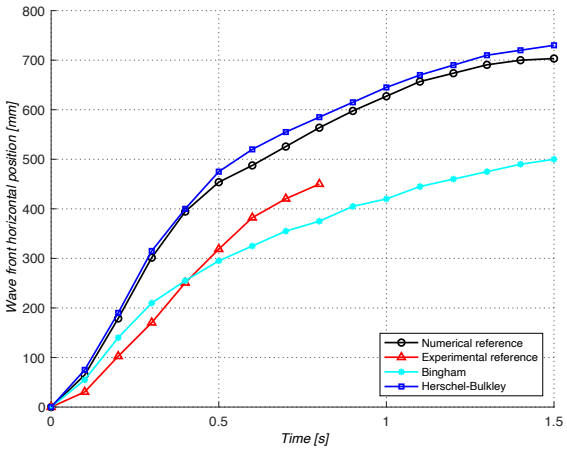


Figure 3.33: Test 2 results

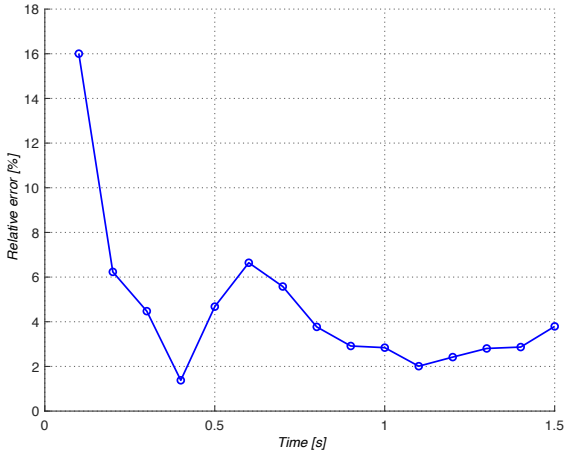


Figure 3.34: Variation of results between Test 2 and reference

In Test 1, the variation with respect to the numerical reference, at the end of the calculation, is less than 10%. However, in Test 2, where the height is greater (0,13 meters), the final variation of the simulation is less than 4%.

In both cases, the initial behavior of the movement is similar to the reference values, and it is only after the displacement has elapsed that the variation begins to increase.

3.2.1.5.2. Fluid 2

Equal conditions in terms of mechanical and geometrical properties except for the H value, shown in Table 3.4.

	$H\ [m]$	$\rho\ \left[\frac{kg}{m^3}\right]$	$\kappa\ [Pa]$	$k\ [Pa \cdot s^n]$	$n\ [-]$	$\tau_o\ [Pa]$
Test 3	0,07	1000	$2{,}1 \cdot 10^9$	1,904	0,531	18,242
Test 4	0,10					
Test 5	0,13					

Table 3.4: Properties of fluid 2 in Herschel-Bulkley validation simulations

In the simulation of this fluid, there is, in the first test (Test 3), an important difference of movement. Again, this large variation in the final results occurs in the geometry of lower initial fluid height. However, there are very favorable results in the other two tests (4 and 5).

In test 3, variations of an order of magnitude greater than 15% are shown. However, and even being the same fluid with the same mechanical properties, in test 4 and 5 the variations decrease to less than 6% and 1%, respectively, being a very close fit to the references. The greater the initial height, the smaller the final difference, which leads to a very similar representation of the numerical method proposed in this work and the reference method.

In an attempt to find out what has happened in Test 3, it was simulated again by decreasing both the time step value and the mesh size, refining it (Figure 3.41 and Figure 3.42).

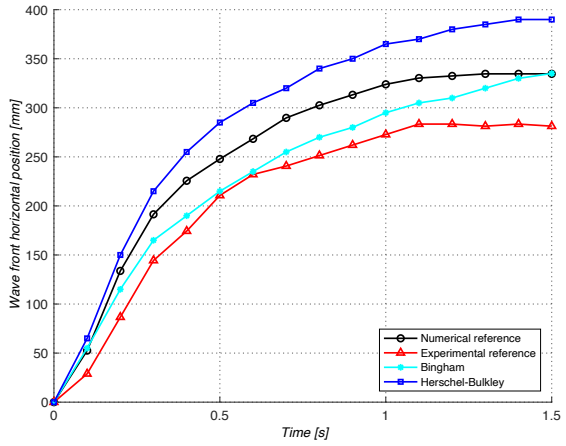


Figure 3.35: Test 3 results

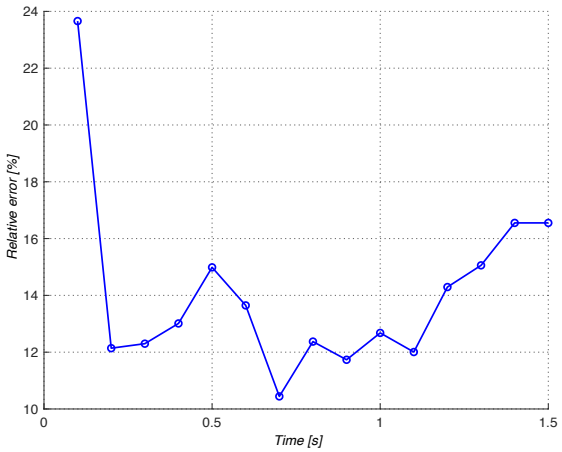


Figure 3.36: Variation of results between Test 3 and reference

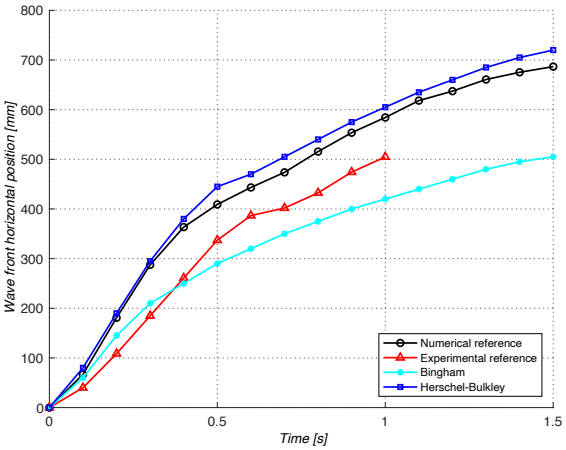


Figure 3.37: Test 4 results

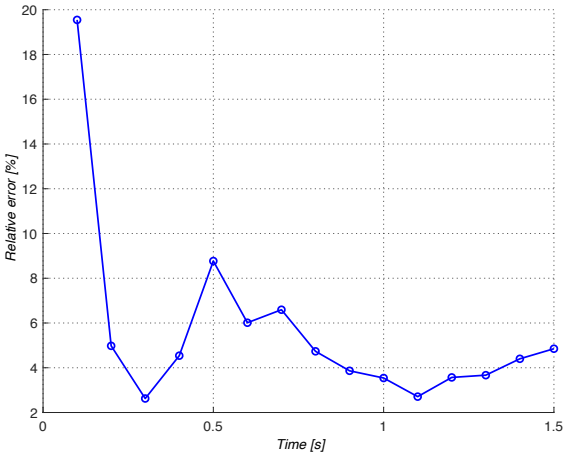


Figure 3.38: Variation of results between Test 4 and reference

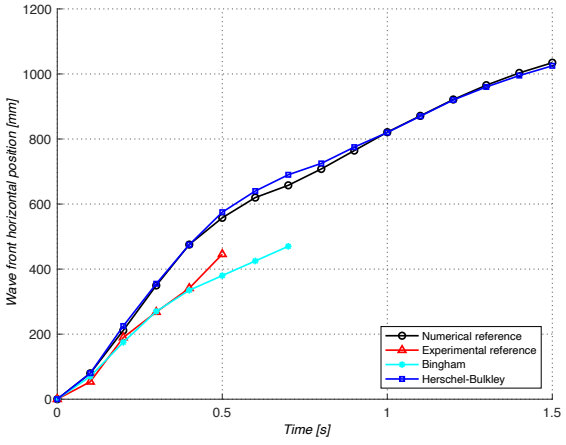


Figure 3.39: Test 5 results

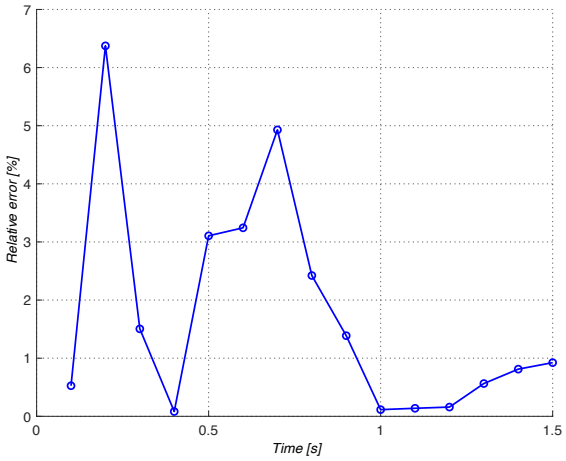


Figure 3.40: Variation of results between Test 5 and reference

The Table 3.5 shows the modifications that have been made in the simulations and the computational cost of these changes. The fact of varying the mesh size greatly penalizes the numerical calculation.

	$\Delta t^{step}\ [s]$	$h^{mesh}\ [m]$	Computational cost $[s]$
Test 3	0,001	0,005	2007,88
Test 3a	0,0005	0,005	3084,07
Test 3b	0,001	0,0025	8120,46
Test 3c	0,0005	0,0025	-

Table 3.5: Redefinition of parameters in Test 3

It can be seen that having refined the mesh (decreased its size) and/or changed the time step, the variation with respect to the reference is hardly affected.

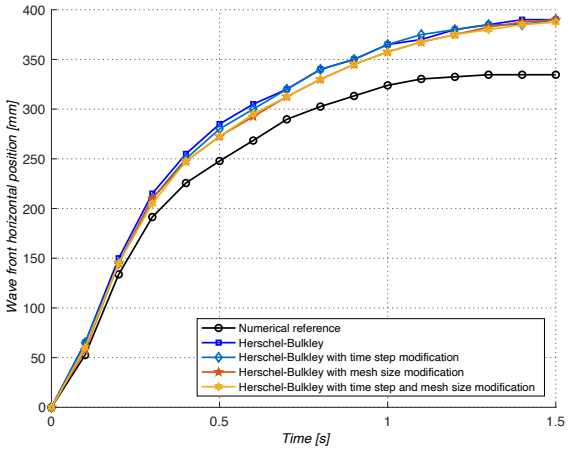


Figure 3.41: Comparison of the same Test 3 varying properties

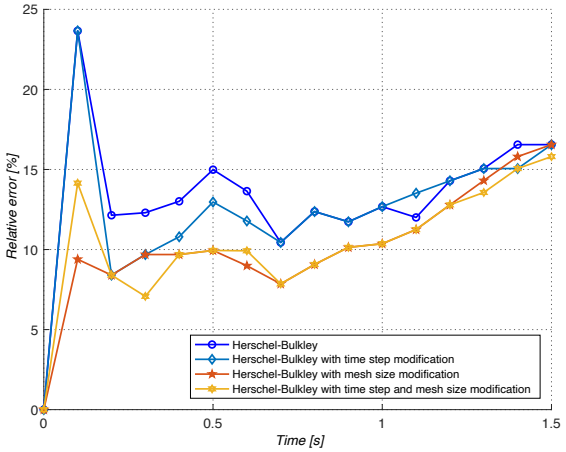


Figure 3.42: Variations of Test 3 simulations with respect to the numerical reference

However, and this is the case in most tests, varying the alpha shape of the simulation parameters decreases the relative error by a little more than 1%. Thus, it can also be said that it does not affect the results. This is shown in the following Figure 3.43 and Figure 3.44:

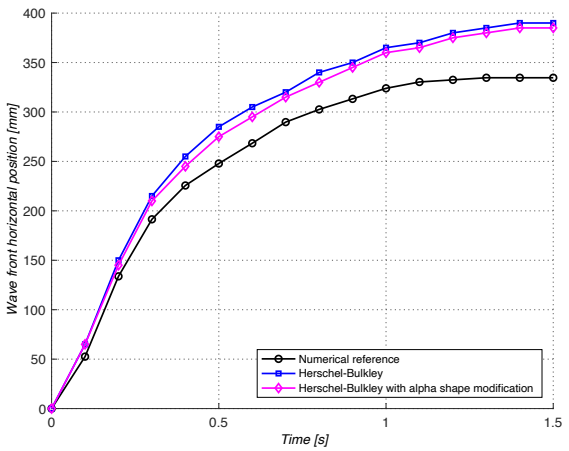


Figure 3.43: Test 3 results by varying the alpha shape

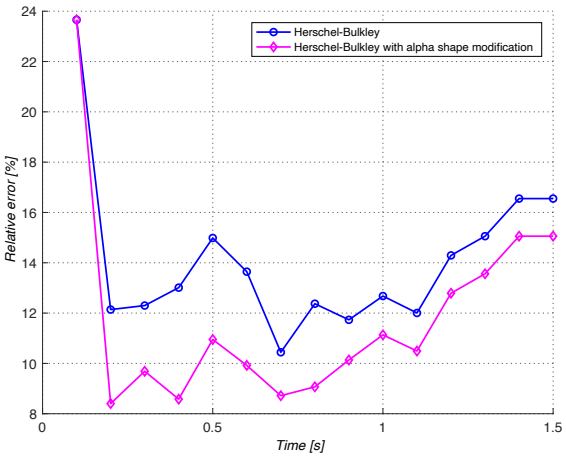


Figure 3.44: Variations of Test 3 with respect to the references by varying the alpha shape

3.2.1.5.3. Fluid 3

Equal conditions in terms of mechanical and geometrical properties except for the H value, shown in Table 3.6.

	$H\ [m]$	$\rho\ \left[\frac{kg}{m^3}\right]$	$\kappa\ [Pa]$	$k\ [Pa \cdot s^n]$	$n\ [-]$	$\tau_o\ [Pa]$
Test 6	0,07	1000	$2{,}1 \cdot 10^9$	7,837	0,442	49,179
Test 7	0,10					
Test 8	0,13					

Table 3.6: Properties of fluid 3 in Herschel-Bulkley validation simulations

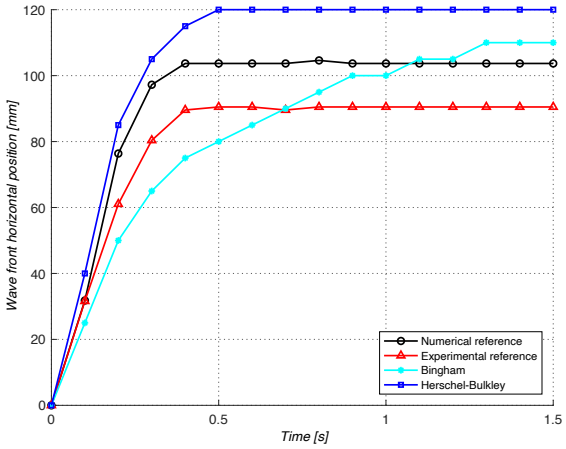


Figure 3.45: Test 6 results

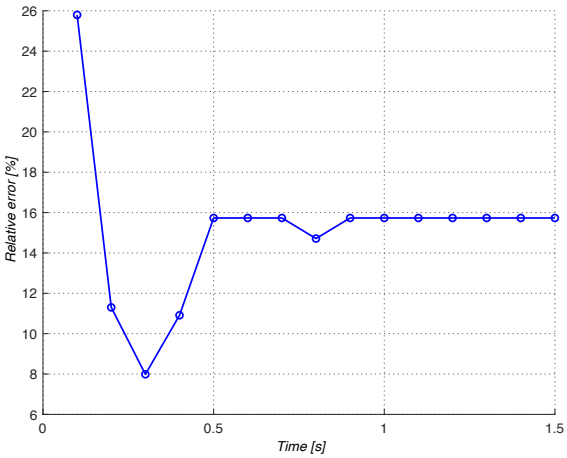


Figure 3.46: Variation of results between Test 6 and reference

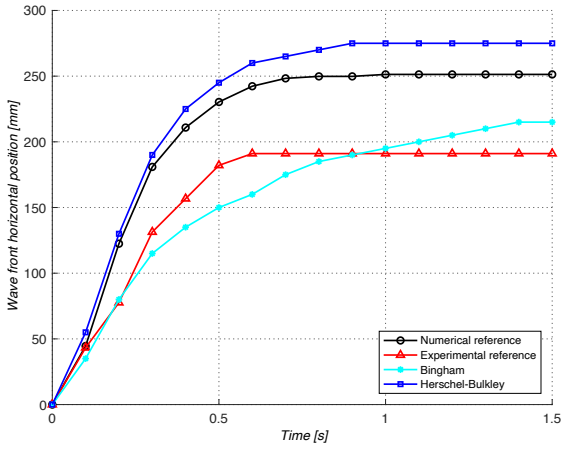


Figure 3.47: Test 7 results

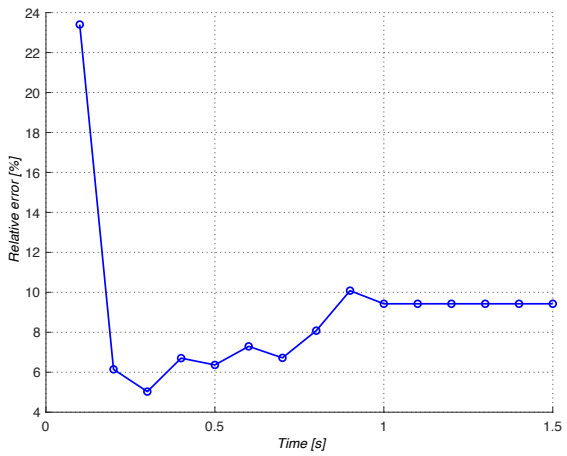


Figure 3.48: Variation of results between Test 7 and reference

In these tests it can be seen that the fluid stops its movement much earlier than the rest due to the higher yield stress value. It is also observed that the smaller the fluid height, the greater the difference in fluid motion front results between the simulations (including the reference ones) and the laboratory experimental values.

In general, it is observed that the PFEM analyses overestimate the numerical and experimental results of this test. In section 4.2.2. some of these tests are studied in three dimensions, to analyze the effect that

can be produced, for example, by the friction of the side walls at the final location of the fluid front after the study time has elapsed.

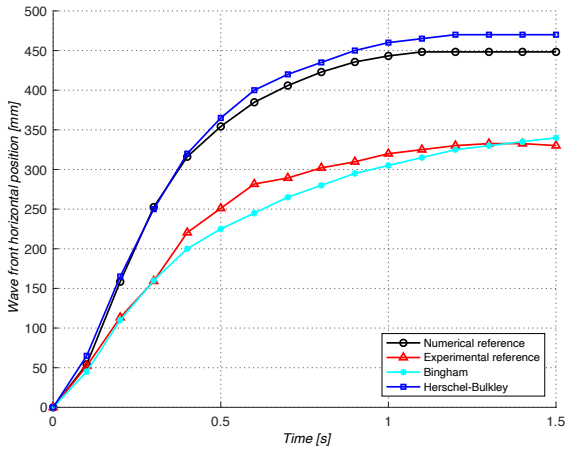


Figure 3.49: Test 8 results

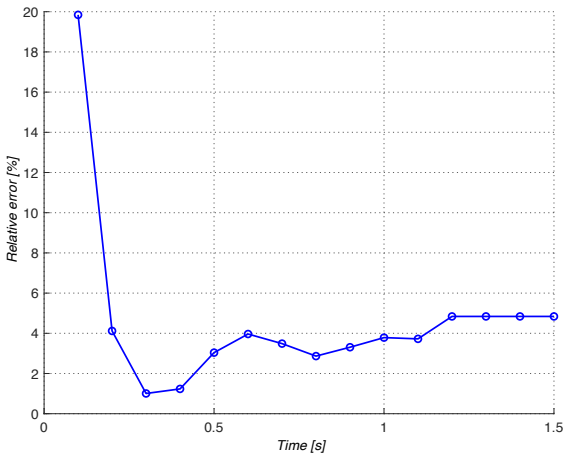


Figure 3.50: Variation of results between Test 8 and reference

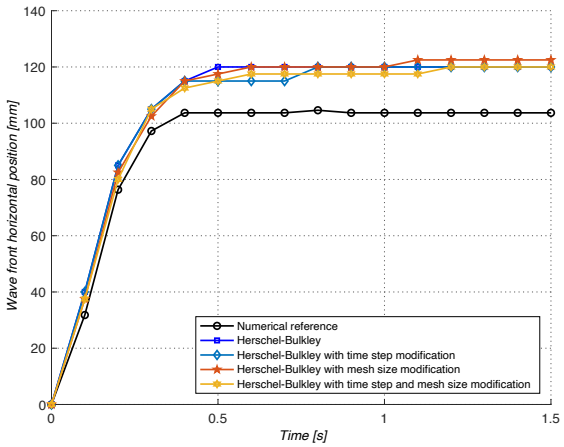


Figure 3.51: Comparison of the same Test 6 varying properties

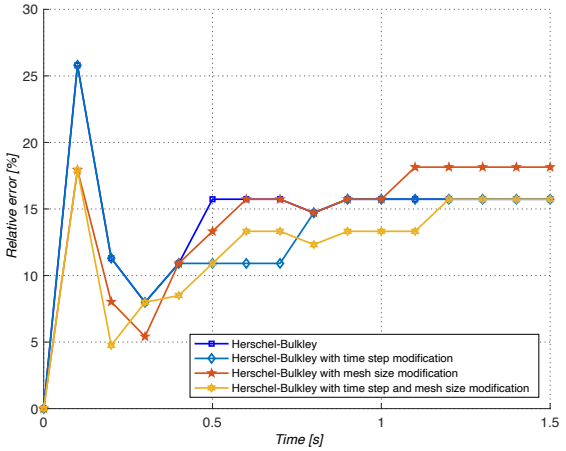


Figure 3.52: Variations of Test 6 simulations with respect to the numerical reference

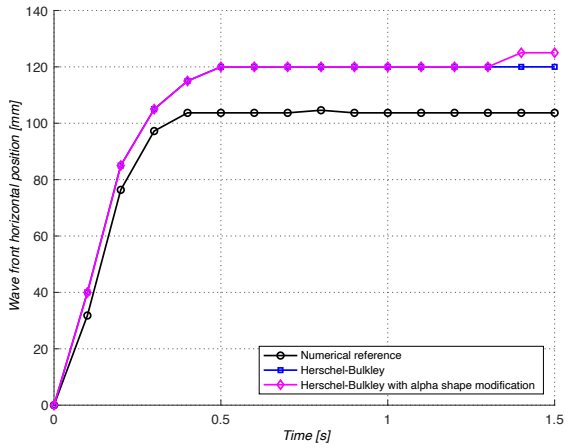


Figure 3.53: Test 6 results by varying the alpha shape

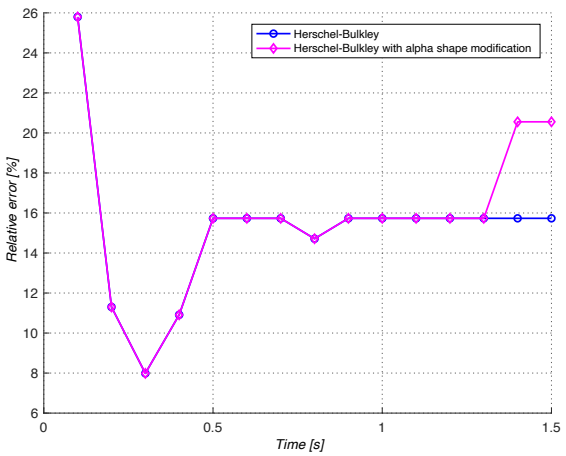


Figure 3.54: Variations of Test 6 with respect to the references by varying the alpha shape

In an attempt to find out what has happened in Test 6, it was simulated again by decreasing both the time step value and the mesh size, refining it (Figure 3.51 and Figure 3.53). This is the same as in the re-simulation of Test 3, there are hardly any variations with respect to the reference simulation.

3.2.1.5.4. Fluid 4

Equal conditions in terms of mechanical and geometrical properties except for the H value, shown in Table 3.7.

	$H\ [m]$	$\rho\ \left[\frac{kg}{m^3}\right]$	$\kappa\ [Pa]$	$k\ [Pa \cdot s^n]$	$n\ [-]$	$\tau_o\ [Pa]$
Test 9	0,1	1000	$2{,}1 \cdot 10^9$	4,55	0,470	31,366
Test 10	0,13					

Table 3.7: Properties of fluid 4 in Herschel-Bulkley validation simulations

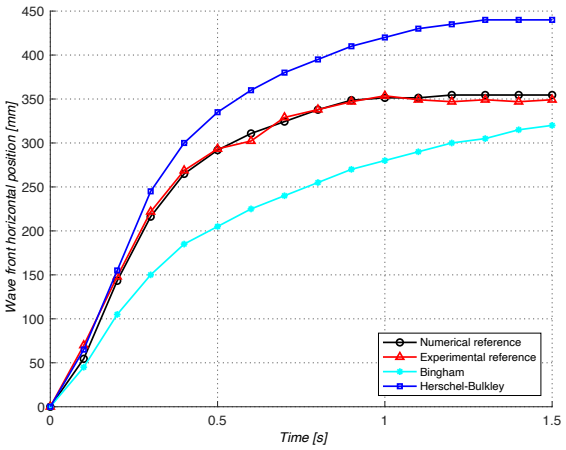


Figure 3.55: Test 9 results

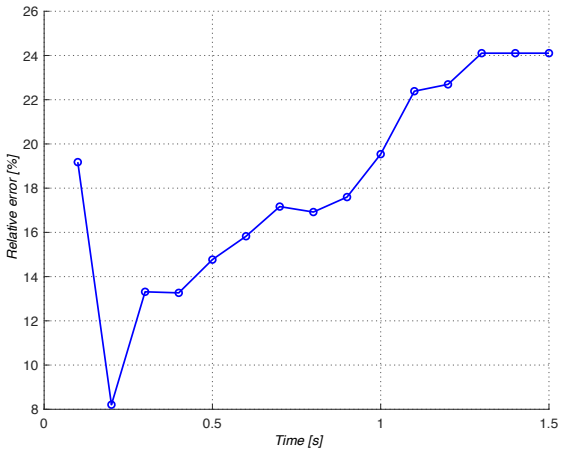


Figure 3.56: Variation of results between Test 9 and reference

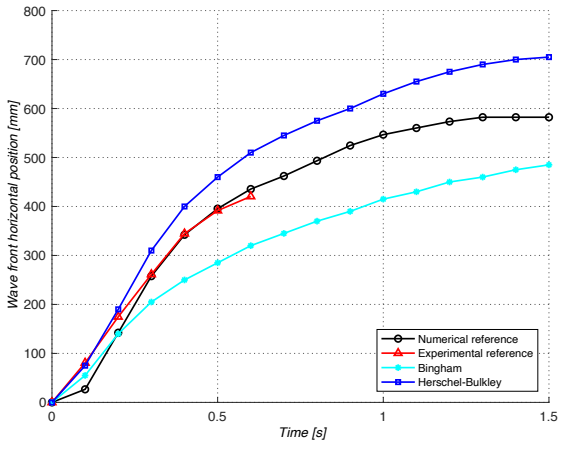


Figure 3.57: Test 10 results

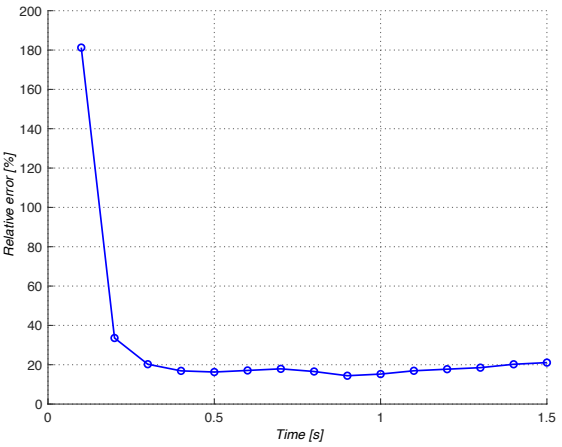


Figure 3.58: Variation of results between Test 10 and reference

The two numerical references are quite close to the results obtained in the laboratory; however, the simulations recreated with PFEM overestimate these results. There are even variations in the data of more than 20%.

3.2.1.5.5. Fluid 5

Equal conditions in terms of mechanical and geometrical properties except for the H value, shown in Table 3.8.

	$H\ [m]$	$\rho\ \left[\frac{kg}{m^3}\right]$	$\kappa\ [Pa]$	$k\ [Pa \cdot s^n]$	$n\ [-]$	$\tau_o\ [Pa]$
Test 11	0,07	1000	$2{,}1 \cdot 10^9$	7,263	0,446	39,170
Test 12	0,10					
Test 13	0,13					

Table 3.8: Properties of fluid 5 in Herschel-Bulkley validation simulations

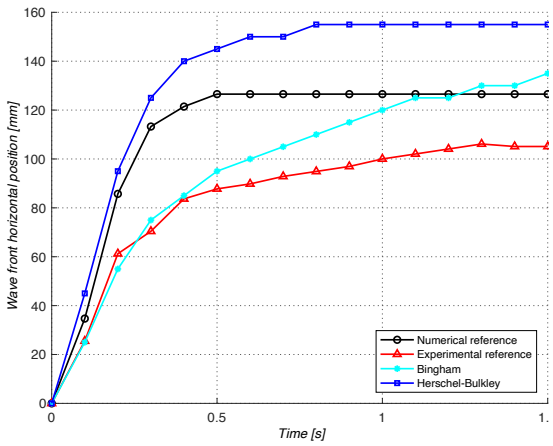


Figure 3.59: Test 11 results

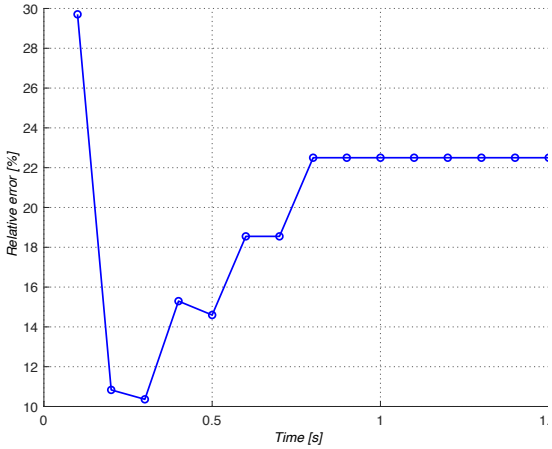


Figure 3.60: Variation of results between Test 11 and reference

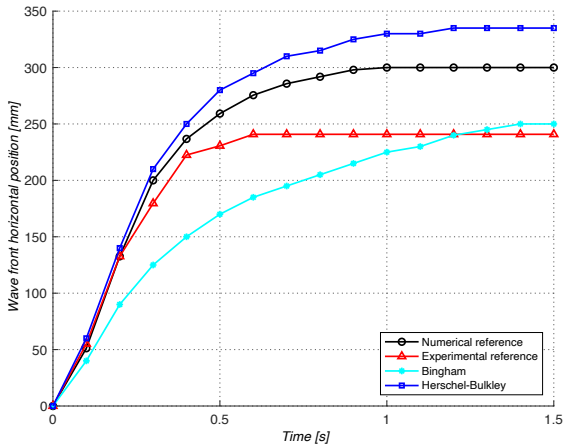


Figure 3.61: Test 12 results

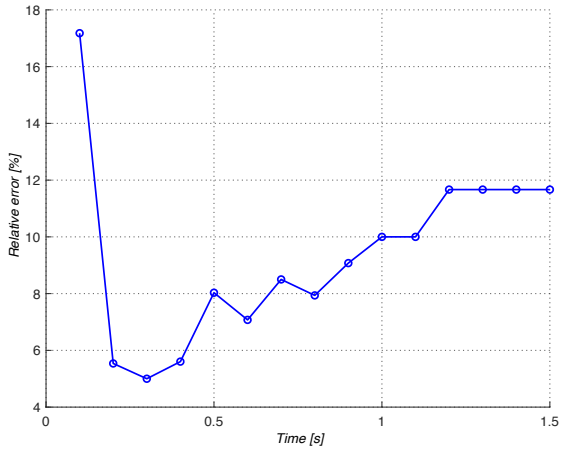


Figure 3.62: Variation of results between Test 12 and reference

In these simulations, something similar happens again as in the case of fluid 2. For a height of 0.07 meters, the results deviate by more than 22% from the numerical reference data. However, for greater heights, the variations with respect to the reference are decreasing. But, contrary to fluid 2, the results still show some differences in the results.

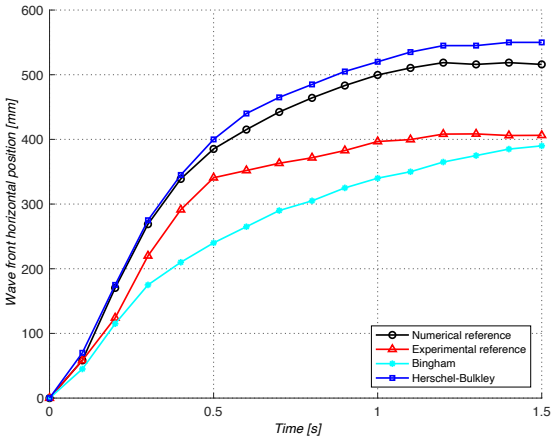


Figure 3.63: Test 13 results

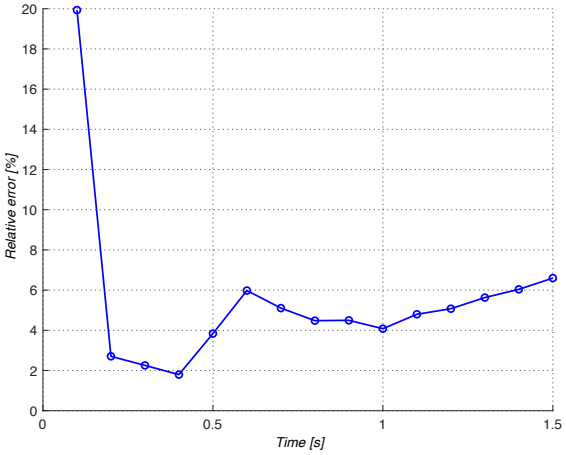


Figure 3.64: Variation of results between Test 13 and reference

3.2.2. Results in 2D

The following figures show the results obtained at the end of the simulation of Test 1 (fluid 1). Both Figure 3.65 and Figure 3.66 show variables result iteratively in the numerical calculation.

The following Figure 3.67, Figure 3.68, Figure 3.69 and Figure 3.70 show an approximate comparison of the variation of movement, both in the experimental reference (with grid), in the numerical reference (black) and in the simulation proposed by this work (blue).

The movement and, therefore, the speed of the system, is produced, above all, on the right side, just where the dam break occurs. This behavior can be better visualized in the Figure 3.71 where the velocity vector field is shown according to the fluid region of Test 5.

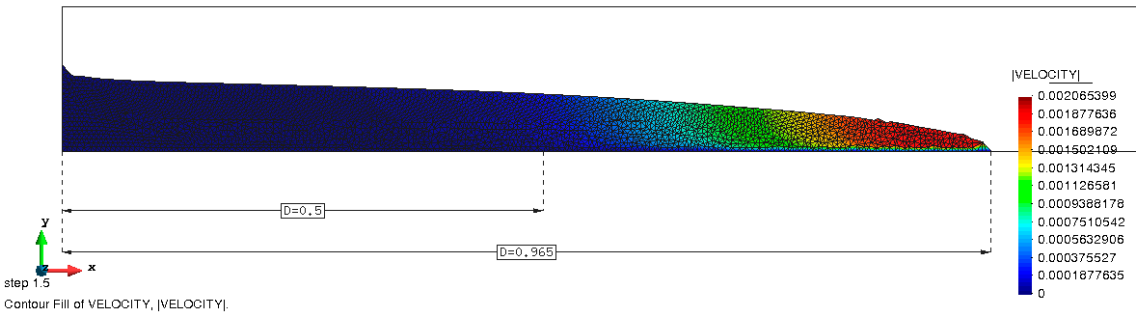


Figure 3.65: Velocity field of Test 1 at time step 1.5 s

The pressure presented by the simulation is somewhat lower than the hydrostatic pressure (the density of the calculated fluids is the same as that of water as well).

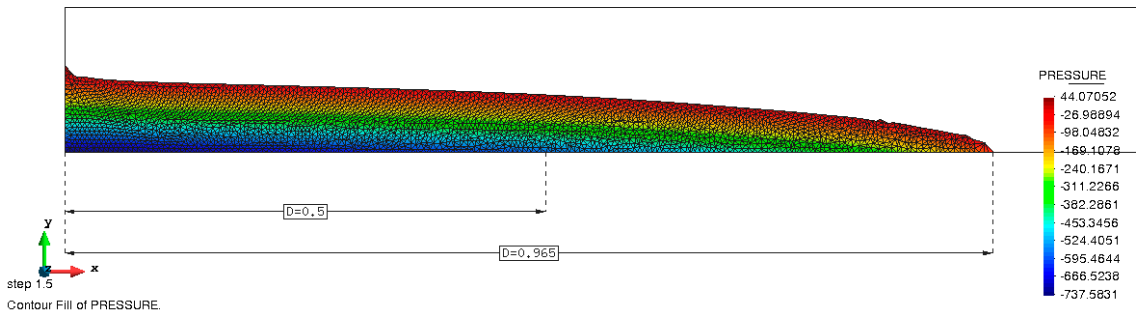


Figure 3.66: Pressure field of Test 1 at time step 1.5 s

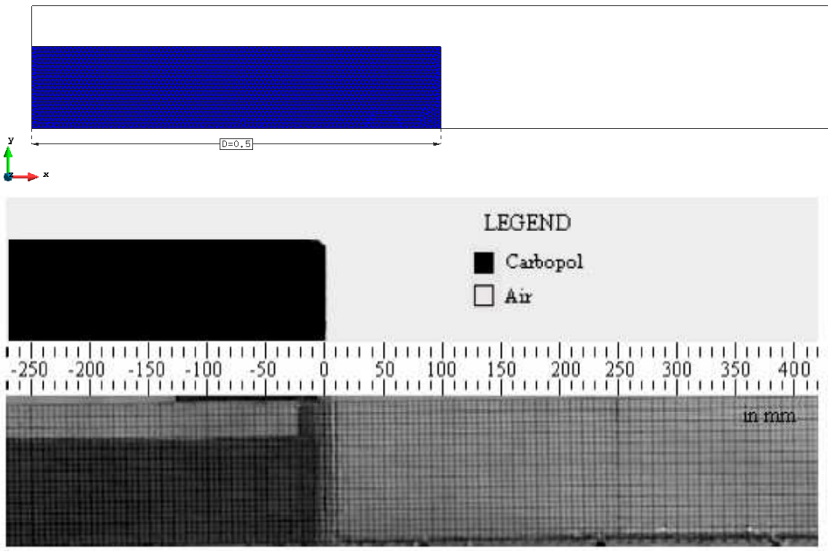


Figure 3.67: Comparison at t = 0s between experimental, numerical [21] and own references

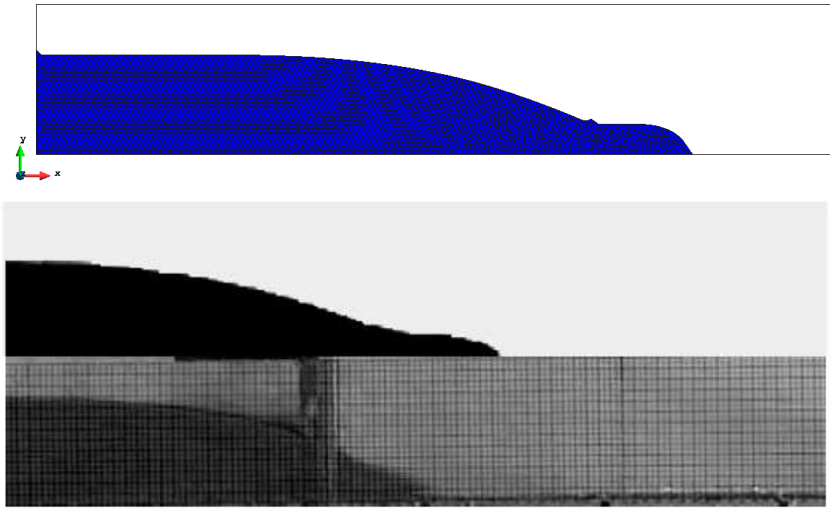


Figure 3.68: Comparison at t = 0.2s between experimental, numerical [21] and own references.

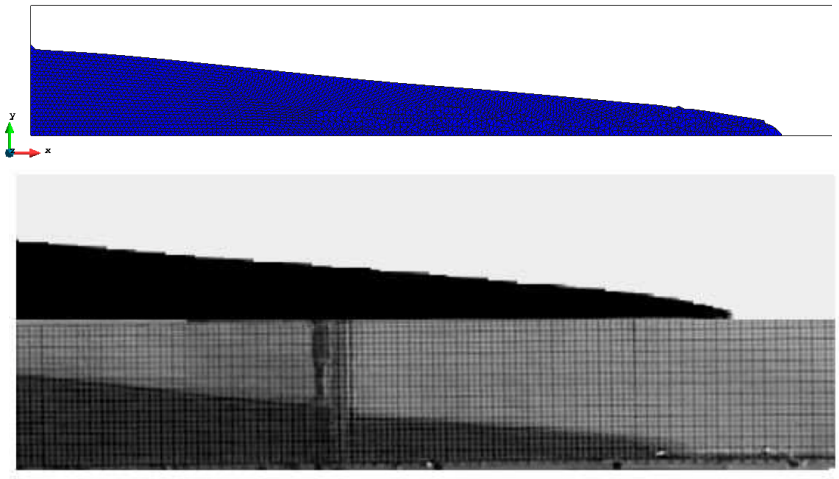


Figure 3.69: Comparison at t = 0.6s between experimental, numerical [21] and own references.

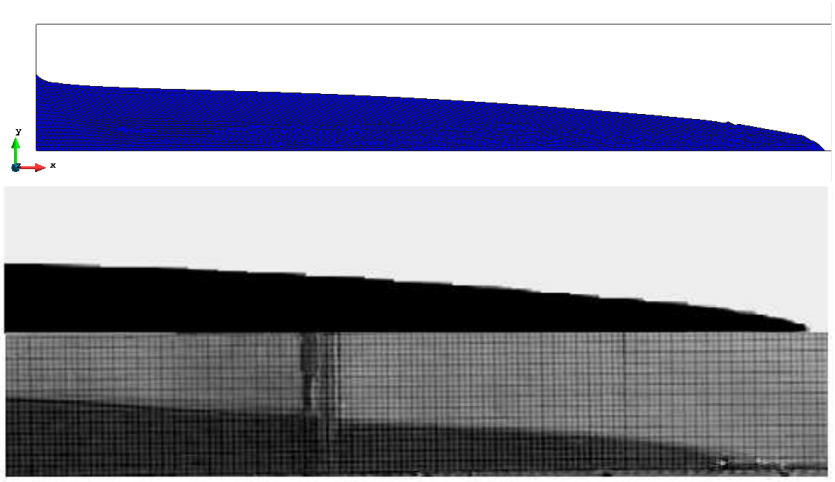


Figure 3.70: Comparison at t = 1s between experimental, numerical [21] and own references.

It is observed that the flow represented by both the references and the calculations themselves are close to each other and even to the experimental data.

It can be seen in the evolution of the displacement as the initial peak is preserved but progressively decreasing its fluid size. This may be due to the consistency it presents.

In this occasion, modeling the fluids with Herschel-Bulkley, the sticking on the fluid wall follows quite closely the behavior of the numerical reference and even the experimental one, being this opposite in the Bingham modeling.

In [21], it is provided several results, of which are added in this work for comparison. In Figure 3.71 the vector field of several initial stages of Test 5 is shown. In Figure 3.72 the same type of field can be observed but with the simulations performed with PFEM.

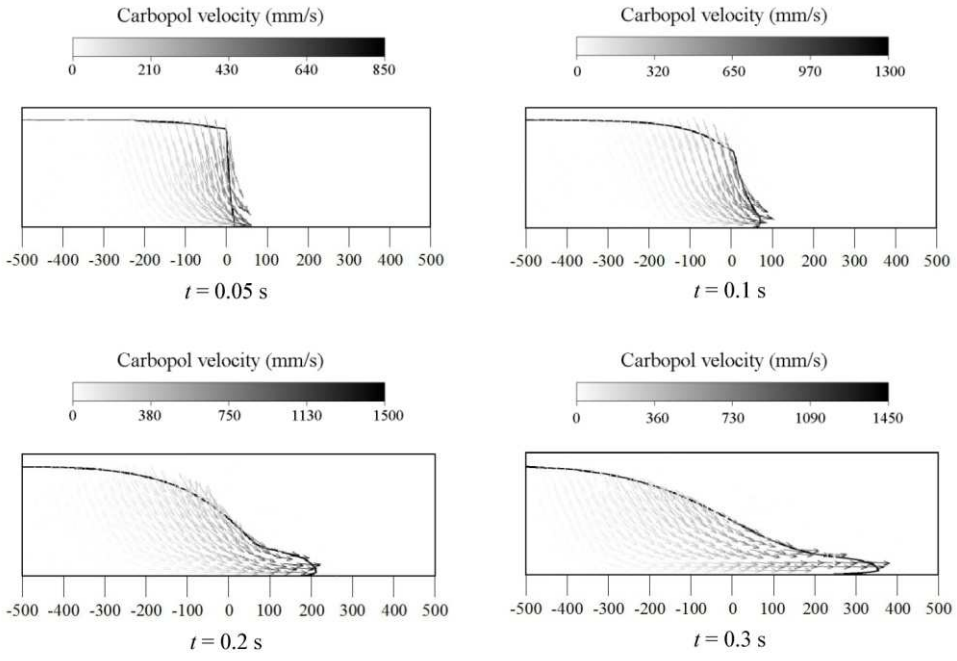


Figure 3.71: Test 5 reference velocity vectors field [21]

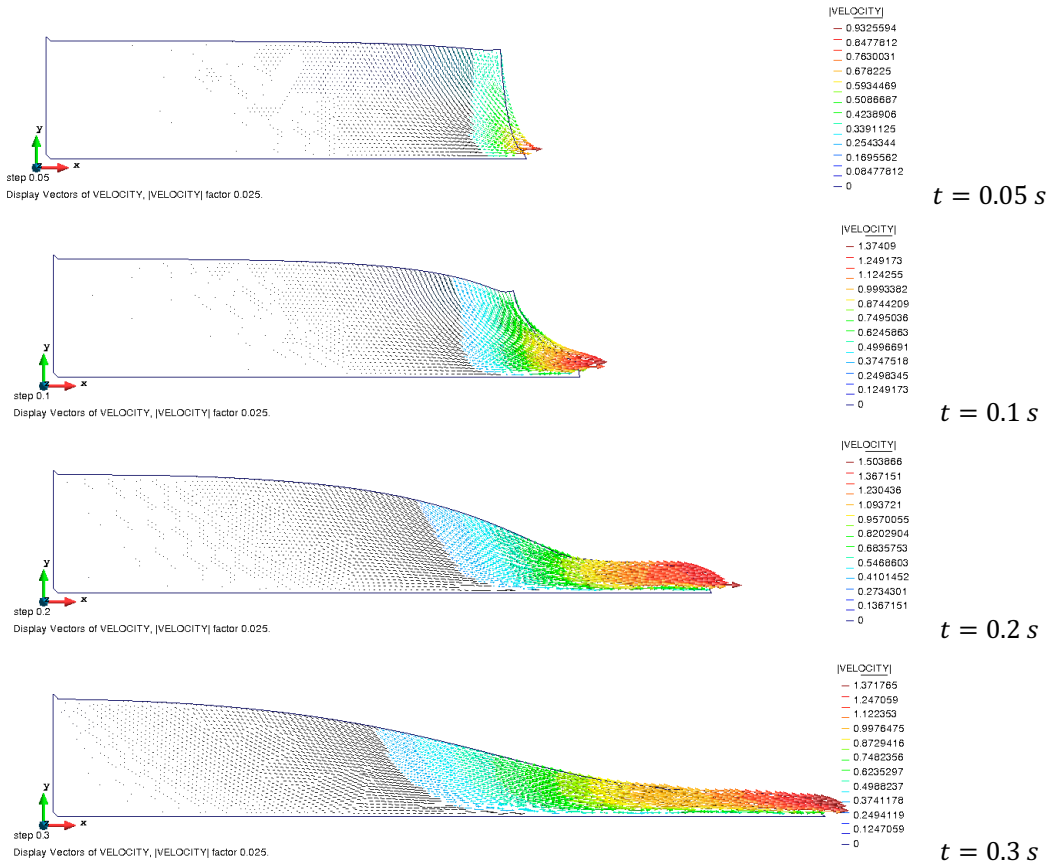


Figure 3.72: Velocity vector field of Test 5 performed with PFEM

The pressures in the initial stages of Test 5, both reference (Figure 3.73) and simulated (Figure 3.74), are also added.

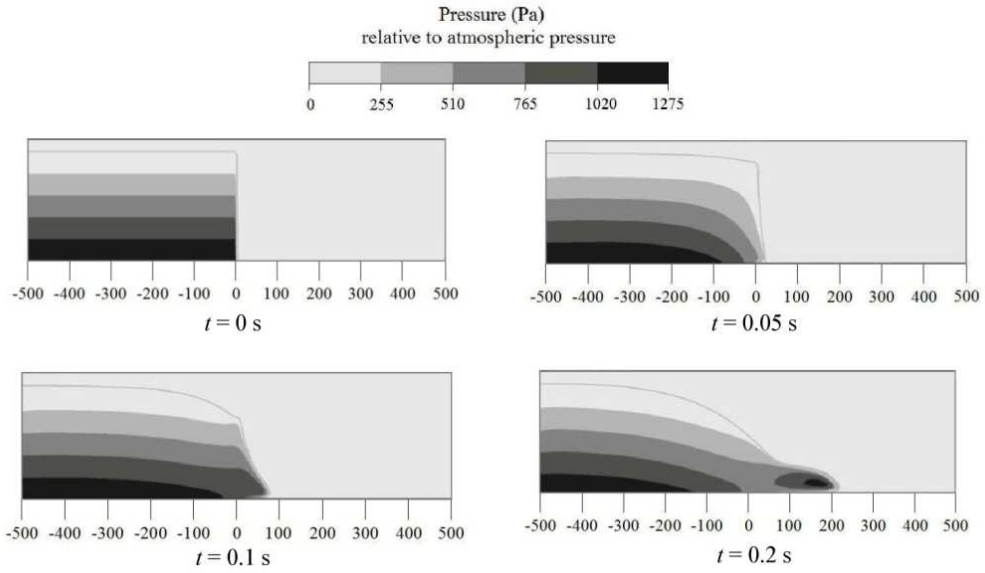


Figure 3.73: Pressures of Test 5 (validation reference) [21]

Both the lower and upper values of the two simulations are remarkably close. There are hardly any differences in values. However, there are some differences in the spatial distribution of these calculations.

These variations are present at the fluid flow front. In the numerical reference they show more pressure inside the front than in the PFEM simulations.

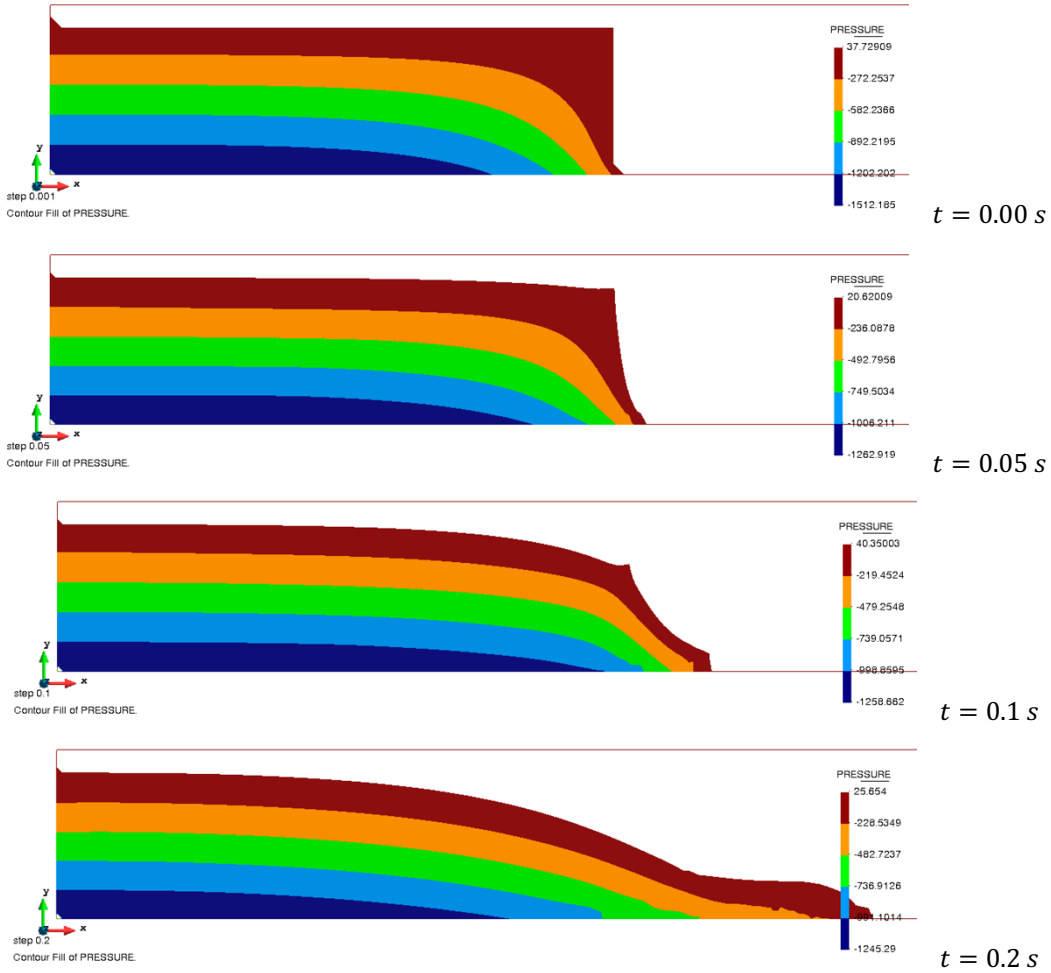


Figure 3.74: Test 5 pressures results obtained in the simulation with PFEM

3.2.3. Three dimensions validation

All three-dimensional test simulations were performed using a computer cluster provided by CIMNE. This is due to the high computational load of the 3D resolution of the validation examples. A cluster is a set of computers that work together so that they can be interpreted as a single system.

Tests 6, 7, 11 and 12 were chosen to be simulated because their movement slowed down quickly and they stopped soon after the simulation (due to the properties of the fluid). This would reduce the computational cost of each of the simulations. They are also tests with a relevant difference between the simulation and the reference values.

Attempts of 0.01, 0.005, 0.004 meters in the size of the medium discretization mesh were studied using the cluster. Smaller mesh size could not be analyzed due to the excessive computational cost of the analyses.

3.2.3.1. Test 6

The Figure 3.75 shows, once again, the movement of the horizontal front of the fluid measured in millimeters (ordinate axis) while the abscissa axis represents the advance in time measured in seconds of the calculated simulation. Within it several curves are plotted: the advance of the front according to both Bingham and Herschel-Bulkley models in two and three dimensions while keeping plotting the numerical reference data for further comparison. This description is also valid for the Figure 3.77, Figure 3.79 and Figure 3.81.

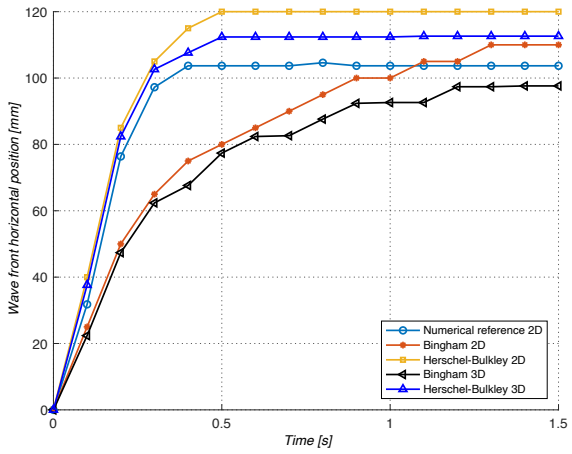


Figure 3.75: Test 6 results in 3D

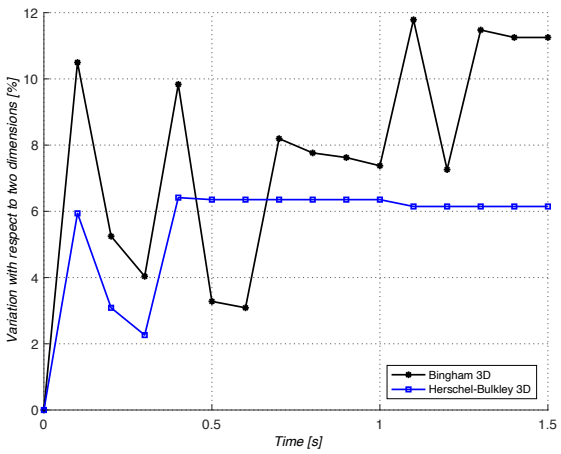


Figure 3.76: Variation of results between Test 6 in 3D and experimental reference

The variations between the simulations in two and three dimensions of both models (Herschel-Bulkley and Bingham) are shown in Figure 3.76 in order to see the influence of adding the third dimension in the resolution of the numerical calculation. This description is also valid for the Figure 3.78, Figure 3.80 and Figure 3.82

It goes without saying that the addition of an extra dimension generates a much higher computational cost. These were simulations where the calculation took more than 6 days per test performed.

In this test it can be seen how the extra dimension influences the slowing down of the movement and, as a consequence, the stopping of the movement earlier than in the two-dimensional case. In the case of Bingham modeling a similar behavior occurs. This may be due to the friction of the side walls, which causes

a generalized drag on the whole front of the movement (similar to what happens in a flow through a pipe). Even so, it does not quite fit the numerical contrast reference nor the experimental data, remaining above the values.

3.2.3.2. Test 7

This test represents the data obtained from the numerical simulation of Test 7. It can be seen that, as in the calculation of Test 6, the three-dimensional Bingham model offers values below the two-dimensional case. However, in the Herschel-Bulkley case the same thing happens but with a notable difference: the results are below the values of the numerical reference.

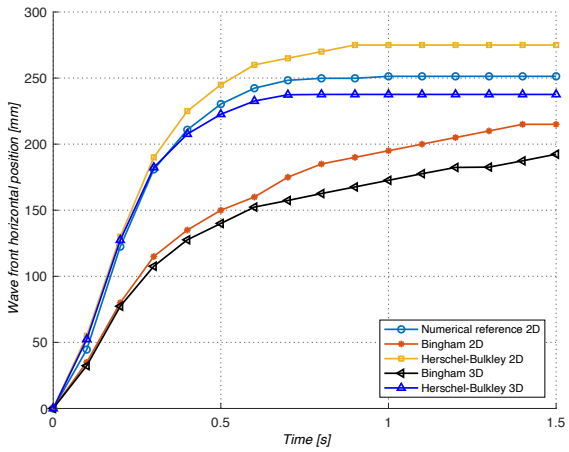


Figure 3.77: Test 7 results in 3D

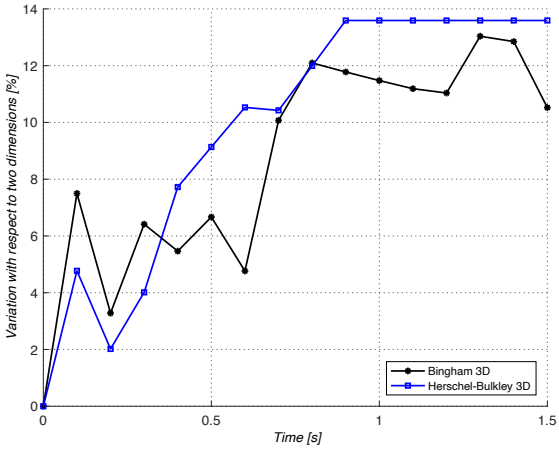


Figure 3.78: Variation of results between Test 7 in 3D and experimental reference

Figure 3.78 shows variations of more than 13% from the two-dimensional to the three-dimensional step.

3.2.3.3. Test 11

The same behavior occurs in this test as in Test 6. The movement is slowed down but the results are above the reference data.

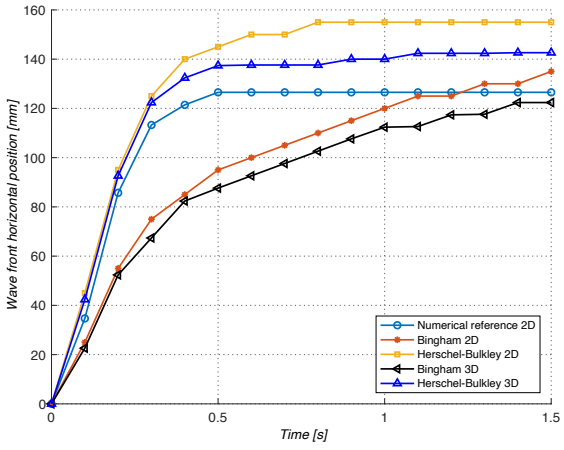


Figure 3.79: Test 11 results in 3D

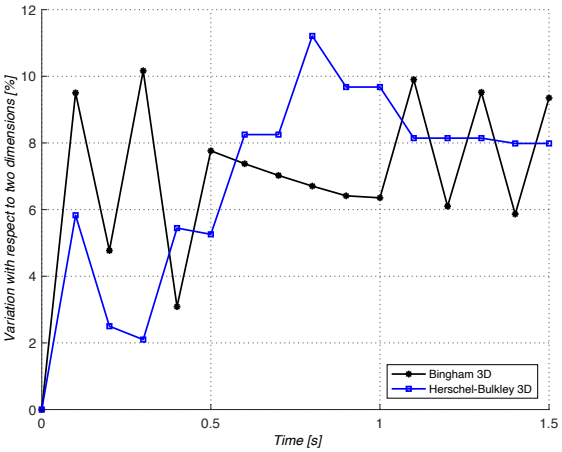


Figure 3.80: Variation of results between Test 11 in 3D and experimental reference

3.2.3.4. Test 12

The results of Test 12 show the same type of behavior as in the three-dimensional case of the results of Test 7.

The Herschel-Bulkley model returns lower results than the numerical reference data. The Bingham model, once again, follows the same three-dimensional behavior as in the previous 3 tests.

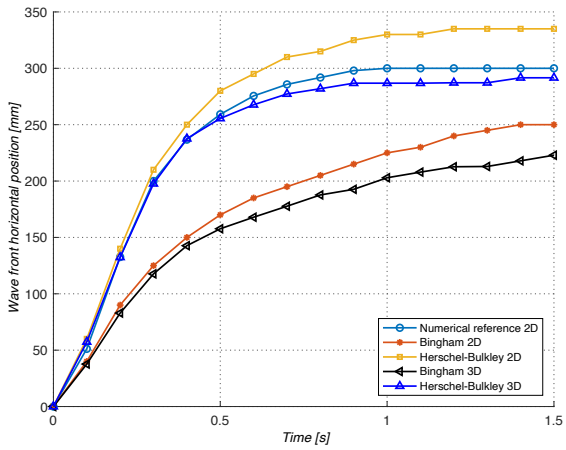


Figure 3.81: Test 12 results in 3D

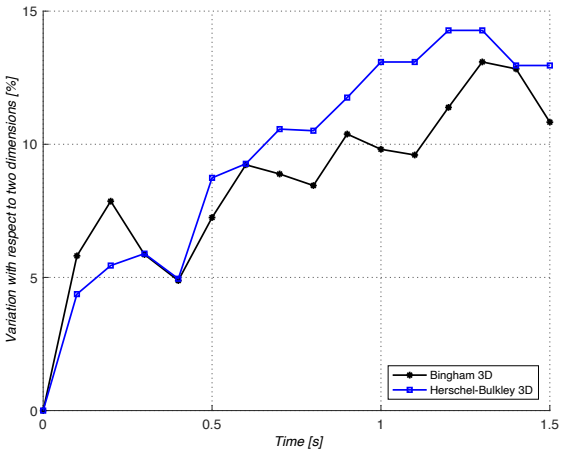
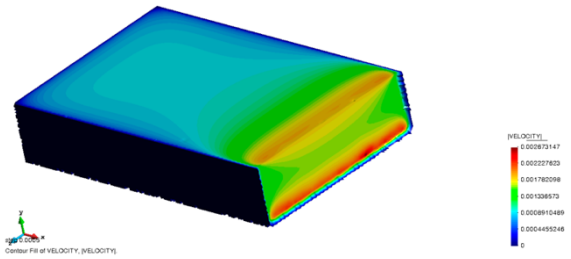


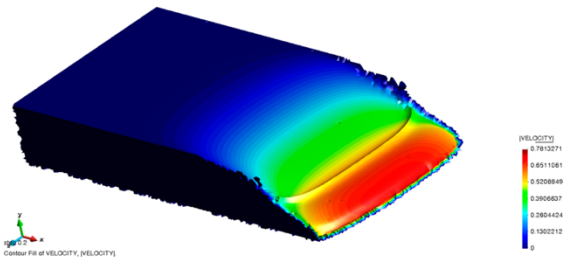
Figure 3.82: Variation of results between Test 12 in 3D and experimental reference

3.2.4. Results in 3D

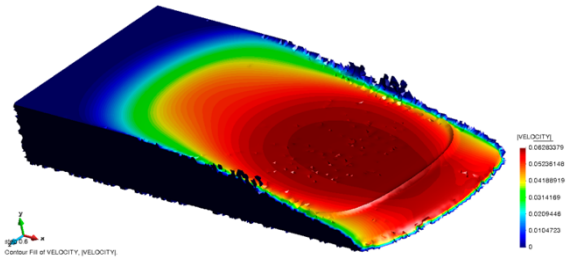
The evolution of the speed of Test 7 in three dimensions is attached. In the Figure 3.83, it can be observed that the last 5 seconds of simulation the velocity data hardly change, which also implies a null displacement, already visualized previously in the Figure 3.77.



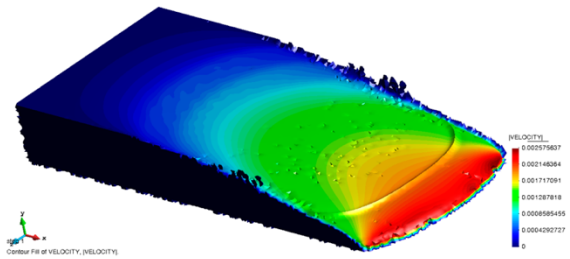
$t = 0.0\ s$



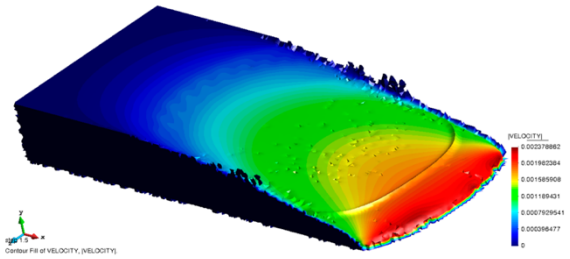
$t = 0.2\ s$



$t = 0.6\ s$



$t = 1.0\ s$



$t = 1.5\ s$

Figure 3.83: Test 7 velocity field in three dimensions at different time steps.

It is clearly observed that the friction of the side walls causes friction between the particles of the system thus slowing down the whole movement.

In the Figure 3.84 and Figure 3.85 it can see the effects produced by the three-dimensional simulation of the same case. The motion stops earlier and the velocity field is slightly different because of this. The results are even lower than the two-dimensional numerical reference values.

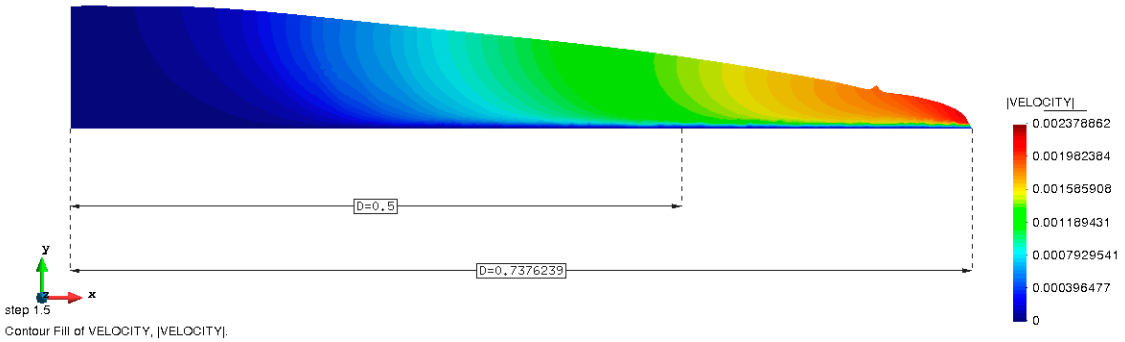


Figure 3.84: Test 7 velocity field in three dimensions at the end of the simulation (split in half)

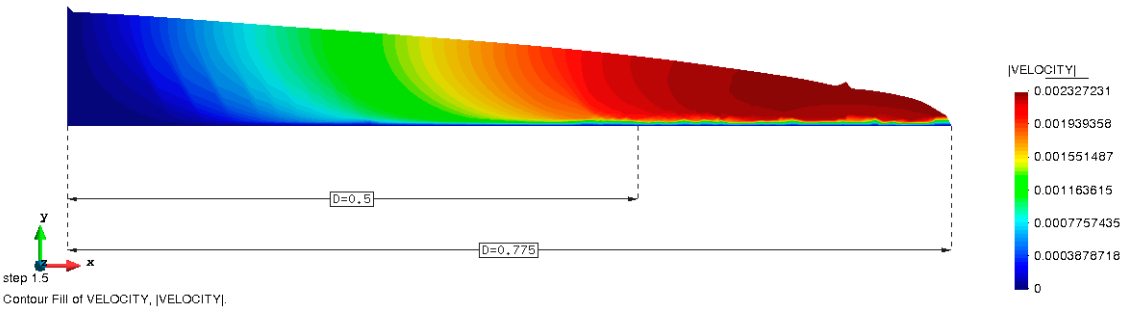


Figure 3.85: Test 7 velocity field in two dimensions at the end of the simulation

Even if the same upper limitation of the velocity of the fluid motion front exists, it is not distributed in the same way.

In the three-dimensional case, velocities along the expansion wave are lower than in the two-dimensional case. This is, as previously commented, due to the friction of the side walls, which exert forces in the opposite direction to the movement, thus slowing down the movement of the fluid in general. This also translates to the fact that progress over a given period of time is lower in 3D.

4. Conclusion and future lines of research

4.1. Conclusions

The objective of the present work was the implementation with subsequent validation by means of contrasting examples found in the literature of the Herschel-Bulkley constitutive law model of fluid mechanics in the Kratos Multiphysics program solved by means of the PFEM formulation. In order to avoid numerical calculation problems, this constitutive law has been regularized by the Papanastasiou parameter, thus facilitating the implementation of the law in computation.

Not only the Herschel-Bulkley model but also the Bingham model has been used, both in two and three dimensions. This has been done basing on the previous implementation of the Bingham model in the Kratos Multiphysics software in order to make the pertinent modifications to the programming code and transform it into a more generalized law.

Before performing the Herschel-Bulkley validations, convergence analyses of the mesh size, the time step, the final computation time and the Papanastasiou regulator parameter have been carried out in order to understand and visualize to what extent their values influence the numerical results.

It is concluded that both the mesh size and the step time variation, the smaller their values, the closer they are to the contrast examples. However, their price to pay is the computational cost associated with the decrease of their values. In the numerical analyses, it has been shown how the cost increases as the step time increases. The opposite happens when the Papanastasiou parameter is varied. There comes a point, a value, that from then on, referring to increasing its value, produces hardly any substantial variations to the final results except, of course, to the computational time used, thus increasing the computational cost to values that are not practical for simulations.

It has been observed that, at relatively small motion inertias, the value of the yield stress dictates the stopping of the fluid motion in parameters of position variation either by applying one law or the other, differing only in the time required for this.

The validations present slight variations with the numerical reference values. This may be due to the mass conservation law used when simulating, since the references use the incompressibility of the fluid and here, we start from the quasi-incompressibility of the medium. But even so, it can be seen that it depends on the case treated, because although in lower fluid heights when simulating the breakage of the dam, the results present a greater variation, when it comes to higher heights the differences are negligible.

In the formulation proposed by PFEM, use is made of the bulk modulus that defines the compressibility of the medium. This parameter is related to the propagation velocity of a wave in the fluid in question. It is used because it is better adjusted to the reality of fluids when it comes to their dynamics.

When comparing the two-dimensional simulations with the three-dimensional simulations, the effect of the walls and their slowing down on the movement and therefore the stopping of the movement can be seen, significantly reducing the variations of the results with the numerical examples extracted from the literature.

All the validation plots have been executed using MatLab software having collected the GiD post-processing data in Excel sheets for a sorting of the data. Qualitative plots such as Figure 2.3 and Figure 2.6 have also been plotted using MatLab to better visualize the theoretical explanation of the model discussed.

It can be assured that the Herschel-Bulkley model still has a long way to go when applied by PFEM, testing different regularization parameters that can compensate for the difference in initial mechanical formulations.

The main contributions of this work are:

- Implementation of the Herschel-Bulkley constitutive law in the PFEM code of Kratos Multiphysics.
- Calibration of simulation parameters by convergence analysis.
- Validation of the Herschel-Bulkley against Bingham, numerical and experimental data.
- Matching of both models with two-dimensional and three-dimensional cases.

4.2. Future lines of research

As future lines of research it is proposed to continue contrasting the model by means of more literature of the same.

In order to make a closer approximation to the reality of the model, it is proposed to couple thermodynamics to the mechanics of motion, since articles such as [23] is state the need to detail variables of the Herschel-Bulkley model by means of temperature-dependent functions, such as the consistency index (5.1) (or dynamic viscosity in the particular case of Bingham) or the yield stress itself. In [23] is proposed that the variation with respect to temperature of the parameter is directly proportional to itself.

$$\frac{dk}{d\theta} = -\alpha k \quad \rightarrow \quad k(\theta) = k_o \, e^{\alpha(\theta_o - \theta)} \tag{5.1}$$

Where "k_o" is the viscosity at "θ_o" temperature.

The code has been left open for the implementation of the same model but temperature-dependent, making the implementation within the code of the heat equation in the practical case of null convective terms necessary due to the use of the Lagrangian description of the motion, thus reducing the computational cost of all the simulations and being able to better represent the deformations of the system.

5. References

[1] T. C. Papanastasiou, "Flows of Materials with Yield," The Society of Rheology, Michigan 48109, 1987.

[2] O. C.Zienkiewicz, P. C.Jain and E. Oñate, "Flow of solids during forming and extrusion: Some aspects of numerical solutions," International Journal of Solids and Structures, 1978.

[3] R. P. C. R. P. &. E. V. Bharti, "Steady Flow of Power Law Fluids across a Circular Cylinder," The Canadian Journal of Chemical Engineering, 2008.

[4] R. Tanner and J. Milthorpe, "Numerical simulation of the flow of fluids with yield stresses," Proceedings of the Third International Conference on Numerical Methods in Laminar and Turbulent Flow:, Swansea, 1983.

[5] A. R. D. a. K. W. M. J. Crochet, "Numerical Simulation of Non-Newtonian Flow," *Rheology Seires 1,* no. Physical Sciences, Fluid Mechanics, p. 352, 1984.

[6] X. Oliver Olivella and C. Agelet de Saracíbar Bosch, Mecánica de medios continuos para ingenieros, Barcelona: Edicios UPC, 2000.

[7] A. Franci and X. Zhang, "3D numerical simulation of free-surface Bingham fluids interacting with structures using the PFEM," *Journal of Non-Newtonian Fluid Mechanics,* p. 42, 2018.

[8] M. C. A. Franci, "3D regularized μ(I)-rheology for granular flows simulation," *J. Comput. Phys. 378,* p. 257–277, 2019.

[9] E. O. J. C. A. Franci, "Unified Lagrangian formulation for solid and fluid mechanics and FSI problems," *Comput. Methods Appl. Mech. Engrg. 298,* p. 520–547, 2016.

[10] R. R. E. O. S. I. A. Larese, "Validation of the particle finite element method (PFEM) for simulation of free surface flows," *Int. J. Comput.-Aided Engineering Software,* vol. 25, p. 385–425, 2008.

[11] Z. Han, B. Su, Y. Li, W. Wang, W. Wang, J. Huang and G. Chen, "Numerical simulation of debris-flow behavior based on the SPH method incorporating the Herschel-Bulkley-Papanastasiou rheology model," *Elsevier,* no. Engineering Geology 255, pp. 26-36, 2019.

[12] E. Magnon and E. Cayeux, "Precise Method to Estimate the Herschel-Bulkley Parameters from Pipe Rheometer Measurements," MDPI, Stavanger (Norway), 2021.

[13] E. Bovet, B. Chiaia and L. Preziosi , "A new model for snow avalanche dynamics based on non-Newtonian fluids," *Meccanica,* vol. 45, p. 753–765, 2010.

[14] Y.Liu, N.J.Balmforth, S. Hormozi and D.R.Hewitt, "Two–dimensional viscoplastic dambreaks," *Journal of Non-Newtonian Fluid Mechanics,* vol. 238, pp. 65-79, December 2016.

[15] CIMNE, "GiD. The personal pre and post processor," CIMNE, [Online]. Available: https://www.gidhome.com/. [Accessed January 2022].

[16] E. Moreno and M. Cervera, lementos Finitos Mixtos Estabilizados para Flujos Viscoplásticos. Formulación y Aplicaciones, Barcelona: Monografía CIMNE No-142, 2014, p. 346.

[17] M. Cremonesi, A. Franci, S. Idelsohn and E. Oñate, "A state of the art review of the Particle Finite Element Method (PFEM)," no. Computational Methods in Engineering, p. 27, 2020.

[18] F. Robusté, "Planning and management of transport in the territory (Unit 2 - Location)," Barcelona, 2019.

[19] A. Franci and M. Cremonesi, "On the effect of standard PFEM remeshing on volume conservation in free-surface fluid flow problems," *Comp. Part. Mech,* 2016.

[20] G. Della Vecchia, M. Cremonesi and F. Pisanò, "On the rheological characterisation of liquefied sands through the dam-breaking test," *Int J Numer Anal Methods Geomech,* 2019.

[21] R. Brondani Minussi and G. de Freitas Maciel , "Numerical Experimental Comparison of Dam Break Flows with nonNewtonian Fluids," *J. of the Braz. Soc. of Mech. Sci. & Eng.,* vol. XXXIV, no. 2, pp. 167-178, Abril-June 2021.

[22] E. Lakzian, A. Estiri and M. Niazi, "Investigation of numerical simulation of non-Newtonian flows dam break and dam breach in open channels using modified VOF method," *Progress in Computational Fluid Dynamics,* vol. 19, no. 4, pp. 220-234, 2019.

[23] N. Bernabeu, P. Saramito and C. Smutek, "Modelling lava flow advance using a shallow-depth approximation for three-dimensional cooling of viscoplastic flows," France, 2016.

[24] E. W.V. Chaves, Notes on Continuum Mechanics, Ciudad Real: Springe, 2013.

[25] E. Bendito Perez, Geometría Diferencial y Ecuaciones Diferenciales, Barcelona: UPC Editions, 2020.

[26] M. Lawrence E., Introduction to the Mechanics of a Continuous Medium, Michigan, New Jersey: Prentice-Hall, Inc., 1969.

[27] CIMNE, "GiD. The personal pre and post processor," CIMNE, [Online]. Available: https://www.gidhome.com/. [Accessed 2022].

[28] CIMNE, "Kratos Multiphysics," CIMNE, [Online]. Available: https://www.cimne.com/kratos/. [Accessed 2022].

[29] Wikipedia, "Augmented Lagrangian method," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Augmented_Lagrangian_method. [Accessed January 2022].

[30] E. Moreno, A. Larese and M. Cervera, "Modelling of Bingham and Herschel–Bulkley flows with mixed P1/P1 finite elements stabilized with orthogonal subgrid scale," Journal of Non-Newtonian Fluid Mechanics 228, Barcelona, 2016.

[31] R. Huilgol and Z. Z. You, "Application of the augmented Lagrangian method to steady pipe flows of Bingham, Casson and Herschel–Bulkley fluids," Journal of Non-Newtonian Fluid Mechanics, Adelaide, 2005.

[32] W. Wang, G. Chen, Z. Han, S. Zhou, H. Zhang and P. Jing, "3D numerical simulation of debris-flow motion using SPH method incorporating non-Newtonian fluid behavior," *Nat Hazards,* vol. 81, p. 1981–1998 (18), 2016.

[33] C.-H. Jeon and B. R. Hodges, "Comparing thixotropic and Herschel–Bulkley parameterizations for continuum models of avalanches and subaqueous debris flows," *Nat. Hazards Earth Syst. Sci.,* vol. 18, p. 303–319, 2018.

Annex A. Tensor Calculus

A-A.1. Kronecker delta

$$\underline{\underline{\boldsymbol{\delta}}} = \boldsymbol{\delta_{ij}} = \frac{\boldsymbol{\partial x^i}}{\boldsymbol{\partial x^j}} = \frac{\boldsymbol{\partial x^j}}{\boldsymbol{\partial x^i}} = \begin{cases} \mathbf{0} & \boldsymbol{if\ i \neq j} \\ \mathbf{1} & \boldsymbol{if\ i = j} \end{cases}$$

It is a symmetric matrix:

$$\delta_{ij} = \delta_{ji}$$

Its inverse is itself:

$$[\delta^{-1}]_{ij} = \delta_{ij}$$

By the above property, it is also an orthogonal matrix:

$$[\delta^{-1}]_{ij} = \delta_{ji}$$

For completeness, the definition of the Kronecker delta in covariant, contravariant and mixed way are the same:

$$\delta_{ij} = \delta^{ij} = \delta_i{}^j = \delta^i{}_j$$

Example in three dimensions:

$$\underline{\underline{\delta}} = \delta_{ij} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A-A.2. Levi-Civita Tensor. Permutation symbol

$$\epsilon_{i_1 \dots i_n} = \begin{cases} +1 & if\ (a_1, \dots, a_n)\ is\ an\ even\ permutation\ of\ (1, \dots, n) \\ -1 & if\ (a_1, \dots, a_n)\ is\ an\ odd\ permutation\ of\ (1, \dots, n) \\ 0 & otherwise \end{cases}$$

The definition of the Levi-Civita symbol in covariant and contravariant way are the same:

$$\epsilon_{i_1 \dots i_n} = \epsilon^{i_1 \dots i_n}$$

Differential operators of the Permutation Symbol:

$$\boldsymbol{Gradient} \quad \rightarrow \quad \underline{\boldsymbol{\nabla}}\, \underline{\underline{\underline{\boldsymbol{\epsilon}}}} = \epsilon_{jkl,i} = \frac{\partial \epsilon_{jkl}}{\partial x_i} = \mathbb{O}_{ijkl} = \frac{\partial \epsilon_{ijk}}{\partial x_l} = \epsilon_{ijk,l} = \underline{\underline{\underline{\epsilon}}}\, \underline{\nabla}$$

$$\boldsymbol{Divergence} \quad \rightarrow \quad \underline{\boldsymbol{\nabla}} \cdot \underline{\underline{\underline{\boldsymbol{\epsilon}}}} = \epsilon_{ijk,i} = \frac{\partial \epsilon_{ijk}}{\partial x_i} = \mathbb{O}_{jk} = \mathbb{O}_{ij} = \frac{\partial \epsilon_{ijk}}{\partial x_k} = \epsilon_{ijk,k} = \underline{\underline{\underline{\epsilon}}} \cdot \underline{\nabla}$$

$$\boldsymbol{Curl} \quad \rightarrow \quad \underline{\nabla} \times \underline{\underline{\epsilon}} = \epsilon_{ijk}\epsilon_{klm,j} = \epsilon_{ijk}\frac{\partial \epsilon_{klm}}{\partial x_j} = \mathbb{O}_{ilm}$$

Where "$\mathbb{O}_{ij}$", "$\mathbb{O}_{ijk}$" and "$\mathbb{O}_{ijkl}$" are the second, third and fourth order tensor respectively with all its components equal to zero.

Example in three dimensions:

$\underline{\underline{\epsilon}} = \epsilon_{ijk} =$

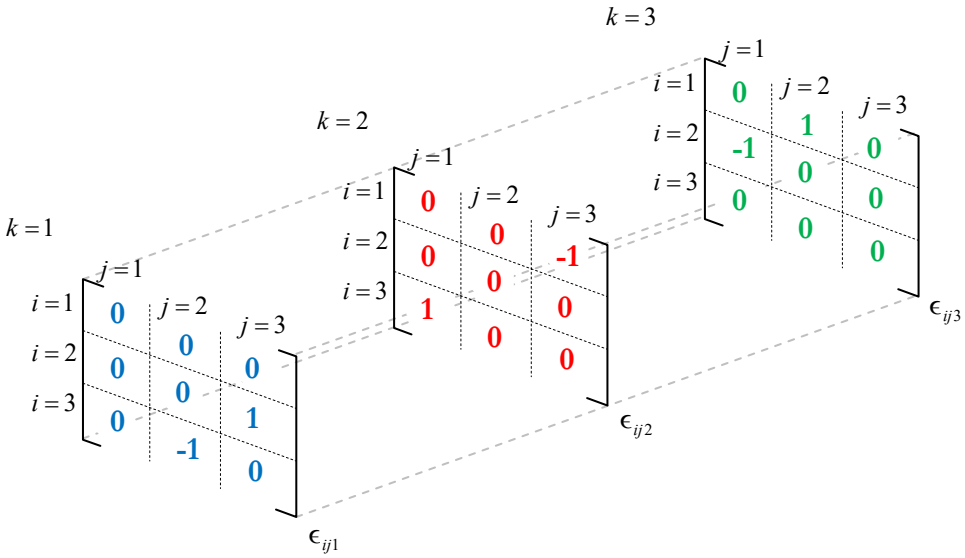


Figure A-A.0.1:Levi-Cita Tensor. Source: [24]

A-A.3. Scalar/Dot product

Scalar/Dot Product between two vectors:

$$\underline{a} \cdot \underline{b} = \underline{b} \cdot \underline{a} = \delta_{ij}\, a_i\, b_j$$

An interesting property is:

$$\underline{a} \cdot \underline{b} = tr(\underline{a} \otimes \underline{b})$$

A-A.4. Double index contraction

A-A.4.1. Double (vertical) dot product

$$\underline{\underline{\boldsymbol{A}}} : \underline{\underline{\boldsymbol{B}}} = \boldsymbol{tr}\left(\underline{\underline{\boldsymbol{A}}} \cdot \underline{\underline{\boldsymbol{B}}}^{\boldsymbol{T}}\right) = tr\left(\underline{\underline{A}}^T \cdot \underline{\underline{B}}\right) = tr\left(\underline{\underline{B}}^T \cdot \underline{\underline{A}}\right) = tr\left(\underline{\underline{B}} \cdot \underline{\underline{A}}^T\right) = \underline{\underline{B}} : \underline{\underline{A}}$$

$$\underline{\underline{\boldsymbol{A}}} : \underline{\underline{\boldsymbol{B}}} = A_{ij}\, B_{ij} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} : \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix}$$

$$\underline{\underline{\boldsymbol{A}}} : \underline{\underline{\boldsymbol{B}}} = A_{ij}\, B_{ij} = A_{11}B_{11} + A_{12}B_{12} + A_{13}B_{13} + A_{21}B_{21} + A_{22}B_{22} + A_{23}B_{23} + A_{31}B_{31} + A_{32}B_{32} + A_{33}B_{33}$$

A-A.4.2. Double (horizontal) dot product

$$\underline{\underline{\boldsymbol{A}}} \cdot\cdot \underline{\underline{\boldsymbol{B}}} = \boldsymbol{tr}\left(\underline{\underline{\boldsymbol{A}}} \cdot \underline{\underline{\boldsymbol{B}}}\right) = \boldsymbol{tr}\left(\underline{\underline{\boldsymbol{B}}} \cdot \underline{\underline{\boldsymbol{A}}}\right) = \underline{\underline{\boldsymbol{B}}} \cdot\cdot \underline{\underline{\boldsymbol{A}}}$$

$$\underline{\underline{A}} \cdot\cdot \underline{\underline{B}} = A_{ij}\, B_{ji} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} : \begin{pmatrix} B_{11} & B_{21} & B_{31} \\ B_{12} & B_{22} & B_{32} \\ B_{13} & B_{23} & B_{33} \end{pmatrix}$$

$$\underline{\underline{\boldsymbol{A}}} \cdot\cdot \underline{\underline{\boldsymbol{B}}} = A_{ij}\, B_{ji} = A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31} + A_{21}B_{12} + A_{22}B_{22} + A_{23}B_{32} + A_{31}B_{13} + A_{32}B_{23} + A_{33}B_{33}$$

Relation between the both types of double index contraction:

$$\underline{\underline{A}} : \underline{\underline{B}} = \underline{\underline{A}} \cdot\cdot \underline{\underline{B}} \quad \leftrightarrow \quad \underline{\underline{A}} = \underline{\underline{A}}^T \;\; or \;\; \underline{\underline{B}} = \underline{\underline{B}}^T$$

$$If \;\; A_{ij} = A_{ji} \;\; and \;\; B_{ij} = -B_{ji} \;\; \rightarrow \;\; A_{ij}\, B_{ij} = \underline{\underline{A}} : \underline{\underline{B}} = 0$$

A-A.5. Norm

The "*norm*" of a second order tensor (*matrix*) is defined [6]:

$$\boldsymbol{A} = \left\|\underline{\underline{\boldsymbol{A}}}\right\| = \left(\underline{\underline{\boldsymbol{A}}} : \underline{\underline{\boldsymbol{A}}}\right)^{\frac{1}{2}} \qquad A = \left\|A_{ij}\right\| = \left(A_{ij}\, A_{ij}\right)^{\frac{1}{2}}$$

Example in three dimensions:

$$A = \left\|\underline{\underline{A}}\right\| = \sqrt{\underline{\underline{A}} : \underline{\underline{A}}} = \sqrt{(A_{11})^2 + (A_{12})^2 + (A_{13})^2 + (A_{21})^2 + (A_{22})^2 + (A_{23})^2 + (A_{31})^2 + (A_{32})^2 + (A_{33})^2}$$

Symmetric second order tensor norm:

$$If \;\; A_{ij} = A_{ji} \;\; \rightarrow \;\; \left\|A_{ij}\right\| = \sqrt{(A_{11})^2 + (A_{22})^2 + (A_{33})^2 + 2(A_{12})^2 + 2(A_{13})^2 + 2(A_{23})^2}$$

A-A.6. Differential Operators

A-A.6.1. Scalars

A-A.6.1.1. Gradient

$$\boldsymbol{a}\, \underline{\boldsymbol{\nabla}} = a \otimes \underline{\nabla} = a \otimes \nabla_i = [a] \otimes [\nabla]_i = [a\, \nabla]_i = [a \otimes \nabla]_i = \frac{\boldsymbol{\partial a}}{\boldsymbol{\partial x_i}} = a_{,i}$$

Example in three dimensions in cartesian coordinates:

$$a\ \underline{\nabla} = \begin{pmatrix} \frac{\partial a}{\partial x_1} \\ \frac{\partial a}{\partial x_2} \\ \frac{\partial a}{\partial x_3} \end{pmatrix}$$

The commutative property is satisfied:

$$\underline{\boldsymbol{\nabla}}\ \boldsymbol{a} = \underline{\nabla}\otimes a = \nabla_i \otimes a = [\nabla]_i \otimes [a] = [\nabla\ a]_i = [\nabla \otimes a]_i = \boldsymbol{\frac{\partial a}{\partial x_i}} = a_{,i} = a\ \underline{\nabla}$$

$$\underline{\nabla}\ a = a\ \underline{\nabla}$$

A-A.6.1.2. Divergence

It does not exist.

A-A.6.1.3. Rotational

It does not exist.

A-A.6.2. Vectors

A-A.6.2.1. Gradient

$$\underline{\boldsymbol{a}}\ \underline{\boldsymbol{\nabla}} = \underline{a}\otimes\underline{\nabla} = a_i \otimes \nabla_j = [a]_i \otimes [\nabla]_j = [a\ \nabla]_{ij} = [a \otimes \nabla]_{ij} = \boldsymbol{\frac{\partial a_i}{\partial x_j}} = a_{i,j}$$

Example in three dimensions in cartesian coordinates:

$$\underline{a}\ \underline{\nabla} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_3} \end{pmatrix} = \begin{pmatrix} \frac{\partial a_1}{\partial x_1} & \frac{\partial a_1}{\partial x_2} & \frac{\partial a_1}{\partial x_3} \\ \frac{\partial a_2}{\partial x_1} & \frac{\partial a_2}{\partial x_2} & \frac{\partial a_2}{\partial x_3} \\ \frac{\partial a_3}{\partial x_1} & \frac{\partial a_3}{\partial x_2} & \frac{\partial a_3}{\partial x_3} \end{pmatrix}$$

Jacobian matrix:

$$\begin{cases} \underline{\underline{J}} = \underline{a}\ \underline{\nabla} \\ J_{ij} = a_{i,j} \end{cases}$$

A-A.6.2.2. Transpose vector gradient

$$\underline{\boldsymbol{\nabla}}\ \underline{\boldsymbol{a}} = \left(\underline{a}\ \underline{\nabla}\right)^T = \underline{\nabla}\otimes\underline{a} = \nabla_i \otimes a_j = [\nabla]_i \otimes [a]_j = [\nabla\ a]_{ij} = [\nabla \otimes a]_{ij} = \boldsymbol{\frac{\partial a_j}{\partial x_i}} = a_{j,i}$$

Example in three dimensions in cartesian coordinates:

$$\underline{\nabla}\,\underline{a} = \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} \end{pmatrix} (a_1 \quad a_2 \quad a_3) = \begin{pmatrix} \frac{\partial a_1}{\partial x_1} & \frac{\partial a_2}{\partial x_1} & \frac{\partial a_3}{\partial x_1} \\ \frac{\partial a_1}{\partial x_2} & \frac{\partial a_2}{\partial x_2} & \frac{\partial a_3}{\partial x_2} \\ \frac{\partial a_1}{\partial x_3} & \frac{\partial a_2}{\partial x_3} & \frac{\partial a_3}{\partial x_3} \end{pmatrix}$$

A-A.6.2.3. Laplacian

$$\begin{cases} \underline{\nabla} \cdot \left(\underline{\nabla}\,\underline{a}\right) = \underline{\nabla}^2\,\underline{a} = \Delta\,\underline{a} \\ \left(a_{j,i}\right)_{,i} = a_{j,ii} \end{cases}$$

Example in three dimensions in cartesian coordinates:

$$\Delta\,\underline{a} = \left(\frac{\partial^2}{\partial {x_1}^2} \quad \frac{\partial^2}{\partial {x_2}^2} \quad \frac{\partial^2}{\partial {x_3}^2}\right) \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \frac{\partial^2 a_1}{\partial {x_1}^2} + \frac{\partial^2 a_2}{\partial {x_2}^2} + \frac{\partial^2 a_3}{\partial {x_3}^2}$$

A-A.6.2.4. Divergence

$$\underline{\boldsymbol{\nabla}}\,\underline{\boldsymbol{a}} : \underline{\underline{\boldsymbol{\delta}}} = \underline{\boldsymbol{\nabla}} \cdot \underline{\boldsymbol{a}} = \nabla_i \cdot a_i = [\nabla]_i \cdot [a]_i = [\nabla \cdot a] = \boldsymbol{\frac{\partial a_i}{\partial x_i}} = a_{i,i}$$

$$\underline{\boldsymbol{a}}\,\underline{\boldsymbol{\nabla}} : \underline{\underline{\boldsymbol{\delta}}} = \underline{\boldsymbol{a}} \cdot \underline{\boldsymbol{\nabla}} = a_i \cdot \nabla_i = [a]_i \cdot [\nabla]_i = [a \cdot \nabla] = \boldsymbol{\frac{\partial a_i}{\partial x_i}} = a_{i,i}$$

Example in three dimensions in cartesian coordinates:

$$\underline{\boldsymbol{\nabla}} \cdot \underline{\boldsymbol{a}} = \left(\frac{\partial}{\partial x_1} \quad \frac{\partial}{\partial x_2} \quad \frac{\partial}{\partial x_3}\right) \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \frac{\partial a_1}{\partial x_1} + \frac{\partial a_2}{\partial x_2} + \frac{\partial a_3}{\partial x_2}$$

There are texts (clear examples are books on Differential Geometry [24]) where it is defined:

$$\underline{a} \cdot \underline{\nabla} = a_i \frac{\partial}{\partial x_i} = a_1 \frac{\partial}{\partial x_1} + a_2 \frac{\partial}{\partial x_2} + a_3 \frac{\partial}{\partial x_3}$$

However, in this wording the order of the divergence of a vector is used interchangeably.

A-A.6.2.5. Curl

Also known as Rotational [6]**:**

$$\underline{\boldsymbol{\nabla}} \times \underline{\boldsymbol{a}} = \underline{\boldsymbol{\nabla}} \wedge \underline{\boldsymbol{a}} = \boldsymbol{\epsilon_{ijk}} \boldsymbol{\frac{\partial a_k}{\partial x_j}} = \epsilon_{ijk}\, a_{k,j}$$

Example in three dimensions in cartesian coordinates:

$$\underline{\nabla} \times \underline{a} = \begin{pmatrix} \frac{\partial a_3}{\partial x_2} - \frac{\partial a_2}{\partial x_3} \\ \frac{\partial a_1}{\partial x_3} - \frac{\partial a_3}{\partial x_1} \\ \frac{\partial a_2}{\partial x_1} - \frac{\partial a_1}{\partial x_2} \end{pmatrix}$$

The anticommutative property is satisfied:

$$\underline{\boldsymbol{a}} \times \underline{\boldsymbol{\nabla}} = \frac{\boldsymbol{\partial a_i}}{\boldsymbol{\partial x_j}} \boldsymbol{\epsilon_{ijk}} = -\underline{\boldsymbol{\nabla}} \times \underline{\boldsymbol{a}}$$

A-A.6.3. Matrixes

A-A.6.3.1. Gradient

$$\underline{\boldsymbol{\nabla}}\,\underline{\underline{\boldsymbol{A}}} = \underline{\nabla} \otimes \underline{\underline{A}} = \nabla_i \otimes A_{jk} = [\nabla]_i \otimes [A]_{jk} = [\nabla\, A]_{ijk} = [\nabla \otimes A]_{ijk} = A_{jk,i} = \frac{\boldsymbol{\partial A_{jk}}}{\boldsymbol{\partial x_i}}$$

$$\underline{\underline{\boldsymbol{A}}}\,\underline{\boldsymbol{\nabla}} = \underline{\underline{A}} \otimes \underline{\nabla} = A_{ij} \otimes \nabla_k = [A]_{ij} \otimes [\nabla]_k = [A\, \nabla]_{ijk} = [A \otimes \nabla]_{ijk} = A_{ij,k} = \frac{\boldsymbol{\partial A_{ij}}}{\boldsymbol{\partial x_k}}$$

Example in three dimensions in cartesian coordinates:

$$\underline{\nabla}\,\underline{\underline{A}} = \left[\begin{pmatrix} \frac{\partial A_{11}}{\partial x_1} & \frac{\partial A_{21}}{\partial x_1} & \frac{\partial A_{31}}{\partial x_1} \\ \frac{\partial A_{11}}{\partial x_2} & \frac{\partial A_{21}}{\partial x_2} & \frac{\partial A_{31}}{\partial x_2} \\ \frac{\partial A_{11}}{\partial x_3} & \frac{\partial A_{21}}{\partial x_3} & \frac{\partial A_{31}}{\partial x_3} \end{pmatrix} \begin{pmatrix} \frac{\partial A_{12}}{\partial x_1} & \frac{\partial A_{22}}{\partial x_1} & \frac{\partial A_{32}}{\partial x_1} \\ \frac{\partial A_{12}}{\partial x_2} & \frac{\partial A_{22}}{\partial x_2} & \frac{\partial A_{32}}{\partial x_2} \\ \frac{\partial A_{12}}{\partial x_3} & \frac{\partial A_{22}}{\partial x_3} & \frac{\partial A_{32}}{\partial x_3} \end{pmatrix} \begin{pmatrix} \frac{\partial A_{13}}{\partial x_1} & \frac{\partial A_{23}}{\partial x_1} & \frac{\partial A_{33}}{\partial x_1} \\ \frac{\partial A_{13}}{\partial x_2} & \frac{\partial A_{23}}{\partial x_2} & \frac{\partial A_{33}}{\partial x_2} \\ \frac{\partial A_{13}}{\partial x_3} & \frac{\partial A_{23}}{\partial x_3} & \frac{\partial A_{33}}{\partial x_3} \end{pmatrix} \right]$$

$$\underline{\underline{A}}\,\underline{\nabla} = \left[\begin{pmatrix} \frac{\partial A_{11}}{\partial x_1} & \frac{\partial A_{12}}{\partial x_1} & \frac{\partial A_{13}}{\partial x_1} \\ \frac{\partial A_{21}}{\partial x_1} & \frac{\partial A_{22}}{\partial x_1} & \frac{\partial A_{23}}{\partial x_1} \\ \frac{\partial A_{31}}{\partial x_1} & \frac{\partial A_{32}}{\partial x_1} & \frac{\partial A_{33}}{\partial x_1} \end{pmatrix} \begin{pmatrix} \frac{\partial A_{11}}{\partial x_2} & \frac{\partial A_{12}}{\partial x_2} & \frac{\partial A_{13}}{\partial x_2} \\ \frac{\partial A_{21}}{\partial x_2} & \frac{\partial A_{22}}{\partial x_2} & \frac{\partial A_{23}}{\partial x_2} \\ \frac{\partial A_{31}}{\partial x_2} & \frac{\partial A_{32}}{\partial x_2} & \frac{\partial A_{33}}{\partial x_2} \end{pmatrix} \begin{pmatrix} \frac{\partial A_{11}}{\partial x_3} & \frac{\partial A_{12}}{\partial x_3} & \frac{\partial A_{13}}{\partial x_3} \\ \frac{\partial A_{21}}{\partial x_3} & \frac{\partial A_{22}}{\partial x_3} & \frac{\partial A_{23}}{\partial x_3} \\ \frac{\partial A_{31}}{\partial x_3} & \frac{\partial A_{32}}{\partial x_3} & \frac{\partial A_{33}}{\partial x_3} \end{pmatrix} \right]$$

A-A.6.3.2. Divergence

$$\underline{\boldsymbol{\nabla}}\,\underline{\underline{\boldsymbol{A}}} : \underline{\underline{\boldsymbol{\delta}}} = \underline{\boldsymbol{\nabla}} \cdot \underline{\underline{\boldsymbol{A}}} = \nabla_i \cdot A_{ij} = [\nabla]_i \cdot [A]_{ij} = [\nabla \cdot A]_j = \frac{\boldsymbol{\partial A_{ij}}}{\boldsymbol{\partial x_i}} = A_{ij,i}$$

$$\underline{\underline{\boldsymbol{A}}}\,\underline{\boldsymbol{\nabla}} : \underline{\underline{\boldsymbol{\delta}}} = \boldsymbol{div}\left(\underline{\underline{\boldsymbol{A}}}\right) = \underline{\underline{\boldsymbol{A}}} \cdot \underline{\boldsymbol{\nabla}} = A_{ij} \cdot \nabla_j = [A]_{ij} \cdot [\nabla]_j = [A \cdot \nabla]_i = \frac{\boldsymbol{\partial A_{ij}}}{\boldsymbol{\partial x_j}} = A_{ij,j}$$

Example in three dimensions in cartesian coordinates:

$$\underline{\nabla}\cdot\underline{\underline{A}}=\sum_{i=1}^{3}\frac{\partial A_{ij}}{\partial x_i}=\frac{\partial A_{1j}}{\partial x_1}+\frac{\partial A_{2j}}{\partial x_2}+\frac{\partial A_{3j}}{\partial x_3}=\begin{pmatrix}\frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_3}\end{pmatrix}\cdot\begin{pmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{pmatrix}=\begin{pmatrix}\frac{\partial A_{11}}{\partial x_1}+\frac{\partial A_{21}}{\partial x_2}+\frac{\partial A_{31}}{\partial x_3}\\ \frac{\partial A_{12}}{\partial x_1}+\frac{\partial A_{22}}{\partial x_2}+\frac{\partial A_{32}}{\partial x_3}\\ \frac{\partial A_{13}}{\partial x_1}+\frac{\partial A_{23}}{\partial x_2}+\frac{\partial A_{33}}{\partial x_3}\end{pmatrix}$$

$$\underline{\underline{A}}\cdot\underline{\nabla}=\sum_{j=1}^{3}\frac{\partial A_{ij}}{\partial x_j}=\frac{\partial A_{i1}}{\partial x_1}+\frac{\partial A_{i2}}{\partial x_2}+\frac{\partial A_{i2}}{\partial x_2}=\begin{pmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{pmatrix}\cdot\begin{pmatrix}\frac{\partial}{\partial x_1}\\ \frac{\partial}{\partial x_2}\\ \frac{\partial}{\partial x_3}\end{pmatrix}=\begin{pmatrix}\frac{\partial A_{11}}{\partial x_1}+\frac{\partial A_{12}}{\partial x_2}+\frac{\partial A_{13}}{\partial x_3}\\ \frac{\partial A_{21}}{\partial x_1}+\frac{\partial A_{22}}{\partial x_2}+\frac{\partial A_{23}}{\partial x_3}\\ \frac{\partial A_{31}}{\partial x_1}+\frac{\partial A_{32}}{\partial x_2}+\frac{\partial A_{33}}{\partial x_3}\end{pmatrix}$$

Relationship between one operator order or another:

$$\begin{cases}\underline{\underline{A}}=\underline{\underline{A}}^{T} & \leftrightarrow \quad \underline{\nabla}\cdot\underline{\underline{A}}=\underline{\underline{A}}\cdot\underline{\nabla}\\ A_{ij}=A_{ji} & \leftrightarrow \quad A_{ij,i}=A_{ij,j}\end{cases}$$

Since the **Cauchy Stress Tensor** is a symmetric one, then it could be done:

$$\boldsymbol{div\left(\underline{\underline{\sigma}}\right)=\underline{\underline{\sigma}}\cdot\underline{\nabla}=\underline{\nabla}\cdot\underline{\underline{\sigma}}=\ \sigma_{ij,i}=\sigma_{ij,j}}$$

A-A.6.3.3. Curl

$$\underline{\nabla}\times\underline{\underline{A}}=\boldsymbol{\epsilon_{ijk}\frac{\partial A_{kl}}{\partial x_j}}=\epsilon_{ijk}\,A_{kl,j}$$

Example in three dimensions in cartesian coordinates:

$$\underline{\nabla}\times\underline{\underline{A}}=\begin{pmatrix}\left(\frac{\partial A_{31}}{\partial x_2}-\frac{\partial A_{21}}{\partial x_3}\right) & \left(\frac{\partial A_{32}}{\partial x_2}-\frac{\partial A_{22}}{\partial x_3}\right) & \left(\frac{\partial A_{33}}{\partial x_2}-\frac{\partial A_{23}}{\partial x_3}\right)\\ \left(\frac{\partial A_{11}}{\partial x_3}-\frac{\partial A_{31}}{\partial x_1}\right) & \left(\frac{\partial A_{12}}{\partial x_3}-\frac{\partial A_{32}}{\partial x_1}\right) & \left(\frac{\partial A_{12}}{\partial x_3}-\frac{\partial A_{33}}{\partial x_1}\right)\\ \left(\frac{\partial A_{21}}{\partial x_1}-\frac{\partial A_{11}}{\partial x_2}\right) & \left(\frac{\partial A_{22}}{\partial x_1}-\frac{\partial A_{12}}{\partial x_2}\right) & \left(\frac{\partial A_{23}}{\partial x_1}-\frac{\partial A_{12}}{\partial x_2}\right)\end{pmatrix}$$

A relationship to take into account due to its use in the development of the compatibility equations in of continuous media mechanics (relationship between the medium motion equations and its undergoing strain tensor):

$$\underline{\underline{A}}\times\underline{\nabla}=\frac{\partial A_{jk}}{\partial x_i}\epsilon_{kil}=-\left(\underline{\nabla}\times\underline{\underline{A}}^{T}\right)^{T}$$

A-A.7. Integral theorems

A-A.7.1. Gauss Divergence Theorem

$$\int_{\Gamma \equiv \partial\Omega} \boldsymbol{\mathcal{X}_{ij\ldots klm}} \cdot \boldsymbol{d\Gamma_m} = \int_{\Gamma \equiv \partial\Omega} \mathcal{X}_{ij\ldots km} \cdot n_m \, d\Gamma = \int_{\Omega} \frac{\boldsymbol{\partial \mathcal{X}_{ij\ldots km}}}{\boldsymbol{\partial x_m}} \boldsymbol{d\Omega}$$

Example:

$$\int_{\partial\Omega} \underline{\underline{A}} \cdot d\underline{\Gamma} = \int_{\partial\Omega} \underline{\underline{A}} \cdot \underline{n} \, d\Gamma = \int_{\Omega} \underline{\underline{A}} \cdot \underline{\nabla} \, d\Omega$$

A-A.7.2. Generalized theorems

Generalizing the theorems of differential integration, it is obtained:

$$\int_{\boldsymbol{\partial\Omega}} \boldsymbol{\omega} = \int_{\boldsymbol{\Omega}} \boldsymbol{d\omega}$$

$$\int_{Boundary\ domain} function = \int_{Domain} function\ derivatives\ combination$$

A-A.8. Special tensors

$$Volumetric\ Tensor\ Operator \quad \rightarrow \quad \underline{\underline{\underline{\underline{\mathbb{V}}}}} = \mathbb{V}_{ijkl} = \frac{1}{3}\,\mathbb{I}_{ijkl} = \frac{1}{3}\left(\delta_{ij}\,\delta_{kl}\right)$$

$$Deviatoric\ Tensor\ Operator \quad \rightarrow \quad \underline{\underline{\underline{\underline{\mathbb{P}}}}} = \mathbb{P}_{ijkl} = \mathbb{I}_{ijkl} - \mathbb{V}_{ijkl}$$

The main property of these two new operators is the capacity of decomposition of a tensor:

$$\boldsymbol{Spherical\ part} \quad \rightarrow \quad \begin{cases} \underline{\underline{A}}^{sph} = \underline{\underline{\underline{\underline{\mathbb{V}}}}} : \underline{\underline{A}} \\ {A_{ij}}^{sph} = \mathbb{V}_{ijkl}\,A_{kl} \end{cases} \qquad \& \qquad \boldsymbol{Deviatoric\ part} \quad \rightarrow \quad \begin{cases} \underline{\underline{A}}' = \underline{\underline{\underline{\underline{\mathbb{P}}}}} : \underline{\underline{A}} \\ {A_{ij}}' = \mathbb{P}_{ijkl}\,A_{kl} \end{cases}$$

For any second-order tensor, can be split in 2 parts: the spherical part (or component) and the deviatoric part:

$$\begin{cases} \underline{\underline{A}} = \underline{\underline{A}}^{sph} + \underline{\underline{A}}' \\ A_{ij} = {A_{ij}}^{sph} + {A_{ij}}' \end{cases}$$

The spherical part is defined as:

$$\begin{cases} \underline{\underline{A}}^{sph} = \frac{1}{3}\, tr\left(\underline{\underline{A}}\right)\underline{\underline{1}} = \frac{1}{3}\left(\underline{\underline{A}} : \underline{\underline{1}}\right)\underline{\underline{1}} \\ A_{ij}{}^{sph} = \frac{1}{3}\, A_{kk}\, \delta_{ij} \end{cases} \quad \rightarrow \quad A^{sph} = \frac{(A_{11} + A_{22} + A_{33})}{3} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The deviatoric part can be defined as:

$$\begin{cases} \underline{\underline{A}}' = \underline{\underline{A}} - \underline{\underline{A}}^{sph} \\ A_{ij}{}' = A_{ij} - A_{ij}{}^{sph} \end{cases} \quad \rightarrow \quad A' = \begin{pmatrix} A_{11}{}' & A_{12}{}' & A_{13}{}' \\ A_{21}{}' & A_{22}{}' & A_{23}{}' \\ A_{31}{}' & A_{32}{}' & A_{33}{}' \end{pmatrix}$$

A-A.9. Invariants

A-A.9.1. Total 2nd order tensor invariants I

$$\boldsymbol{I_1} = tr\left(\underline{\underline{A}}\right) = \underline{\underline{A}} : \underline{\underline{1}} = A_{ij}\, \delta_{ij} = A_{ii} = A_{jj} = A_{11} + A_{22} + A_{33}$$

$$\boldsymbol{I_2} = \frac{1}{2}\left({I_1}^2 - \underline{\underline{A}} : \underline{\underline{A}}\right) = \frac{1}{2}[2\, A_{11}A_{22} + 2\, A_{11}A_{33} + 2\, A_{22}A_{33} - (A_{12})^2 - (A_{13})^2 - (A_{21})^2 - (A_{23})^2 - (A_{31})^2 - (A_{32})^2]$$

$$\boldsymbol{I_3} = det\left(\underline{\underline{A}}\right) = det(A_{ij}) = \frac{1}{6}\epsilon_{ijk}\epsilon_{pqr}A_{pi}A_{qj}A_{rk} = A_{11}A_{22}A_{33} + A_{21}A_{32}A_{13} + A_{31}A_{12}A_{23} - A_{13}A_{22}A_{31} - A_{23}A_{32}A_{11} - A_{33}A_{12}A_{21}$$

Symmetric total 2nd order tensor invariants I:

$$\boldsymbol{I_1} = A_{11} + A_{22} + A_{33}$$

$$\boldsymbol{I_2} = A_{11}A_{22} + A_{11}A_{33} + A_{22}A_{33} - (A_{12})^2 - (A_{13})^2 - (A_{23})^2$$

$$\boldsymbol{I_3} = A_{11}A_{22}A_{33} + 2\, A_{21}A_{32}A_{13} - A_{22}(A_{13})^2 - A_{11}(A_{23})^2 - A_{33}(A_{12})^2$$

A-A.9.2. Total 2nd order tensor invariants J

$$\boldsymbol{J_1} = I_1 = A_{ii} = A_{11} + A_{22} + A_{33}$$

$$\boldsymbol{J_2} = \frac{1}{2}\left({I_1}^2 + 2I_2\right) = \frac{1}{2}\left(\underline{\underline{A}} : \underline{\underline{A}}\right) = \frac{1}{2}\left(A_{ij}\, A_{ij}\right) = \frac{1}{2}\left({A_{11}}^2 + {A_{12}}^2 + {A_{13}}^2 + {A_{21}}^2 + {A_{22}}^2 + {A_{23}}^2 + {A_{31}}^2 + {A_{32}}^2 + {A_{33}}^2\right)$$

$$\boldsymbol{J_3} = \frac{1}{3}\, tr\left(\underline{\underline{A}} \cdot \underline{\underline{A}} \cdot \underline{\underline{A}}\right) = \frac{1}{3}\left(\underline{\underline{A}} \cdot \underline{\underline{A}} \cdot \underline{\underline{A}}\right) : \underline{\underline{1}} = \frac{1}{3}\, A_{ij}\, A_{jk}\, A_{kl}\, \delta_{il} = \frac{1}{3}\, A_{ij}A_{jk}A_{ki}$$

$$\boldsymbol{J_i = \frac{1}{i}\, tr\left(\underline{\underline{A}}^i\right)}$$

Symmetric total 2nd order tensor invariants J:

$$\boldsymbol{J_1} = A_{11} + A_{22} + A_{33}$$

$$\boldsymbol{J_2} = \frac{1}{2}\left({A_{11}}^2 + {A_{22}}^2 + {A_{33}}^2 + 2\,{A_{12}}^2 + 2\,{A_{13}}^2 + 2\,{A_{23}}^2\right)$$

A-A.9.3. Deviatoric 2nd order tensor invariants I

$$\boldsymbol{I'_1} = tr\left(\underline{\underline{A}}'\right) = {A_{ii}}' = 0$$

$$\boldsymbol{I'_2} = \frac{1}{2}\left(\underline{\underline{A}}' : \underline{\underline{A}}' - {I'_1}^2\right) = \frac{1}{2}\left(\underline{\underline{A}}' : \underline{\underline{A}}'\right) = \frac{1}{2}\left({A_{ij}}'{A_{ij}}'\right)$$

$$\boldsymbol{I'_3} = det\left(\underline{\underline{A}}'\right) = det\left({A_{ij}}'\right) = \frac{1}{3}{A_{ij}}'{A_{jk}}'{A_{ji}}'$$

A-A.9.4. Deviatoric 2nd order tensor invariants J

$$\boldsymbol{J'_1} = I'_1 = 0$$

$$\boldsymbol{J'_2} = I'_2 = \frac{1}{2}\left({A_{ij}}'{A_{ij}}'\right)$$

$$\boldsymbol{J'_3} = I'_3 = \frac{1}{3}{A_{ij}}'{A_{jk}}'{A_{ji}}'$$

Annex B. Review of the Continuum Mechanics

A-B.1. Description of motion

The mathematical description of the particle properties can be done in two ways [6]:

1) **Material (Lagrangian) description**, normally used in solid mechanics, described in terms of the material coordinates and time.
   Non-deformed or Reference configuration:

$$[\underline{X}] = [X_i] = \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \end{Bmatrix} = \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = Material\ coordiantes$$

2) **Spatial (Eulerian) description**, normally used in fluid mechanics, described in terms of the spatial coordinates and time.
   Deformed or Present configuration:

$$[\underline{x}] = [x_i] = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = Spatial\ coordiantes$$

A-B.1.1. Time derivatives

The time derivative of a given property can be defined based on the:

1) Material description $\Gamma(\underline{X}, t) \rightarrow$ **Total** or **Material derivative**: partial time derivative of the material description of the property.

$$Material\ derivative \equiv \frac{\partial \Gamma(\underline{X}, t)}{\partial t}$$

2) Spatial description $\gamma(\underline{x}, t) \rightarrow$ **Local** or **Spatial derivative**: partial time derivative of the spatial description of the property.

$$Spatial\ derivative \equiv \frac{\partial \gamma(\underline{x}, t)}{\partial t}$$

The material derivative of whatever function can be computed in terms of spatial descriptions:

TIME DERIVATIVE

$$\underbrace{\frac{d\underline{f}(\underline{x},t)}{dt}}_{\text{Material derivative}} = \underbrace{\frac{\partial \underline{f}(\underline{x},t)}{\partial t}}_{\text{Spatial derivative}} + \underbrace{\underline{v}(\underline{x},t) \cdot \underline{\nabla}\,\underline{f}(\underline{x},t)}_{\text{Convective derivative}}$$

Using the indices (*Einstein*) notation:

$$\dot{f}_i = f_{i,t} + v_j f_{i,j}$$

A-B.1.2. Velocity and acceleration

A-B.1.2.1. Velocity

It is the time derivative of the motion equations

$$Material\ description \quad \rightarrow \quad \underline{\boldsymbol{V}}(\underline{\boldsymbol{X}},\boldsymbol{t}) = \frac{\boldsymbol{\partial}\underline{\boldsymbol{x}}(\underline{\boldsymbol{X}},\boldsymbol{t})}{\boldsymbol{\partial t}}$$

$$Spatial\ description \quad \rightarrow \quad \underline{v}(\underline{x},t) = \underline{V}(\underline{X}(\underline{x},t),t)$$

A-B. 1.2.2. Acceleration

It is the time derivative of the velocity field

$$Material\ description \quad \rightarrow \quad \underline{A}(\underline{X},t) = \frac{\partial \underline{V}(\underline{X},t)}{\partial t}$$

$$Spatial\ description \quad \rightarrow \quad \underline{\boldsymbol{a}}(\underline{\boldsymbol{x}},\boldsymbol{t}) = \frac{\boldsymbol{d}\underline{\boldsymbol{v}}(\underline{\boldsymbol{x}},\boldsymbol{t})}{\boldsymbol{dt}} = \begin{cases} \underline{A}(\underline{X}(\underline{x},t),t) \\ \dfrac{\boldsymbol{\partial}\underline{\boldsymbol{v}}(\underline{\boldsymbol{x}},\boldsymbol{t})}{\boldsymbol{\partial t}} + \underline{\boldsymbol{v}}(\underline{\boldsymbol{x}},\boldsymbol{t}) \cdot \underline{\boldsymbol{\nabla}}\,\underline{\boldsymbol{v}}(\underline{\boldsymbol{x}},\boldsymbol{t}) = \dfrac{\partial \underline{v}}{\partial t} + \underline{v} \cdot \underline{\underline{l}} = \dfrac{\partial \underline{v}}{\partial t} + \dfrac{1}{2}\underline{\nabla}(v^2) \end{cases}$$

A-B.1.2.3. Stationarity

A property is stationarity (or steady-state) when its spatial description is not dependent on time:

$$Stationarity \quad \rightarrow \quad f(\underline{x},t) = f(\underline{x}) \quad \leftrightarrow \quad \frac{\partial f(\underline{x},t)}{\partial t} = 0$$

It cannot be assumed that the time independence in the spatial description implies time independence in the material description.

$$f(\underline{x},t) = f(\underline{x}) \quad \nleftrightarrow \quad F(\underline{X},t) = F(\underline{X})$$

A-B. 1.2.4. Uniformity

A property is uniform when its spatial description is not dependent on the spatial coordinates:

$$Uniformity \quad \rightarrow \quad f(\underline{x},t) = f(t) \quad \leftrightarrow \quad \frac{\partial f(\underline{x},t)}{\partial \underline{x}} = \underline{\nabla}\, f(\underline{x},t) = 0$$

On this occasion, it can be assumed that if its spatial description does not depend on the coordinates, neither does its material one:

$$f(\underline{x},t) = f(t) \quad \leftrightarrow \quad F(\underline{X},t) = F(t)$$

A-B.2. Deformation and Strain

Deformation is the transformation of a body from a reference configuration to a current configuration. It includes changes of size and shape.

A-B.2.1 Material deformation gradient tensor

It is a primary measure of deformation

$$\begin{cases} \underline{\underline{F}}(\underline{X},t) = \underline{x}\,\underline{\nabla} = \dfrac{\partial \underline{x}(\underline{X},t)}{\partial \underline{X}} \\ F_{ij} = \dfrac{\partial x_i}{\partial X_j} \end{cases} \rightarrow \quad \begin{matrix} \boldsymbol{d\underline{x} = \underline{\underline{F}} \cdot d\underline{X}} \\ \boldsymbol{dx_i = F_{ij}\, dX_j} \end{matrix}$$

Example in three dimensions cartesian coordinates:

$$\begin{pmatrix} dx_1 \\ dx_2 \\ dx_3 \end{pmatrix} = \begin{pmatrix} \dfrac{\partial x_1}{\partial X_1} & \dfrac{\partial x_1}{\partial X_2} & \dfrac{\partial x_1}{\partial X_3} \\ \dfrac{\partial x_2}{\partial X_1} & \dfrac{\partial x_2}{\partial X_2} & \dfrac{\partial x_2}{\partial X_3} \\ \dfrac{\partial x_3}{\partial X_1} & \dfrac{\partial x_3}{\partial X_2} & \dfrac{\partial x_3}{\partial X_3} \end{pmatrix} \cdot \begin{pmatrix} dX_1 \\ dX_2 \\ dX_3 \end{pmatrix}$$

A-B.2.2. Strain tensors

Strain is a normalized measure of deformation which characterizes the changes of distances and angles between particles. It reduces to zero when there is no change of distances and angles between particles.

A-B.2.2.1. Euler-Almansi or Spatial Strain Tensor

$$\begin{cases} \underline{\underline{e}}(\underline{x},t) = \dfrac{1}{2}\left(\underline{\underline{\delta}} - \underline{\underline{F}}^{-T} \cdot \underline{\underline{F}}^{-1}\right) \\ e_{ij} = \dfrac{1}{2}\left(\delta_{ij} - {F_{ki}}^{-1}\, {F_{kj}}^{-1}\right) = \dfrac{1}{2}\left[\underline{u}\,\underline{\nabla} + \underline{\nabla}\,\underline{u} - (\underline{u}\,\underline{\nabla}) \cdot (\underline{\nabla}\,\underline{u})\right] \end{cases}$$

Where "$\underline{u}$" is the spatial description (Eulerian form) of the displacements

It is a symmetric tensor:

$$e_{ij} = e_{ji}$$

A-B.2.2.2. Green-Lagrange or Material Strain Tensor

$$\begin{cases} \underline{\underline{E}}(\underline{X},t) = \dfrac{1}{2}\left(\underline{\underline{F}}^{T} \cdot \underline{\underline{F}} - \underline{\underline{\delta}}\right) = \dfrac{1}{2}\left[\underline{U}\,\underline{\nabla} + \underline{\nabla}\,\underline{U} + (\underline{U}\,\underline{\nabla}) \cdot (\underline{\nabla}\,\underline{U})\right] \\ E_{ij} = \dfrac{1}{2}\left(F_{ki}\, F_{kj} - \delta_{ij}\right) = \dfrac{1}{2}\left(\dfrac{\partial U_i}{\partial X_j} + \dfrac{\partial U_j}{\partial X_i} + \dfrac{\partial U_k}{\partial X_i}\dfrac{\partial U_k}{\partial X_j}\right) \end{cases}$$

Where "$\underline{U}$" is the material description (Lagrangian form) of the displacements

It is a symmetric tensor:

$$E_{ij} = E_{ji}$$

Example in three dimensions cartesian coordinates:

$$E_{ij} = \begin{pmatrix} \frac{\partial U_1}{\partial X_1} + \frac{1}{2}\left[\left(\frac{\partial U_1}{\partial X_1}\right)^2 + \left(\frac{\partial U_2}{\partial X_1}\right)^2 + \left(\frac{\partial U_3}{\partial X_1}\right)^2\right] & \frac{1}{2}\left(\frac{\partial U_1}{\partial X_2} + \frac{\partial U_2}{\partial X_1} + \frac{\partial U_1}{\partial X_1}\frac{\partial U_1}{\partial X_2} + \frac{\partial U_2}{\partial X_1}\frac{\partial U_2}{\partial X_2} + \frac{\partial U_3}{\partial X_1}\frac{\partial U_3}{\partial X_2}\right) & \frac{1}{2}\left(\frac{\partial U_1}{\partial X_3} + \frac{\partial U_3}{\partial X_1} + \frac{\partial U_1}{\partial X_1}\frac{\partial U_1}{\partial X_3} + \frac{\partial U_2}{\partial X_1}\frac{\partial U_2}{\partial X_3} + \frac{\partial U_3}{\partial X_1}\frac{\partial U_3}{\partial X_3}\right) \\ \frac{1}{2}\left(\frac{\partial U_2}{\partial X_1} + \frac{\partial U_1}{\partial X_2} + \frac{\partial U_1}{\partial X_2}\frac{\partial U_1}{\partial X_1} + \frac{\partial U_2}{\partial X_2}\frac{\partial U_2}{\partial X_1} + \frac{\partial U_3}{\partial X_2}\frac{\partial U_3}{\partial X_1}\right) & \frac{\partial U_2}{\partial X_2} + \frac{1}{2}\left[\left(\frac{\partial U_1}{\partial X_2}\right)^2 + \left(\frac{\partial U_2}{\partial X_2}\right)^2 + \left(\frac{\partial U_3}{\partial X_2}\right)^2\right] & \frac{1}{2}\left(\frac{\partial U_2}{\partial X_3} + \frac{\partial U_3}{\partial X_2} + \frac{\partial U_1}{\partial X_2}\frac{\partial U_1}{\partial X_3} + \frac{\partial U_2}{\partial X_2}\frac{\partial U_2}{\partial X_3} + \frac{\partial U_3}{\partial X_2}\frac{\partial U_3}{\partial X_3}\right) \\ \frac{1}{2}\left(\frac{\partial U_3}{\partial X_1} + \frac{\partial U_1}{\partial X_3} + \frac{\partial U_1}{\partial X_3}\frac{\partial U_1}{\partial X_1} + \frac{\partial U_2}{\partial X_3}\frac{\partial U_2}{\partial X_1} + \frac{\partial U_3}{\partial X_3}\frac{\partial U_3}{\partial X_1}\right) & \frac{1}{2}\left(\frac{\partial U_3}{\partial X_2} + \frac{\partial U_2}{\partial X_3} + \frac{\partial U_1}{\partial X_3}\frac{\partial U_1}{\partial X_2} + \frac{\partial U_2}{\partial X_3}\frac{\partial U_2}{\partial X_2} + \frac{\partial U_3}{\partial X_3}\frac{\partial U_3}{\partial X_2}\right) & \frac{\partial U_3}{\partial X_3} + \frac{1}{2}\left[\left(\frac{\partial U_1}{\partial X_3}\right)^2 + \left(\frac{\partial U_2}{\partial X_3}\right)^2 + \left(\frac{\partial U_3}{\partial X_3}\right)^2\right] \end{pmatrix}$$

A-B.2.2.3. Infinitesimal Strain Tensor

The infinitesimal strain theory (also called small strain theory) is based on the simplifying hypothesis:

1) Displacements are very small

2) Displacements gradients re infinitesimal

These assumptions lead to the material and spatial coordinates coincide, no difference between the material and spatial differential operators and local and material time derivatives coincide.

$$\begin{cases} \underline{\underline{\varepsilon}} = \frac{1}{2}\left(\underline{u}\,\underline{\nabla} + \underline{\nabla}\,\underline{u}\right) = \underline{\nabla}^s\,\underline{u} \\ \varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = \frac{1}{2}\left(\frac{\partial U_i}{\partial X_j} + \frac{\partial U_j}{\partial X_i}\right) \end{cases}$$

Example in three dimensions cartesian coordinates:

$$\varepsilon_{ij} = \begin{pmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{pmatrix} = \begin{pmatrix} \varepsilon_1 & \frac{1}{2}\gamma_{12} & \frac{1}{2}\gamma_{13} \\ \frac{1}{2}\gamma_{21} & \varepsilon_2 & \frac{1}{2}\gamma_{23} \\ \frac{1}{2}\gamma_{31} & \frac{1}{2}\gamma_{32} & \varepsilon_3 \end{pmatrix} = \begin{pmatrix} \frac{\partial u_1}{\partial x_1} & \frac{1}{2}\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}\right) & \frac{1}{2}\left(\frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1}\right) \\ \frac{1}{2}\left(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2}\right) & \frac{\partial u_2}{\partial x_2} & \frac{1}{2}\left(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2}\right) \\ \frac{1}{2}\left(\frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3}\right) & \frac{1}{2}\left(\frac{\partial u_3}{\partial x_2} + \frac{\partial u_2}{\partial x_3}\right) & \frac{\partial u_3}{\partial x_3} \end{pmatrix} = [-]$$

A diagram will help the understanding of these three types of tensioners:

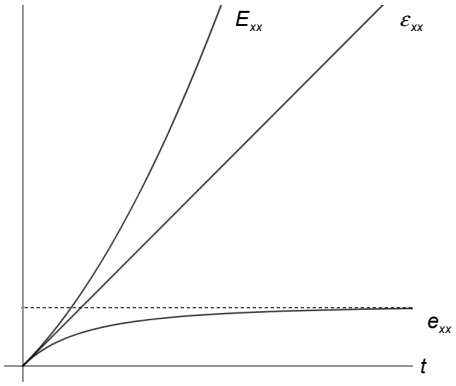


Figure A-B.0.1: Strain Tensor plot. Source: [6]

A-B.2.3. Strain rate

A-B.2.3.1. Spatial velocity gradient tensor

$$\begin{cases} \underline{\underline{l}}(\underline{x},t) = \underline{v}\,\underline{\nabla} = (\underline{\nabla}\,\underline{v})^T = \dfrac{\partial \underline{v}(\underline{x},t)}{\partial \underline{x}} \\ \\ l_{ij} = \dfrac{\partial v_i}{\partial x_j} = v_{i,j} \end{cases} \rightarrow \quad \begin{matrix} \boldsymbol{d\underline{v} = \underline{\underline{l}} \cdot d\underline{x}} \\ \\ \boldsymbol{dv_i = l_{ij}\, dx_j} \end{matrix}$$

Example in three dimensions cartesian coordinates:

$$\begin{pmatrix} dv_1 \\ dv_2 \\ dv_3 \end{pmatrix} = \begin{pmatrix} \dfrac{\partial v_1}{\partial x_1} & \dfrac{\partial v_1}{\partial x_2} & \dfrac{\partial v_1}{\partial x_3} \\ \dfrac{\partial v_2}{\partial x_1} & \dfrac{\partial v_2}{\partial x_2} & \dfrac{\partial v_2}{\partial x_3} \\ \dfrac{\partial v_3}{\partial X_1} & \dfrac{\partial v_3}{\partial x_2} & \dfrac{\partial v_3}{\partial x_3} \end{pmatrix} \cdot \begin{pmatrix} dx_1 \\ dx_2 \\ dx_3 \end{pmatrix}$$

The spatial velocity gradient tensor can be split into a symmetrical and skew-symmetrical tensor:

$$l_{ij} = sym\,(l_{ij}) + skew\,(l_{ij}) = \frac{1}{2}(l_{ij} + l_{ji}) + \frac{1}{2}(l_{ij} - l_{ji}) = d_{ij} + w_{ij}$$

A-B.2.3.2. Strain rate tensor

$$\begin{cases} \underline{\underline{d}}(\underline{x},t) = \dfrac{1}{2}(\underline{v}\,\underline{\nabla} + \underline{\nabla}\,\underline{v}) = \underline{\nabla}^s\,\underline{v} \\ \\ d_{ij} = \dfrac{1}{2}\left(\dfrac{\partial v_i}{\partial x_j} + \dfrac{\partial v_j}{\partial x_i}\right) \end{cases}$$

Example in three dimensions cartesian coordinates:

$$d_{ij} = \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix} = \begin{pmatrix} \frac{\partial v_1}{\partial x_1} & \frac{1}{2}\left(\frac{\partial v_1}{\partial x_2} + \frac{\partial v_2}{\partial x_1}\right) & \frac{1}{2}\left(\frac{\partial v_1}{\partial x_3} + \frac{\partial v_3}{\partial x_1}\right) \\ \frac{1}{2}\left(\frac{\partial v_2}{\partial x_1} + \frac{\partial v_1}{\partial x_2}\right) & \frac{\partial v_2}{\partial x_2} & \frac{1}{2}\left(\frac{\partial v_2}{\partial x_3} + \frac{\partial v_3}{\partial x_2}\right) \\ \frac{1}{2}\left(\frac{\partial v_3}{\partial x_1} + \frac{\partial v_1}{\partial x_3}\right) & \frac{1}{2}\left(\frac{\partial v_3}{\partial x_2} + \frac{\partial v_2}{\partial x_3}\right) & \frac{\partial v_3}{\partial x_3} \end{pmatrix} = \left[\frac{1}{s}\right]$$

An interesting property to note is:

$$\underline{\nabla} \cdot \underline{v} = tr\left(\underline{\underline{d}}\right) = \underline{\underline{d}} : \underline{\underline{\delta}} = d_{ij}\,\delta_{ij} = d_{ii} = \frac{\partial v_i}{\partial x_i} = v_{i,i} = \frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2} + \frac{\partial v_3}{\partial x_3}$$

If the infinitesimal strain tensor time derivative is computed:

$$\frac{d\varepsilon_{ij}}{dt} = \underline{\underline{\dot{\varepsilon}}} = \dot{\boldsymbol{\varepsilon}}_{\boldsymbol{ij}} = \frac{1}{2}\frac{d}{dt}\left(u_{i,j} + u_{j,i}\right) = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right)_{,t} = \frac{1}{2}\left(u_{i,jt} + u_{j,it}\right) = \frac{1}{2}\left(v_{i,j} + v_{j,i}\right) = \boldsymbol{d}_{\boldsymbol{ij}} = \underline{\underline{d}}$$

In engineering notation, it is usual to use this type of symbology for the strain rate tensor:

$$\underline{\dot{\gamma}} = \dot{\gamma}_{ij} = \left(\underline{v}\,\underline{\nabla} + \underline{\nabla}\,\underline{v}\right) = \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) = \left[\frac{1}{s}\right]$$

It is from here that one notation or another is justified:

$$\begin{cases} \underline{\underline{d}} = \underline{\underline{\dot{\varepsilon}}} = \frac{1}{2}\underline{\dot{\gamma}} \\ \\ d_{ij} = \dot{\varepsilon}_{ij} = \frac{1}{2}\dot{\gamma}_{ij} \end{cases}$$

A-B.2.3.3. Rotation rate (or Spin) tensor

$$\begin{cases} \underline{\underline{w}}(\underline{x}, t) = \frac{1}{2}\left(\underline{v}\,\underline{\nabla} - \underline{\nabla}\,\underline{v}\right) = \underline{\nabla}^a\,\underline{v} \\ \\ w_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right) \end{cases}$$

Example in three dimensions cartesian coordinates:

$$w_{ij} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2}\left(\frac{\partial v_1}{\partial x_2} - \frac{\partial v_2}{\partial x_1}\right) & \frac{1}{2}\left(\frac{\partial v_1}{\partial x_3} - \frac{\partial v_3}{\partial x_1}\right) \\ \frac{1}{2}\left(\frac{\partial v_2}{\partial x_1} - \frac{\partial v_1}{\partial x_2}\right) & 0 & \frac{1}{2}\left(\frac{\partial v_2}{\partial x_3} - \frac{\partial v_3}{\partial x_2}\right) \\ \frac{1}{2}\left(\frac{\partial v_3}{\partial x_1} - \frac{\partial v_1}{\partial x_3}\right) & \frac{1}{2}\left(\frac{\partial v_3}{\partial x_2} - \frac{\partial v_2}{\partial x_3}\right) & 0 \end{pmatrix} = \left[\frac{1}{s}\right]$$

An interesting property to note is:

$$\boldsymbol{tr}\left(\underline{\underline{\boldsymbol{w}}}\right) = \underline{\underline{w}} : \underline{\underline{\delta}} = w_{ij}\,\delta_{ij} = w_{ii} = \mathbf{0}$$

Therefore:

$$\boldsymbol{tr}\left(\underline{\underline{\boldsymbol{l}}}\right) = tr\left(\underline{\underline{d}}\right) + tr\left(\underline{\underline{w}}\right) = \underline{\boldsymbol{\nabla}} \cdot \underline{\boldsymbol{v}}$$

$$d\underline{v} = \boldsymbol{Stretch\ velocity} + \boldsymbol{Rotation\ velocity} = \left(\underline{\underline{d}} \cdot d\underline{x}\right) + \left(\underline{\underline{w}} \cdot d\underline{x}\right)$$

A-B.3. Balance Principles

These principles are always valid, regardless the material type and the displacements or deformation range.

A-B.3.1. Conservation of mass

It is postulated that during a motion there are neither mass sources nor mass sinks, so the mass of a continuum body is a conserved quantify (for any part of the body).

$$\left.\begin{array}{l} Density \rightarrow \rho = \dfrac{dm}{dV} \rightarrow m = \displaystyle\int \rho\, dV \\ \boldsymbol{Mass\ conservation} \rightarrow \boldsymbol{\dfrac{dm}{dt} = 0} \end{array}\right\} \rightarrow \frac{d}{dt}\int \rho\, dV = 0$$

A-B.3.1.1. Spatial form:

$$Global\ or\ Integral\ spatial\ form \rightarrow \frac{d}{dt}\int_V \rho\, dV = \int_V \left[\frac{d\rho}{dt} + \rho\left(\underline{\nabla} \cdot \underline{v}\right)\right] dV = 0$$

$$Local\ or\ Differential\ spatial\ form \rightarrow \begin{cases} \boldsymbol{\dfrac{d\rho}{dt} + \rho\left(\underline{\boldsymbol{\nabla}} \cdot \underline{\boldsymbol{v}}\right) = \dfrac{\partial \rho}{\partial t} + \underline{\nabla} \cdot \left(\rho\, \underline{v}\right) = 0} \\ \boldsymbol{\dot{\rho} + \rho\, v_{i,i} = \rho_{,t} + \left(\rho\, v_i\right)_{,i} = 0} \end{cases} \quad \forall \underline{x} \in V\ ,\ \forall t \in \mathbb{R}^+$$

A very interesting way to rewrite the equation and which will be used in this work is in pressure "p" terms:

Mass Conservation Equation (**Quasi-Incompressible** Formulation):

$$\boldsymbol{Bulk\ Modulus} \rightarrow \boldsymbol{\kappa = \rho \frac{dp}{d\rho}} = [Pa]$$

$$\frac{d\rho}{dt} + \rho\left(\underline{\nabla} \cdot \underline{v}\right) = 0 \rightarrow \frac{d\rho}{dp}\frac{dp}{dt} + \rho\left(\underline{\nabla} \cdot \underline{v}\right) = 0 \rightarrow \frac{d\rho}{dp}\dot{p} + \rho\left(\underline{\nabla} \cdot \underline{v}\right) = 0 \rightarrow \frac{1}{\rho}\frac{d\rho}{dp}\dot{p} + \underline{\nabla} \cdot \underline{v} = 0 \rightarrow \boldsymbol{\frac{1}{\kappa}\dot{p} + \underline{\boldsymbol{\nabla}} \cdot \underline{\boldsymbol{v}} = 0}$$

Many of the studies in the fluid dynamics field have treated its behavior as incompressible:

$$\boldsymbol{Incompressible\ medium} \quad \leftrightarrow \quad \begin{cases} \dfrac{d\rho(\underline{x},t)}{dt} = 0 \\ \\ \dot{\rho} = 0 \end{cases}$$

Hence, the mass continuity equation is simplified as:

$$\boldsymbol{Mass\ conservation\ for\ incompressible\ medium} \quad \rightarrow \quad \begin{cases} \underline{\nabla} \cdot \underline{v} = tr\left(\underline{\underline{d}}\right) = 0 \\ \\ \dfrac{\partial v_i}{\partial x_i} = d_{ii} = 0 \end{cases} \quad \rightarrow \quad \underline{\underline{d}}^{sph} = \underline{\underline{0}} \ \rightarrow \ \underline{\underline{d}} = \underline{\underline{d}}'$$

A-B.3.1.2. Material form:

$$Global\ material\ form \quad \rightarrow \quad \int_V \frac{\partial}{\partial t}\left(\rho \left|\underline{\underline{F}}\right|\right) dV_o = 0$$

$$Local\ material\ form \quad \rightarrow \quad \rho_o = \rho(t_o) = \rho_t \left|\underline{\underline{F}}\right|_t = \rho(t) \left|\underline{\underline{F}}\right|_t$$

A-B.3.2. Linear momentum balance

The Cauchy's motion equations proceed from the development of Newton's second law (the resulting force acting on the medium).

$$\begin{cases} \underline{\nabla} \cdot \underline{\underline{\sigma}} + \rho\, \underline{b} = \rho \dfrac{d\underline{v}}{dt} \\ \\ \sigma_{ij,j} + \rho\, b_i = \rho \dot{v}_i \end{cases} \qquad \forall \underline{x} \in V\,, \ \forall t \in \mathbb{R}^+$$

Where "$\underline{b}$" is the spatial description of the vector field of the body forces per unit of mass. It can be taken as:

$$\underline{b}(\underline{x},t) = \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} \qquad where\ \ g = 9{,}81 \left[\frac{m}{s^2}\right]$$

A-B.3.3. Angular momentum balance

This balance proceeds from the development of Newton's second law (the resulting torque acting on the medium). This results in symmetry of the Cauchy stress tensor:

$$\begin{cases} \underline{\underline{\sigma}} = \underline{\underline{\sigma}}^T \\ \\ \sigma_{ij} = \sigma_{ji} \end{cases} \qquad \forall \underline{x} \in V\,, \ \forall t \in \mathbb{R}^+$$

A-B.4. Constitutive equations in fluids

Pascal law establish that in a confined fluid at rest (no shear stresses), pressure acts equally in all directions at a given point.

The stress in a fluid at rest is isotropic (has the same properties in all directions) and must be of the form:

$$\begin{cases} \underline{\underline{\sigma}} = -p_o \, \underline{\underline{\delta}} \\ \sigma_{ij} = -p_o \, \delta_{ij} \end{cases}$$

Where "p_o" is the **hydrostatic pressure**, the normal compressive stress exerted on a fluid in equilibrium.

$$Mean\ Pressure \quad \rightarrow \quad p_m = \overline{p} = -\sigma_m = \begin{cases} -\frac{1}{3} tr\left(\underline{\underline{\sigma}}\right) = \frac{1}{3}\left(\underline{\underline{\sigma}} : \underline{\underline{\delta}}\right) \\ -\frac{1}{3}\sigma_{ij}\,\delta_{ij} = \frac{1}{3}\sigma_{ii} \end{cases}$$

$$Thermodynamic\ Pressure \quad \rightarrow \quad p = f(\rho, p, \theta)$$

A-B.4.1. Constitutive equations

The general form of thermo-mechanical constitutive equation for a viscous fluid:

$$\begin{cases} \underline{\underline{\sigma}} = -p\,\underline{\underline{\delta}} + \underline{\underline{\tau}} = -p\,\underline{\underline{\delta}} + \underline{\underline{f}}\left(\underline{\underline{d}}, \rho, \theta\right) = -p\,\underline{\underline{\delta}} + \underline{\underline{\underline{\underline{\mathbb{K}}}}} : \underline{\underline{d}} \\ \sigma_{ij} = -p\,\delta_{ij} + \tau_{ij} = p\,\delta_{ij} + f_{ij}(d_{ij}, \rho, \theta) = -p\,\delta_{ij} + \mathbb{K}_{ijkl}\,d_{kl} \end{cases}$$

*Where "p" is the **thermodynamic pressure**, "$\underline{\underline{\tau}}$" is **viscous stress tensor** and "$\underline{\underline{f}}$" is a symmetrical tensor function:*

$$\underline{\underline{f}} : \mathbb{R}^3 \times \mathbb{R}^+ \rightarrow \mathbb{R}^{n \times n}$$

Type	Description
Stokesian / Non-Newtonian fluid	$\underline{\underline{f}}$ is a non-linear function of its arguments
Newtonian fluid	$\underline{\underline{f}}$ is a linear function of its arguments
Perfect / Non-Viscous fluid	$\underline{\underline{f}}$ is null

An interesting relation extracted from the calculation of the deflection trace of the above stress tensor is:

$$\begin{cases} p = \overline{p} + \mathcal{K}\, tr\left(\underline{\underline{d}}\right) = \overline{p} + \mathcal{K}\left(\underline{\underline{d}} : \underline{\underline{1}}\right) = \overline{p} + \mathcal{K}\left(\underline{\nabla} \cdot \underline{v}\right) \\ p = \overline{p} + \mathcal{K}\, d_{ii} = \overline{p} + \mathcal{K}\, v_{i,i} \end{cases}$$

Newtonian Fluid

If the fluid is Newtonian and also isotropic, the following definition can be used:

$$Generic\ Isotropic\ Tensor \rightarrow \begin{cases} \underline{\underline{\underline{\underline{\mathbb{A}}}}} = a_1\ \overline{\overline{\mathbb{I}}} + a_2\ \mathbb{I} + a_3\ \overline{\mathbb{I}} \\ \mathbb{A}_{ijkl} = a_1\ \delta_{ij}\delta_{kl} + a_2\ \delta_{ik}\delta_{jl} + a_3\ \delta_{il}\delta_{jk} \end{cases}$$

At the time, the following fourth order tensor was proposed to model the constitutive law:

$$\begin{cases} \underline{\underline{\underline{\underline{\mathbb{K}}}}} = \lambda\ \underline{\underline{\delta}} \otimes \underline{\underline{\delta}} + 2\mu\ \underline{\underline{\underline{\underline{I}}}} \\ \mathbb{K}_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) \end{cases}$$

Therefore:

$$\mathbb{K}_{ijkl}\ d_{kl} = \lambda\ d_{kk}\ \delta_{ij} + 2\ \mu\ d_{ij}$$

Then the stress tensor in an isotropic Newtonian fluid:

$$\begin{cases} \underline{\underline{\boldsymbol{\sigma}}} = -\boldsymbol{p}\ \underline{\underline{\mathbf{1}}} + \boldsymbol{\lambda}\left(\underline{\underline{\boldsymbol{d}}} : \underline{\underline{\mathbf{1}}}\right)\underline{\underline{\mathbf{1}}} + \mathbf{2}\ \boldsymbol{\mu}\ \underline{\underline{\boldsymbol{d}}} \\ \boldsymbol{\sigma_{ij}} = -\boldsymbol{p}\ \boldsymbol{\delta_{ij}} + \boldsymbol{\lambda}\ \boldsymbol{d_{kk}}\ \boldsymbol{\delta_{ij}} + \mathbf{2}\ \boldsymbol{\mu}\ \boldsymbol{d_{ij}} \end{cases}$$

Where "λ" and the "μ" are coefficients. They are not necessarily constant, both could be function of "p" and "θ", for instance.

From it, it follows that:

$$If\ incompressible\ medium \rightarrow p = \overline{p} = p_o$$

Another way to

$$\begin{cases} \underline{\underline{\boldsymbol{\sigma}}} = \underline{\underline{\boldsymbol{\sigma}}}^{\boldsymbol{sph}} + \underline{\underline{\boldsymbol{\sigma}}}' = -\overline{\boldsymbol{p}}\ \underline{\underline{\boldsymbol{\delta}}} + \mathbf{2}\ \boldsymbol{\mu}\ \underline{\underline{\boldsymbol{d}}}' = -\overline{\boldsymbol{p}}\ \underline{\underline{\boldsymbol{\delta}}} + \underline{\underline{\boldsymbol{\tau}}} \\ \boldsymbol{\sigma_{ij}} = \boldsymbol{\sigma_{ij}}^{\boldsymbol{sph}} + \boldsymbol{\sigma_{ij}}' = -\overline{\boldsymbol{p}}\ \boldsymbol{\delta_{ij}} + \mathbf{2}\ \boldsymbol{\mu}\ \boldsymbol{d_{ij}}' = -\overline{\boldsymbol{p}}\ \boldsymbol{\delta_{ij}} + \boldsymbol{\tau_{ij}} \end{cases}$$

Where "μ" is the viscosity, measure in $[Pa \cdot s]$

The **viscosity** is the molecules' ability to move relative to each other. It's the flow resistance. There exist two kinds of viscosities:

1) Absolute or Dynamic viscosity

$$\mu = [Pa \cdot s] = \left[\frac{kg}{m \cdot s}\right]$$

2) Kinetic viscosity

$$\boldsymbol{\nu} = \frac{Dynamic\ viscosity}{Fluid\ density} = \frac{\boldsymbol{\mu}}{\boldsymbol{\rho}} = \left[\frac{m^2}{s}\right]$$

A-B.5. Fluid mechanics

A-B.5.1. Energy equation

The energy equations for Newtonian fluids arise from the union of the internal energy balance with the definition of the stress tensor for Newtonian fluids:

$$\begin{cases} \boldsymbol{\rho}\dfrac{\boldsymbol{du}}{\boldsymbol{dt}} = -\boldsymbol{p}\left(\underline{\boldsymbol{\nabla}}\cdot\underline{\boldsymbol{v}}\right) + \boldsymbol{\rho r} + \underline{\boldsymbol{\nabla}}\cdot\left(\underline{\underline{\boldsymbol{K}}}\,\underline{\boldsymbol{\nabla}}\boldsymbol{\theta}\right) + \mathcal{K}\left(\underline{\boldsymbol{\nabla}}\cdot\underline{\boldsymbol{v}}\right)^2 + 2\,\boldsymbol{\mu}\left(\underline{\underline{\boldsymbol{d}}}' : \underline{\underline{\boldsymbol{d}}}'\right) \\ \\ \boldsymbol{\rho}\,\dot{\boldsymbol{u}} = -\boldsymbol{p}\,\boldsymbol{v}_{i,i} + \boldsymbol{\rho r} + \left(\boldsymbol{K}_{ij}\,\boldsymbol{\theta}_{,j}\right)_{,i} + \mathcal{K}\left(\boldsymbol{v}_{i,i}\right)^2 + 2\,\boldsymbol{\mu}\,\boldsymbol{d}_{ij}'\,\boldsymbol{d}_{ij}' \end{cases} \qquad \forall \underline{x} \in V\,,\ \forall t \in \mathbb{R}^+$$

A-B.5.2. Navier-Stokes equations

The Navier-Stokes equations arise from the union of the linear momentum balance with the definition of the stress tensor for Newtonian fluids:

$$\begin{cases} -\underline{\boldsymbol{\nabla}}\boldsymbol{p} + \underline{\boldsymbol{\nabla}}\boldsymbol{\lambda}\left(\underline{\boldsymbol{\nabla}}\cdot\underline{\boldsymbol{v}}\right) + (\boldsymbol{\lambda}+\boldsymbol{\mu})\,\underline{\boldsymbol{\nabla}}\left(\underline{\boldsymbol{\nabla}}\cdot\underline{\boldsymbol{v}}\right) + \underline{\boldsymbol{\nabla}}\boldsymbol{\mu}\cdot\left(\underline{\boldsymbol{v}}\,\underline{\boldsymbol{\nabla}} + \underline{\boldsymbol{\nabla}}\,\underline{\boldsymbol{v}}\right) + \boldsymbol{\mu}\,\underline{\boldsymbol{\nabla}}^2\,\underline{\boldsymbol{v}} + \boldsymbol{\rho}\underline{\boldsymbol{b}} = \boldsymbol{\rho}\dfrac{\boldsymbol{d}\underline{\boldsymbol{v}}}{\boldsymbol{dt}} \\ \\ -\boldsymbol{p}_{,i} + \boldsymbol{\lambda}_{,i}\,\boldsymbol{v}_{j,j} + (\boldsymbol{\lambda}+\boldsymbol{\mu})\boldsymbol{v}_{j,ji} + \boldsymbol{\mu}_{,j}\left(\boldsymbol{v}_{i,j} + \boldsymbol{v}_{j,i}\right) + \boldsymbol{\mu}\,\boldsymbol{v}_{i,jj} + \boldsymbol{\rho b}_i = \boldsymbol{\rho}\dot{\boldsymbol{v}}_i \end{cases} \qquad \forall \underline{x} \in V\,,\ \forall t \in \mathbb{R}^+$$

Where "λ" is the second viscosity, composed by the bulk viscosity "$\mathcal{K}$" (do not confuse with bulk modulus "κ") and the dynamic viscosity "μ":

$$\boldsymbol{\lambda} = \boldsymbol{\mathcal{K}} - \frac{\boldsymbol{2}}{\boldsymbol{3}}\boldsymbol{\mu} = [\boldsymbol{Pa}\cdot\boldsymbol{s}]$$

Example in three dimensions cartesian coordinates, with second and bulk viscosity as constants:

$$-\frac{\partial p}{\partial x_1} + (\lambda+\mu)\left(\frac{\partial^2 v_1}{\partial {x_1}^2} + \frac{\partial^2 v_2}{\partial x_1 \partial x_2} + \frac{\partial^2 v_3}{\partial x_1 \partial x_3}\right) + \mu\left(\frac{\partial^2 v_1}{\partial {x_1}^2} + \frac{\partial^2 v_1}{\partial {x_2}^2} + \frac{\partial^2 v_1}{\partial {x_3}^2}\right) + \rho b_1 = \rho\left(\frac{\partial v_1}{\partial t} + v_1\frac{\partial v_1}{\partial x_1} + v_2\frac{\partial v_1}{\partial x_2} + v_3\frac{\partial v_1}{\partial x_3}\right) = \left[\frac{N}{m^3}\right]$$

$$-\frac{\partial p}{\partial x_2} + (\lambda+\mu)\left(\frac{\partial^2 v_1}{\partial x_2 \partial x_1} + \frac{\partial^2 v_2}{\partial {x_2}^2} + \frac{\partial^2 v_3}{\partial x_2 \partial x_3}\right) + \mu\left(\frac{\partial^2 v_2}{\partial {x_1}^2} + \frac{\partial^2 v_2}{\partial {x_2}^2} + \frac{\partial^2 v_2}{\partial {x_3}^2}\right) + \rho b_2 = \rho\left(\frac{\partial v_2}{\partial t} + v_1\frac{\partial v_2}{\partial x_1} + v_2\frac{\partial v_2}{\partial x_2} + v_3\frac{\partial v_2}{\partial x_3}\right) = \left[\frac{N}{m^3}\right]$$

$$-\frac{\partial p}{\partial x_3} + (\lambda+\mu)\left(\frac{\partial^2 v_1}{\partial x_3 \partial x_1} + \frac{\partial^2 v_2}{\partial x_3 \partial x_2} + \frac{\partial^2 v_3}{\partial {x_3}^2}\right) + \mu\left(\frac{\partial^2 v_3}{\partial {x_1}^2} + \frac{\partial^2 v_3}{\partial {x_2}^2} + \frac{\partial^2 v_3}{\partial {x_3}^2}\right) + \rho b_3 = \rho\left(\frac{\partial v_3}{\partial t} + v_1\frac{\partial v_3}{\partial x_1} + v_2\frac{\partial v_3}{\partial x_2} + v_3\frac{\partial v_3}{\partial x_3}\right) = \left[\frac{N}{m^3}\right]$$

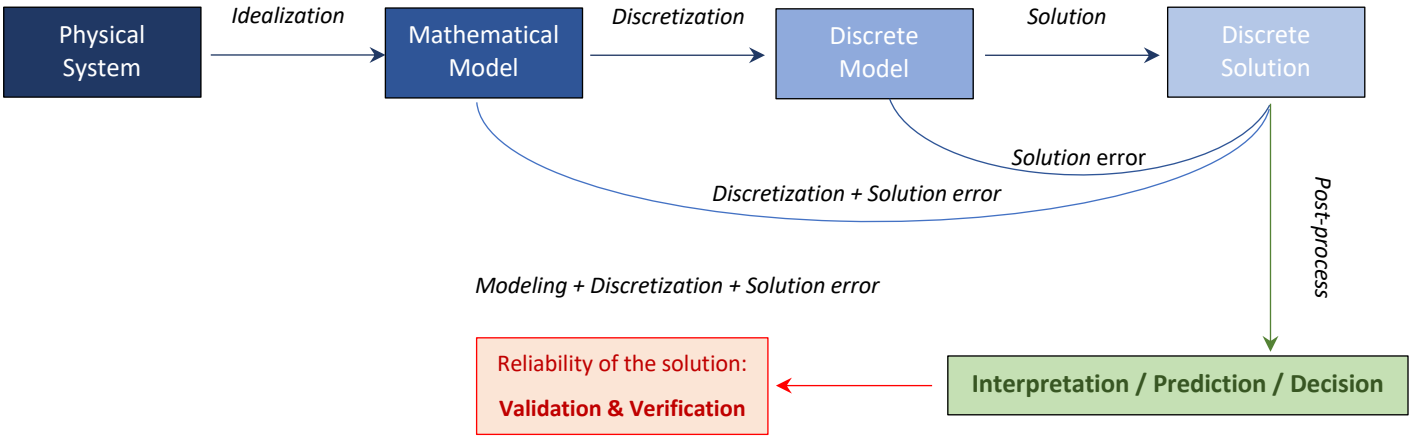
As can be seen, it begins to be really complex to determine any result of these partial derivative equations.

Annex C. Numerical calculus

A-C.1. Computational engineering

Plenty of problems arising in mathematics and engineering cannot be solved analytically, for example, equations systems, Ordinary Differential Equations (O.D.E.), Partial Differential Equations (P.D.E.), computing roots functions, eigenvalues, integrals, etc. The main objective of numerical modeling/analysis (modeling + numerical solution + result analysis) is to provide effective methods to solve problems. The fact that numerical methods can be implemented on a digital computer makes it an essential tool in today's numerical studies. Thus, the use of the computer to solve scientific and engineering problems is increasing, and solutions are obtained from mathematical models that represent real concrete situations.

When computational engineering faces a new scientific-technological challenge resolution, it is guided by the following scheme:



Idealization is the process of capturing a physical phenomenon in a mathematical model, ideally as close as possible to the real world.
What does it mean to discretize (**discretization**) a medium? In Annex B you can see the modeling of physical realities such as mechanics and thermodynamics. However, all this theory is based on a fundamental principle: the continuity of the system, i.e., the whole medium is the same "*body*". Representing and calculating this computationally is not possible today. Hence the need to discretize the medium. It is nothing more than dividing the body into elements composed of *edges* and *nodes* (as if it were a **graph**). A machine is capable of understanding this and, moreover, it knows very well how to treat it. In the **Solution** section is where the numerical method for solving the discrete problem is applied. There are many methods of numerical computation. A classification of them is presented below.

Explicit method

It is based on the calculation of the state of a system at a previous time than the state of the system at the current time.

It consists of obtaining the unknown nodal variable for the new time, by means of the variables known in the previous time. Therefore, the calculation of the unknown variables is direct and independent of the value of the variable in neighboring nodes for the new calculated time.

$$\begin{cases} Goal \rightarrow Y(t+\Delta t) \\ Method \rightarrow By\ means\ of\ a\ function\ that\ implies\ the\ previous\ step\ to\ the\ searched\ one \end{cases}$$

$$Y(t+\Delta t) = f\,[Y(t)]$$

Implicit Method

It finds the solution by solving an equation involving both the current state of the system and the last one.

It consists of evaluating all the variables at the same time in the new time, instead of using the previous time as in the explicit method. Therefore, the value of a variable at a node and at a given time instant will depend on the variables of the contiguous nodes which, in general, are unknown, which makes it necessary to solve the system of equations simultaneously to obtain the solution.

$$\begin{cases} Goal \rightarrow Y(t+\Delta t) \\ Method \rightarrow By\ means\ of\ a\ function\ involving\ the\ prior\ step\ to\ the\ one\ and\ the\ one\ searched\ for \end{cases}$$

$$f\,[Y(t), Y(t+\Delta t)] = 0$$

Iterative Method

It is a mathematical procedure with the aim of approximating the solution to an equation or set of equations. It is based on an "*initial estimation*" and from it, it begins to develop, in an iterative (successive, repeating) way, an approximate solution that for each iteration (repetition) gets closer to the real solution.

As many iterations as needed are made so that the calculation error committed is lower than the tolerance value pre-set by oneself. Once the error is lower, then you go from "step" (which represents the time variable).

1st procedure → Iterations
2nd procedure → Tolerance check
3rd procedure → Next step

$$Iterations \in Steps$$

The **Validation & Verification** section is an essential part of this branch of engineering, because although it compiles a program and provides results, to this day, it is still up to the human mind to determine whether the model is accurate and meets the needs.

A-C.2. Variational Calculus

In **computational mechanic**s problems (computational engineering branch) are solved by cooperation of mechanics, computers and numerical methods. This provides an additional approach to problem-solving, besides the theoretical and experimental science. It includes disciplines such as solid mechanics, fluid dynamics, thermodynamics and electromagnetics.

Variational calculus is a mathematical tool that allows working with the so called integral or weak form of the governing differential equations of a problem (stated in Annex B). Given a differential equation system, which must be verified in local form (point by point, strong from or differential form) of a certain domain, the variational principles allow obtaining an integral formulation (global in the domain), whose imposition, nonetheless, guarantees that the aforementioned differential equations are satisfied. Integral formulations are of particular interest when treating and solving the problem by means of numerical methods.

In the traditional mechanics literature, any physical fact can be expressed as the integral of the domain of the Lagrangian of the function (resulting from the difference of the kinetic energy and the potential energy):

$$\mathcal{S} = \int_{t_o}^{t_f} \mathcal{L}(\underline{x}, \dot{\underline{x}}, t) dt$$

*By **the Principle of Least Action**, we can proceed in such a way that:* $\boldsymbol{\delta \mathcal{S} = 0}$

$$\delta \mathcal{S} = \int_{t_o}^{t_f} \delta \mathcal{L}(\underline{x}, \dot{\underline{x}}, t)\, dt = \int_{t_o}^{t_f} \left[\underline{\nabla}_{\underline{x}} \mathcal{L}(\underline{x}, \dot{\underline{x}}, t) \delta \underline{x} + \underline{\nabla}_{\dot{\underline{x}}} \mathcal{L}(\underline{x}, \dot{\underline{x}}, t) \delta \dot{\underline{x}} + \dot{\mathcal{L}}(\underline{x}, \dot{\underline{x}}, t) \delta t \right] dt = 0$$

Developing the integral by parts and applying the boundary conditions regarding the time variable, the expression becomes the well-known **Fundamental Equation of Classical Mechanics** (also known as **Euler-Lagrange Equations**)

$$\boldsymbol{\frac{d}{dt}\left(\underline{\nabla}_{\dot{\underline{x}}} \mathcal{L}\right) = \underline{\nabla}_{\underline{x}} \mathcal{L}}$$

Variational Principal

$$\delta \mathbb{F}(\underline{u}; \delta \underline{u}) = \int_{\Omega} \mathbb{E} \cdot \delta \underline{u}\, d\Omega + \int_{\Gamma_\sigma} \mathbb{T} \cdot \delta \underline{u}\, d\Gamma = 0 \quad , \quad \forall\, \delta \underline{u} \text{ " } \delta \underline{u}\big|_{\underline{X} \in \Gamma_u} = \underline{0}$$

Fundamental Theorem of Variational Calculus

The expression

$$\int_{\Omega} \mathbb{E}\left(\underline{x}, \underline{u}(\underline{x}), \underline{\nabla u}(\underline{x})\right) \cdot \delta \underline{u}\, d\Omega + \int_{\Gamma_\sigma} \mathbb{T}\left(\underline{x}, \underline{u}(\underline{x}), \underline{\nabla u}(\underline{x})\right) \cdot \delta \underline{u}\, d\Gamma = 0 \qquad \forall \delta \underline{u} \text{ " } \delta \underline{u}\big|_{\underline{X} \in \Gamma_u} = \underline{0}$$

is satisfied if and only if:

$$\begin{cases} \mathbb{E}\left(\underline{x}, \underline{u}(\underline{x}), \underline{\nabla u}(\underline{x})\right) = 0 \quad \forall \underline{x} \in \Omega \quad \rightarrow \quad \boldsymbol{Euler - Lagrange\ Equations} \\ \mathbb{T}\left(\underline{x}, \underline{u}(\underline{x}), \underline{\nabla u}(\underline{x})\right) = 0 \quad \forall \underline{x} \in \Gamma_\sigma \quad \rightarrow \quad Natural - Boundary\ Conditions \end{cases}$$

Where "$\delta \underline{u}$" is the virtual displacements.

Applying the theorem to Continuum mechanics:

$$\begin{cases} \mathbb{E} = \dfrac{d}{dt}\left(\underline{\nabla}_{\dot{\underline{x}}} \mathcal{L}\right) - \underline{\nabla}_{\underline{x}} \mathcal{L} = 0 \quad \forall \underline{x} \in \Omega \quad \rightarrow \quad Euler - Lagrange\ Equations \\ \mathbb{T} = \left(\underline{\nabla}_{\dot{\underline{x}}} \mathcal{L}\right)\Big|_{\underline{x} = \partial\Omega} = 0 \quad \forall \underline{x} \in \Gamma_\sigma \quad \rightarrow \quad Natural\ (Newmann)\ Conditions \end{cases}$$

A-C.2.1. Virtual Work Principal (VWP)

The VWP is a variational principle frequently applied in solid mechanics that can be interpreted as the search of an extrema of a functional of a displacement field "$\mathbb{W}(\underline{u})$", not necessarily known in its explicit form, whose variation (Gateaux derivative) "$\delta\mathbb{W}(\underline{u};\delta\underline{u})$" is known and is given by variational calculus fundamental theorem. Since the Euler-Lagrange equations of the VWP are the Cauchy's equation nd the equilibrium condition at the boundary, its imposition is completely equivalent (yet more convenient when solving the problem through numerical methods) to the imposition in local form of the aforementioned equation and receives the name of the **weak form** of these equations.

The constitutive equation does not intervene in the VWP formulation and the type of kinetics considered (finite or infinitesimal strains) is not distinguished either. Thus, the application of the VWP is not restricted by the type of constitutive equation chosen (elastic, elastoplastic, fluid, etc.) noy by kinematics (finite or infinitesimal strains) considered.

VIRTUAL WORK PRINCIPAL

$$\underbrace{\delta\mathbb{W}(\underline{u};\delta\underline{u})}_{\text{Total Virtual Work}} = \underbrace{\int_V \rho(\underline{b}-\underline{\dot{v}})\cdot\delta\underline{u}\,dV + \int_{\Gamma_\sigma}\underline{t}^*\cdot\delta\underline{u}\,d\Gamma}_{\text{External Virtual Work}} - \underbrace{\int_V \underline{\underline{\sigma}}:\underline{\underline{\nabla}}^s\delta\underline{u}\,dV}_{\text{Internal Virtual Work}} = 0$$

In very summarized notation, it is as follows:

$$\delta\mathbb{W} = \delta\mathbb{W}^{ext} - \delta\mathbb{W}^{int} = 0 \qquad \forall\,\delta\underline{u}\in\mathbb{V}_o$$

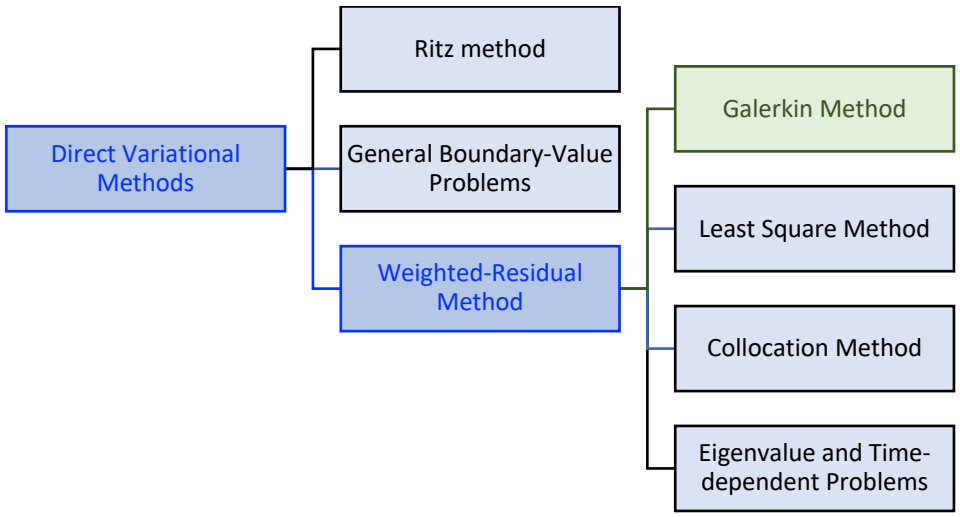


Figure A-C.0.1: Summary of classification of variational methods. Source: [26]

A-C.2.2. Galerkin Method

In [25] explain that the weighted-residual methods are those in which it seeks approximate solutions using weighted-integral statement of the equations.

Various special cases of the weighted-residual method differ from each other due to the choice of the wight function. The most commonly used weight functions are:

1) The Petrov-Galerkin
2) **Galerkin method**
3) Least-squares method
4) Collocation method

The Galerkin method is a special case of the Petrov-Galerkin method in which the approximation functions and the weighted functions are the same. Hence, the Galerkin integral is given by:

$$\int_{\Omega} \phi_i \, \mathcal{R}_n\left(\underline{x}, \{c\}, \{\phi\}, f\right) d\underline{x} = 0 \qquad i = 1, \dots, n$$

Where "$\mathcal{R}_n$" is the residual term of the equations.

$$\boldsymbol{Galerkin\ Method} \rightarrow Approximation\ functions\ \phi_i = Weighted\ functions\ \psi_i$$

If the Galerkin method is used for second-order or higher-order equations, it would involve the use of higher-order coordinate functions and the solution of nonsymmetric equations.

A-C.3. Finite Element Method (FEM)

The finite element method is a procedure that uses the philosophy of the traditional variational methods to derive the equations relating undetermined coefficients. However, the method differs in two ways from traditional variational methods in generating the equations of the problem. First, the approximation functions are often algebraic polynomials that are developed using ideas from interpolation theory; second, the approximation functions are developed for subdomains into which a given domain is divided. The subdomains, called finite elements, are geometrically simple shapes that permit a systematic a systematic construction of the approximation functions over the element.

The division of the whole domain into finite elements not only simplifies the task of generating the approximation functions, but also allows representation of the solution over individual elements. Thus, geometric and/or material discontinuities can be naturally included. Further, since the approximation functions are algebraic polynomials, the computation of the coefficients matrices of the approximation can be automated on a computer.

The construction of the approximation functions is systematic, and the process is independent of the boundary conditions and data of the problem. In short, the finite element method is a piecewise application of classical variational methods. The undetermined parameters often, but not always, represent values of the dependent variables at a finite number of preselected points, whose number and locations are dictated the degree and the form of the approximation functions used. The method is modular and therefore well suited for electronic computation and the development of general-purpose computer programs.

The domain is divided into a number of subdomains or intervals (**finite elements**). This is necessitated by one or both of the following reasons:

1) It is easier to represent the solution function, irrespective of the degree of its variation with its independent variables, by a polynomial of a desired degree over each element than to use a single polynomial to approximate the function over the entire domain.

$$\uparrow \textbf{number of elements} = \downarrow \textbf{error between the true solution and the piecewise linear solution}$$

2) The actual solution is defined piecewise because of the geometric and/or material discontinuities.

The collection of the elements is called the **finite element mesh**. The number of divisions is analogous to the number of parameters in the traditional variational methods, Therefore, the larger the number of elements, the more accurate the solution will be. The minimum number of subdivisions is equal to the number of subdivisions crated by the discontinuities n the data of the problem. An element or interval in one-dimensional problems is a line of finite length. An element or interval in two-dimensional problems is a surface of finite area. The same is true for three-dimensional problems.

The elements in a finite element mesh are connected to neighboring elements at a finite number of points, called **global nodes**. The word “global” refers to the whole problem as opposed to an element. The end points of individual elements are called **element nodes**, and they match with a pair of global nodes.

The interpolation functions “$N_i^{(e)}$” are defined over the element “$\Omega^{(e)}$”, and they are equal to zero outside the element. Hence, they are said to have compact support. The interpolation in which only the functions alone is interpolated – and not its derivative – is known as the Lagrange interpolation, and the corresponding functions are termed the Lagrange interpolation functions.

In the appendices attached to this appendix, the numerical operation of the FEM is developed in a generic way.

A-C.3.1. Advantages

- Possibility of using unstructured meshes: domains with irregular contours (adaptability)
- The boundary conditions are imposed as follows systematically (without casuistry)
- It allows general routines: systematic calculation of everything, describing properly the data of the problem (geometry, boundary conditions, initial conditions, etc.) a single FEM code allows to solve several boundary problems.

Annex D. Kratos

A-D.1. GiD

GiD has been developed by the CIMNE. It's a **universal** (ideal for generating all the information required for the analysis of any problem in science and engineering using numerical methods), **adaptive** (extremely easy to adapt to any numerical simulation code, input and output formats can be customized and made compatible with any existing software) and **user-friendly** (development focused on the user´s needs, provides with simplicity, speed and effectiveness for users) pre and post processor for numerical simulations in science and engineering. It has been designed to cover all the common needs in the numerical simulations field from pre- to post-processing:

- Geometrical modeling (CAD)
- Mesh generation
- Definition of analysis data
- Data transfer to analysis software
- Postprocessing operations
- Visualization of results

Figure A-D.1: Source: [27]

A-D.2. Kratos

Kratos is a free multi-physic Finite Element C++ open-source code. One of the main topics in engineering nowadays is the combination of different analysis (thermal, fluid dynamic, structural) with optimizing methods in one global software package with just one user interface and, even more, the possibility to extend the implemented solution to new problems.

Kratos is designed as an Open-Source framework for the implementation of numerical methods for the solution of engineering problems. It is written in C++ and is designed to allow collaborative development by large teams of researchers focusing on modularity as well as on performance. The Kratos features a "core" and "applications" approach where "standard tools" (databases, linear algebra, search structures, etc...) come as a part of the core and are available as building blocks in the development of "applications" which focus on the solution of the problems of interest. Its ultimate goal is to simplify the development of new numerical methods.

Figure A-D.2: Source: [28]

A-D.2.1. First challenges on the IT side

The first challenge that had to be faced was the fact that the GiD program is not very advanced in the Macintosh operating system in terms of the Kratos problem type. The process of downloading, installing and licensing the program is described in the following table:

Summary of steps to follow:
1) Install **XCode** (integrated development environment, *IDE*, for macOS that contains a set of tools created by Apple for software development for *macOS*, *iOS*, *watchOS* and *tvOS).* 2) Install Clang with OMP support. 3) Install Python 3 (an interpreted high-level general-purpose programming language) 4) Install Boost (it provides free peer-reviewed portable C++ source libraries). 5) Install CMake (open-source, cross-platform family of tools designed to build, test and package software). 6) Download Kratos. 7) Compile Kratos. 8) Set up the shell environment. 9) Test Kratos.

A-D.2.2. File formats

Once the geometry is created, characterized with its mechanical properties, the boundary and initial conditions are imposed, the type of calculation is programmed with its properties and the mesh is generated, the following files are generated when the program is saved.

Extension files:

$.\boldsymbol{cnd}\ \ file\ = extension\ commonly\ associated\ with\ Melco\ Condensed\ Embroidery\ Format\ files$

$.\boldsymbol{geo}\ \ file\ = 3D\ CAD\ file\ saved\ in\ GEO\ format\ (include\ contour\ information, bend\ data, material\ and\ thickness)$

$.\boldsymbol{msh}\ \ file\ = 3D\ model\ created\ by\ Godot\ Engine$

$.\boldsymbol{png}\ \ file\ = image\ saved\ in\ the\ Portable\ Network\ Graphic\ (PNG)\ format$

$.\boldsymbol{prj}\ \ file\ = data\ files\ which\ are\ used\ by\ multiple\ programs\ to\ save\ project\ data\ and\ settings$

$.\boldsymbol{spd}\ \ file\ = contain\ settings\ and\ instructions\ that\ are\ used\ by\ Adobe\ PostScript\ printers$

$.\boldsymbol{tree}\ \ file\ = file\ extension\ indicates\ to\ your\ device\ which\ app\ can\ open\ the\ file$

$.\boldsymbol{vv}\ \ file\ = virt-viewer\ configuration\ file$

Once the calculation button is pressed within the pre-processing of the program interface, the following files appear:

Extension files:

$.\boldsymbol{err}\ file\ =text\ file\ associated\ with\ Error\ Log\ Format\ files\ that\ contains\ error\ messages\ generated\ by\ a\ program$

$.\boldsymbol{info}\ file\ =generic\ text\ information\ file\ that\ may\ be\ included\ with\ various\ software\ programs\ or\ file\ downloads$

$.\boldsymbol{mdpa}\ file\ =Model\ Part\ (Kratos\ current\ input\ data\ archive)$

$.\boldsymbol{bas}\ file\ =Programm\ written\ in\ Basic\ language$

$.\boldsymbol{cpp}\ file\ =Programm\ written\ in\ C++\ language$

$.\boldsymbol{h}\ file\ =header\ file\ referenced\ by\ a\ document\ written\ in\ C, C++, or\ Objective-C\ source\ code$

$.\boldsymbol{py}\ file\ =Programm\ written\ in\ Python\ language$

$\boldsymbol{MainKratos.py}\ file$

$.\boldsymbol{json}\ file\ =\ file\ that\ stores\ simple\ data\ structures\ and\ objects\ in\ Java\ Script\ Object\ Notation\ (JSON)\ format$

$\boldsymbol{PFEMFluidMaterials.json}\ file$

$\boldsymbol{ProjectParameters.json}\ file$

Once all the calculations necessary to solve the model have been performed, the following files are created:

Extension files:

$.\boldsymbol{bin}\ file\ =compressed\ binary\ files\ that\ are\ used\ for\ varied\ purposes\ by\ many\ computer\ applications$

$.\boldsymbol{post}\ file\ =Data\ file\ created\ by\ LIGHT, that\ enables\ users\ to\ record, organize, and\ share\ their\ life\ experiences$

$.\boldsymbol{post.bin}\ file\ =\ .\boldsymbol{post}\ \cup\ .\boldsymbol{bin}$

$.\boldsymbol{lst}\ file\ =Text\ file\ that\ contains\ a\ list\ of\ data$

$.\boldsymbol{iso}\ file\ =File\ that\ contains\ an\ identical\ copy\ of\ data\ found\ on\ an\ optical\ disc\ (CD\ or\ DVD)$

$.\boldsymbol{txt}\ file\ =standard\ text\ document\ that\ contains\ plain\ text$

$\boldsymbol{PtotalVolumeBeforeMeshing.txt}\ file$

Annex E. Codes

A-E.1. herschel_bulkley_2D_law.cpp

```
//-----------------------------------------------------------
//          ___  __            ___ _      _    _
//  KRATOS| _ \/ _|___ _ __  | __| |_  _(_)__| |
//        |  _/  _/ -_) '  \| _|| | || | / _` |
//        |_| |_| \___|_|_|_|_| |_|\_,_|_\__,_|DYNAMICS
//
//  BSD License:    PfemFluidDynamicsApplication/license.txt
//
//  Collaborator:  Timur Tomas
//
//-----------------------------------------------------------
//

// System includes
#include <iostream>

// External includes
#include <cmath>

// Project includes
#include "custom_constitutive/fluid_laws/herschel_bulkley_2D_law.h"
#include "includes/checks.h"
#include "includes/properties.h"
#include "pfem_fluid_dynamics_application_variables.h"

namespace Kratos
{

//**********************************CONSTRUCTOR*************************************

//**********************************************************************************

    HerschelBulkley2DLaw::HerschelBulkley2DLaw() : PfemFluidConstitutiveLaw()
{}

//**********************************COPY CONSTRUCTOR********************************

//**********************************************************************************

    HerschelBulkley2DLaw::HerschelBulkley2DLaw(const HerschelBulkley2DLaw
&rOther) : PfemFluidConstitutiveLaw(rOther) {}


//**********************************CLONE*******************************************

//**********************************************************************************

    ConstitutiveLaw::Pointer HerschelBulkley2DLaw::Clone() const { return
Kratos::make_shared<HerschelBulkley2DLaw>(*this); }
```

```
//*********************************DESTRUCTOR*********************************

//****************************************************************************

    HerschelBulkley2DLaw::~HerschelBulkley2DLaw() {}

    ConstitutiveLaw::SizeType HerschelBulkley2DLaw::WorkingSpaceDimension() {
return 2; }

    ConstitutiveLaw::SizeType HerschelBulkley2DLaw::GetStrainSize() { return
3; }

    void HerschelBulkley2DLaw::CalculateMaterialResponseCauchy(Parameters
&rValues)
    {
        Flags &r_options = rValues.GetOptions();

        const Properties &r_properties = rValues.GetMaterialProperties();

        Vector &r_strain_vector = rValues.GetStrainVector();
        Vector &r_stress_vector = rValues.GetStressVector();

        const double dynamic_viscosity = this-
>GetEffectiveDynamicViscosity(rValues);
        const double yield_shear = this->GetEffectiveYieldShear(rValues);
        const double adaptive_exponent = r_properties[ADAPTIVE_EXPONENT];
        double effective_dynamic_viscosity;
        const double flow_index = this->GetFlowIndex(rValues);

        const double equivalent_strain_rate =
            std::sqrt(2.0 * r_strain_vector[0] * r_strain_vector[0] + 2.0 *
r_strain_vector[1] * r_strain_vector[1] +
                      4.0 * r_strain_vector[2] * r_strain_vector[2]);

        // Ensuring that the case of equivalent_strain_rate = 0 is not
problematic.
        const double tolerance = 1e-8;
        if (equivalent_strain_rate < tolerance)
        {
            effective_dynamic_viscosity = yield_shear * adaptive_exponent;
        }
        else
        {
            //Se tomará el índice de consistencia como la dynamic_vicosity
            double regularization = 1.0 - std::exp(-adaptive_exponent *
equivalent_strain_rate);
            double aux_flow_index = flow_index - 1;
            effective_dynamic_viscosity = dynamic_viscosity *
std::pow(equivalent_strain_rate,aux_flow_index) + regularization * yield_shear
/ equivalent_strain_rate;
```

```
        }

        const double strain_trace = r_strain_vector[0] + r_strain_vector[1];

        //This stress_vector is only deviatoric
        // d' = d - I * tr(d)/3
        r_stress_vector[0] = 2.0 * effective_dynamic_viscosity *
(r_strain_vector[0] - strain_trace / 3.0);
        r_stress_vector[1] = 2.0 * effective_dynamic_viscosity *
(r_strain_vector[1] - strain_trace / 3.0);
        r_stress_vector[2] = 2.0 * effective_dynamic_viscosity *
r_strain_vector[2];

        if (r_options.Is(ConstitutiveLaw::COMPUTE_CONSTITUTIVE_TENSOR))
        {
            this-
>EffectiveViscousConstitutiveMatrix2D(effective_dynamic_viscosity,
rValues.GetConstitutiveMatrix());
        }
    }

    std::string HerschelBulkley2DLaw::Info() const { return
"HerschelBulkley2DLaw"; }

//******************CHECK CONSISTENCY IN THE CONSTITUTIVE LAW*****************

//*****************************************************************************

    int HerschelBulkley2DLaw::Check(const Properties &rMaterialProperties,
const GeometryType &rElementGeometry,
                            const ProcessInfo &rCurrentProcessInfo)
    {

        KRATOS_CHECK_VARIABLE_KEY(DYNAMIC_VISCOSITY);
        KRATOS_CHECK_VARIABLE_KEY(YIELD_SHEAR);
        KRATOS_CHECK_VARIABLE_KEY(ADAPTIVE_EXPONENT);
        KRATOS_CHECK_VARIABLE_KEY(BULK_MODULUS);
        KRATOS_CHECK_VARIABLE_KEY(FLOW_INDEX);

        if (rMaterialProperties[DYNAMIC_VISCOSITY] < 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing DYNAMIC_VISCOSITY provided
in process info for HerschelBulkley2DLaw: "
                         << rMaterialProperties[DYNAMIC_VISCOSITY] <<
std::endl;
        }

        if (rMaterialProperties[YIELD_SHEAR] < 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing YIELD_SHEAR provided in
process info for HerschelBulkley2DLaw: "
```

```
                        << rMaterialProperties[YIELD_SHEAR] << std::endl;
        }

        if (rMaterialProperties[ADAPTIVE_EXPONENT] < 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing ADAPTIVE_EXPONENT provided
in process info for HerschelBulkley2DLaw: "
                        << rMaterialProperties[ADAPTIVE_EXPONENT] <<
std::endl;
        }

        if (rMaterialProperties[BULK_MODULUS] <= 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing BULK_MODULUS provided in
process info for HerschelBulkley2DLaw: "
                        << rMaterialProperties[BULK_MODULUS] << std::endl;
        }

        if (rMaterialProperties[FLOW_INDEX] <= 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing FLOW_INDEX provided in
process info for HerschelBulkley2DLaw: "
                        << rMaterialProperties[FLOW_INDEX] << std::endl;
        }

        return 0;
    }

    double
HerschelBulkley2DLaw::GetEffectiveViscosity(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetConstitutiveMatrix()(2, 2);
    }

    double
HerschelBulkley2DLaw::GetEffectiveDensity(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetMaterialProperties()[DENSITY];
    }

    double
HerschelBulkley2DLaw::GetEffectiveDynamicViscosity(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetMaterialProperties()[DYNAMIC_VISCOSITY];
    }
```

```
    double
HerschelBulkley2DLaw::GetEffectiveYieldShear(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetMaterialProperties()[YIELD_SHEAR];
    }

    double HerschelBulkley2DLaw::GetFlowIndex(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetMaterialProperties()[FLOW_INDEX];
    }

    void HerschelBulkley2DLaw::save(Serializer &rSerializer) const
    {
        KRATOS_SERIALIZE_SAVE_BASE_CLASS(rSerializer,
PfemFluidConstitutiveLaw)
    }

    void HerschelBulkley2DLaw::load(Serializer &rSerializer)
    {
        KRATOS_SERIALIZE_LOAD_BASE_CLASS(rSerializer,
PfemFluidConstitutiveLaw)
    }

} // Namespace Kratos
```

A-E.2. herschel_bulkley_2D_law.h

```
//-----------------------------------------------------------------
//          ___  __            ___ _      _    _
//  KRATOS| _ \/ _|___ _ __  | __| |_  _(_)__| |
//        |  _/  _/ -_) '  \| _|| | || | / _` |
//        |_| |_| \___|_|_|_|_| |_|\_,_|_\__,_|DYNAMICS
//
//  BSD License:    PfemFluidDynamicsApplication/license.txt
//
//  Collaborator:  Timur Tomas
//
//-----------------------------------------------------------------
//

#if !defined(KRATOS_HERSCHEL_BULKLEY_LAW_2D_H_INCLUDED)
#define KRATOS_HERSCHEL_BULKLEY_LAW_2D_H_INCLUDED

// System includes

// External includes

// Project includes
#include "fluid_constitutive_law.h"

namespace Kratos {
/**
 * Defines a 2D Herschel Bulkley non-Newtonian constitutive law
 * This material law is defined by the parameters:
 * 1) DYNAMIC_VISCOSITY
 * 2) YIELD_SHEAR
 * 3) ADAPTIVE_EXPONENT
 * 4) FLOW_INDEX
 */
class KRATOS_API(PFEM_FLUID_DYNAMICS_APPLICATION) HerschelBulkley2DLaw :
public PfemFluidConstitutiveLaw {
   public:
    /**
     * Type Definitions
     */
    typedef ProcessInfo ProcessInfoType;
    typedef ConstitutiveLaw BaseType;
    typedef std::size_t SizeType;

    /**
     * Counted pointer of HerschelBulkley2DLaw
     */
    KRATOS_CLASS_POINTER_DEFINITION(HerschelBulkley2DLaw);

    /**
     * Life Cycle
```

```
     */

    /**
     * Default constructor.
     */
    HerschelBulkley2DLaw();

    /**
     * Clone function (has to be implemented by any derived class)
     * @return a pointer to a new instance of this constitutive law
     */
    ConstitutiveLaw::Pointer Clone() const override;

    /**
     * Copy constructor.
     */
    HerschelBulkley2DLaw(const HerschelBulkley2DLaw& rOther);

    /**
     * Destructor.
     */
    ~HerschelBulkley2DLaw() override;

    /**
     * Operators
     */

    /**
     * Operations needed by the base class:
     */

    /**
     * @return Working space dimension constitutive law
     */
    SizeType WorkingSpaceDimension() override;

    /**
     * @return Size of the strain vector (in Voigt notation) for the
constitutive law
     */
    SizeType GetStrainSize() override;

    void CalculateMaterialResponseCauchy(Parameters& rValues) override;

    /**
     * This function is designed to be called once to perform all the checks
needed
     * on the input provided. Checks can be "expensive" as the function is
designed
     * to catch user's errors.
     * @param rMaterialProperties
```

```
     * @param rElementGeometry
     * @param rCurrentProcessInfo
     * @return
     */
    int Check(const Properties& rMaterialProperties, const GeometryType&
rElementGeometry,
              const ProcessInfo& rCurrentProcessInfo) override;

    /**
     * Input and output
     */

    /**
     * Turn back information as a string.
     */
    std::string Info() const override;

   protected:
    ///@name Protected static Member Variables
    ///@{
    ///@}
    ///@name Protected member Variables
    ///@{
    ///@}
    ///@name Protected Operators
    ///@{
    ///@}
    ///@name Protected Operations
    ///@{

    /// Get the effective viscosity (in dynamic units -- Pa s) for the fluid.
    double GetEffectiveViscosity(ConstitutiveLaw::Parameters& rParameters)
const override;

    /// Get the effective density for the fluid.
    double GetEffectiveDensity(ConstitutiveLaw::Parameters& rParameters) const
override;

    /// Get the effective yield shear for the fluid.
    double GetEffectiveYieldShear(ConstitutiveLaw::Parameters& rParameters)
const;

    /// Get the effective dynamic viscosity for the fluid.
    double GetEffectiveDynamicViscosity(ConstitutiveLaw::Parameters&
rParameters) const;

    /// Get the flow index for the fluid.
    double GetFlowIndex(ConstitutiveLaw::Parameters& rParameters) const;

    ///@}
```

```
    private:
     ///@name Static Member Variables
     ///@{

     ///@}
     ///@name Member Variables
     ///@{

     ///@}
     ///@name Private Operators
     ///@{

     ///@}
     ///@name Private Operations
     ///@{
     ///@}

     ///@}
     ///@name Private  Access
     ///@{
     ///@}

     ///@}
     ///@name Serialization
     ///@{
     friend class Serializer;

     void save(Serializer& rSerializer) const override;

     void load(Serializer& rSerializer) override;
     ///@}

};  // Class HerschelBulkley2DLaw

}  // namespace Kratos.

#endif  // KRATOS_HERSCHEL_BULKLEY_LAW_2D_H_INCLUDED  defined
```

A-E.3. herschel_bulkley_3D_law.cpp

```
//-----------------------------------------------------------------
//          ___  __            ___ _      _    _
//  KRATOS| _ \/ _|___ _ __  | __| |_  _(_)__| |
//        |  _/  _/ -_) '  \| _|| | || | / _` |
//        |_| |_| \___|_|_|_|_| |_|\_,_|_\__,_|DYNAMICS
//
//  BSD License:    PfemFluidDynamicsApplication/license.txt
//
//  Collaborator:  Timur Tomas
//
//-----------------------------------------------------------------
//

// System includes
#include <iostream>

// External includes
#include <cmath>

// Project includes
#include "custom_constitutive/fluid_laws/herschel_bulkley_3D_law.h"
#include "includes/checks.h"
#include "includes/properties.h"
#include "pfem_fluid_dynamics_application_variables.h"

namespace Kratos
{


//*********************************CONSTRUCTOR*********************************

//*****************************************************************************

    HerschelBulkley3DLaw::HerschelBulkley3DLaw() : PfemFluidConstitutiveLaw()
{}

    //*******************************COPY CONSTRUCTOR*************************
    //************************************************************************

    HerschelBulkley3DLaw::HerschelBulkley3DLaw(const HerschelBulkley3DLaw
&rOther) : PfemFluidConstitutiveLaw(rOther) {}


//*********************************CLONE*********************************

//*****************************************************************************

    ConstitutiveLaw::Pointer HerschelBulkley3DLaw::Clone() const { return
Kratos::make_shared<HerschelBulkley3DLaw>(*this); }
```

```
//*********************************DESTRUCTOR*********************************

//****************************************************************************

    HerschelBulkley3DLaw::~HerschelBulkley3DLaw() {}

    ConstitutiveLaw::SizeType HerschelBulkley3DLaw::WorkingSpaceDimension() {
return 3; }

    ConstitutiveLaw::SizeType HerschelBulkley3DLaw::GetStrainSize() { return
6; }

    void HerschelBulkley3DLaw::CalculateMaterialResponseCauchy(Parameters
&rValues)
    {

        Flags &r_options = rValues.GetOptions();

        const Properties &r_properties = rValues.GetMaterialProperties();

        Vector &r_strain_vector = rValues.GetStrainVector();
        Vector &r_stress_vector = rValues.GetStressVector();

        const double dynamic_viscosity = this-
>GetEffectiveDynamicViscosity(rValues);
        const double yield_shear = this->GetEffectiveYieldShear(rValues);
        const double adaptive_exponent = r_properties[ADAPTIVE_EXPONENT];
        double effective_dynamic_viscosity;
        const double flow_index = this->GetFlowIndex(rValues);

        const double equivalent_strain_rate =
            std::sqrt(2.0 * r_strain_vector[0] * r_strain_vector[0] + 2.0 *
r_strain_vector[1] * r_strain_vector[1] +
                      2.0 * r_strain_vector[2] * r_strain_vector[2] + 4.0 *
r_strain_vector[3] * r_strain_vector[3] +
                      4.0 * r_strain_vector[4] * r_strain_vector[4] + 4.0 *
r_strain_vector[5] * r_strain_vector[5]);

        // Ensuring that the case of equivalent_strain_rate = 0 is not
problematic
        const double tolerance = 1e-8;
        if (equivalent_strain_rate < tolerance)
        {
            effective_dynamic_viscosity = yield_shear * adaptive_exponent;
        }
        else
        {
            double regularization = 1.0 - std::exp(-adaptive_exponent *
equivalent_strain_rate);
```

```
            effective_dynamic_viscosity = dynamic_viscosity *
pow(equivalent_strain_rate,flow_index - 1) + regularization * yield_shear /
equivalent_strain_rate;
        }

        const double strain_trace = r_strain_vector[0] + r_strain_vector[1];



        const double strain_trace = r_strain_vector[0] + r_strain_vector[1] +
r_strain_vector[2];

        r_stress_vector[0] = 2.0 * effective_dynamic_viscosity *
(r_strain_vector[0] - strain_trace / 3.0);
        r_stress_vector[1] = 2.0 * effective_dynamic_viscosity *
(r_strain_vector[1] - strain_trace / 3.0);
        r_stress_vector[2] = 2.0 * effective_dynamic_viscosity *
(r_strain_vector[2] - strain_trace / 3.0);
        r_stress_vector[3] = 2.0 * effective_dynamic_viscosity *
r_strain_vector[3];
        r_stress_vector[4] = 2.0 * effective_dynamic_viscosity *
r_strain_vector[4];
        r_stress_vector[5] = 2.0 * effective_dynamic_viscosity *
r_strain_vector[5];

        if (r_options.Is(ConstitutiveLaw::COMPUTE_CONSTITUTIVE_TENSOR))
        {
            this-
>EffectiveViscousConstitutiveMatrix3D(effective_dynamic_viscosity,
rValues.GetConstitutiveMatrix());
        }
    }

    std::string HerschelBulkley3DLaw::Info() const { return
"HerschelBulkley3DLaw"; }

    //******************CHECK CONSISTENCY IN THE CONSTITUTIVE LAW*************

//*************************************************************************************

    int HerschelBulkley3DLaw::Check(const Properties &rMaterialProperties,
const GeometryType &rElementGeometry,
                            const ProcessInfo &rCurrentProcessInfo)
    {

        KRATOS_CHECK_VARIABLE_KEY(DYNAMIC_VISCOSITY);
        KRATOS_CHECK_VARIABLE_KEY(YIELD_SHEAR);
        KRATOS_CHECK_VARIABLE_KEY(ADAPTIVE_EXPONENT);
        KRATOS_CHECK_VARIABLE_KEY(BULK_MODULUS);
        KRATOS_CHECK_VARIABLE_KEY(FLOW_INDEX);
```

```
        if (rMaterialProperties[DYNAMIC_VISCOSITY] < 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing DYNAMIC_VISCOSITY provided
in process info for HerschelBulkley3DLaw: "
                         << rMaterialProperties[DYNAMIC_VISCOSITY] <<
std::endl;
        }

        if (rMaterialProperties[YIELD_SHEAR] < 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing YIELD_SHEAR provided in
process info for HerschelBulkley3DLaw: "
                         << rMaterialProperties[YIELD_SHEAR] << std::endl;
        }

        if (rMaterialProperties[ADAPTIVE_EXPONENT] < 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing ADAPTIVE_EXPONENT provided
in process info for HerschelBulkley3DLaw: "
                         << rMaterialProperties[ADAPTIVE_EXPONENT] <<
std::endl;
        }

        if (rMaterialProperties[BULK_MODULUS] <= 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing BULK_MODULUS provided in
process info for HerschelBulkley3DLaw: "
                         << rMaterialProperties[BULK_MODULUS] << std::endl;
        }

        if (rMaterialProperties[FLOW_INDEX] <= 0.0)
        {
            KRATOS_ERROR << "Incorrect or missing FLOW_INDEX provided in
process info for HerschelBulkley3DLaw: "
                         << rMaterialProperties[FLOW_INDEX] << std::endl;
        }

        return 0;
    }

    double
HerschelBulkley3DLaw::GetEffectiveViscosity(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetConstitutiveMatrix()(5, 5);
    }

    double
HerschelBulkley3DLaw::GetEffectiveDensity(ConstitutiveLaw::Parameters
&rParameters) const
    {
```

```
        return rParameters.GetMaterialProperties()[DENSITY];
    }

    double
HerschelBulkley3DLaw::GetEffectiveDynamicViscosity(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetMaterialProperties()[DYNAMIC_VISCOSITY];
    }

    double
HerschelBulkley3DLaw::GetEffectiveYieldShear(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetMaterialProperties()[YIELD_SHEAR];
    }

    double HerschelBulkley3DLaw::GetFlowIndex(ConstitutiveLaw::Parameters
&rParameters) const
    {
        return rParameters.GetMaterialProperties()[FLOW_INDEX];
    }

    void HerschelBulkley3DLaw::save(Serializer &rSerializer) const
    {
        KRATOS_SERIALIZE_SAVE_BASE_CLASS(rSerializer,
PfemFluidConstitutiveLaw)
    }

    void HerschelBulkley3DLaw::load(Serializer &rSerializer)
    {
        KRATOS_SERIALIZE_LOAD_BASE_CLASS(rSerializer,
PfemFluidConstitutiveLaw)
    }

} // Namespace Kratos
```

A-E.4. herschel_bulkley_3D_law.h

```
//-----------------------------------------------------------------
//          ___  __            ___ _      _    _
//  KRATOS| _ \/ _|___ _ __  | __| |_  _(_)__| |
//        |  _/  _/ -_) '  \| _|| | || | / _` |
//        |_| |_| \___|_|_|_|_| |_|\_,_|_\__,_|DYNAMICS
//
//  BSD License:    PfemFluidDynamicsApplication/license.txt
//
//  Collaborator:  Timur Tomas
//
//-----------------------------------------------------------------
//

#if !defined(KRATOS_HERSCHEL_BULKLEY_LAW_3D_H_INCLUDED)
#define KRATOS_HERSCHEL_BULKLEY_LAW_3D_H_INCLUDED

// System includes

// External includes

// Project includes
#include "fluid_constitutive_law.h"

namespace Kratos {
/**
 * Defines a 3D Herschel Bulkley non-Newtonian constitutive law
 * This material law is defined by the parameters:
 * 1) DYNAMIC_VISCOSITY
 * 2) YIELD_SHEAR
 * 3) ADAPTIVE_EXPONENT
 * 4) FLOW_INDEX
 */
class KRATOS_API(PFEM_FLUID_DYNAMICS_APPLICATION) HerschelBulkley3DLaw :
public PfemFluidConstitutiveLaw {
   public:
    /**
     * Type Definitions
     */
    typedef ProcessInfo ProcessInfoType;
    typedef ConstitutiveLaw BaseType;
    typedef std::size_t SizeType;

    /**
     * Counted pointer of HerschelBulkley3DLaw
     */
    KRATOS_CLASS_POINTER_DEFINITION(HerschelBulkley3DLaw);

    /**
     * Life Cycle
```

```
     */

    /**
     * Default constructor.
     */
    HerschelBulkley3DLaw();

    /**
     * Clone function (has to be implemented by any derived class)
     * @return a pointer to a new instance of this constitutive law
     */
    ConstitutiveLaw::Pointer Clone() const override;

    /**
     * Copy constructor.
     */
    HerschelBulkley3DLaw(const HerschelBulkley3DLaw& rOther);

    /**
     * Destructor.
     */
    ~HerschelBulkley3DLaw() override;

    /**
     * Operators
     */

    /**
     * Operations needed by the base class:
     */

    /**
     * @return Working space dimension constitutive law
     */
    SizeType WorkingSpaceDimension() override;

    /**
     * @return Size of the strain vector (in Voigt notation) for the
constitutive law
     */
    SizeType GetStrainSize() override;

    void CalculateMaterialResponseCauchy(Parameters& rValues) override;

    /**
     * This function is designed to be called once to perform all the checks
needed
     * on the input provided. Checks can be "expensive" as the function is
designed
     * to catch user's errors.
     * @param rMaterialProperties
```

```
     * @param rElementGeometry
     * @param rCurrentProcessInfo
     * @return
     */
    int Check(const Properties& rMaterialProperties, const GeometryType&
rElementGeometry,
              const ProcessInfo& rCurrentProcessInfo) override;

    /**
     * Input and output
     */

    /**
     * Turn back information as a string.
     */
    std::string Info() const override;

   protected:
    ///@name Protected static Member Variables
    ///@{
    ///@}
    ///@name Protected member Variables
    ///@{
    ///@}
    ///@name Protected Operators
    ///@{
    ///@}
    ///@name Protected Operations
    ///@{

    /// Get the effective viscosity (in dynamic units -- Pa s) for the fluid.
    double GetEffectiveViscosity(ConstitutiveLaw::Parameters& rParameters)
const override;

    /// Get the effective density for the fluid.
    double GetEffectiveDensity(ConstitutiveLaw::Parameters& rParameters) const
override;

    /// Get the effective yield shear for the fluid.
    double GetEffectiveYieldShear(ConstitutiveLaw::Parameters& rParameters)
const;

    /// Get the effective dynamic viscosity for the fluid.
    double GetEffectiveDynamicViscosity(ConstitutiveLaw::Parameters&
rParameters) const;

    /// Get the flow index for the fluid.
    double GetFlowIndex(ConstitutiveLaw::Parameters& rParameters) const;

    ///@}
```

```
    private:
     ///@name Static Member Variables
     ///@{

     ///@}
     ///@name Member Variables
     ///@{

     ///@}
     ///@name Private Operators
     ///@{

     ///@}
     ///@name Private Operations
     ///@{
     ///@}

     ///@}
     ///@name Private  Access
     ///@{
     ///@}

     ///@}
     ///@name Serialization
     ///@{
     friend class Serializer;

     void save(Serializer& rSerializer) const override;

     void load(Serializer& rSerializer) override;
     ///@}

};  // Class HerschelBulkley3DLaw

}  // namespace Kratos.

#endif  // KRATOS_HERSCHEL_BULKLEY_LAW_3D_H_INCLUDED  defined
```