19th Brazilian Congress of Thermal Sciences and Engineering
November 6th–10th, 2022, Bento Gonçalves - RS - Brazil

ENC-2022-0091
Numerical Simulation of Projectile Impact on Newtonian Fluid Pool

Vitor Pinheiro Pinto
Rachel Lucena
Norberto Mangiavacchi
Rio de Janeiro State University, Rio de Janeiro, RJ, Brazil
vitor.pinto@graduacao.uerj.br,rachel.lucena@uerj.br, norberto.mangiavacchi@eng.uerj.br

Rogério Arving Serra
Polícia Técnico-Científica do Estado do Rio de Janeiro, RJ, Brazil
arving4@gmail.com

Abstract. *Ballistic chambers are employed by police forensic investigation teams to solve crimes involving fire guns. The goal of this computational study is to analyze the impact dynamics of a high-speed projectile on a steady pool of Newtonian fluid, and produce data to be employed in the design of ballistic chambers. The projectile is modeled as a moving rigid body and the fluid flow is modeled by the unsteady incompressible Navier-Stokes and continuity equations. The cavity interface is treated as a deformable free-surface (zero-stress) boundary, and the fluid-projectile interface is a moving contact point problem. The governing equations are solved using an axis-symmetric (2D) Arbitrary Lagrangian-Eulerian Finite Element Method (ALE-FEM), the Galerkin's formulation and semi-Lagragian approach. The fluid domain is discretized using a triangular element unstructured mesh using quadratic base functions and the projectile is modeled as a 3-degrees of freedom rigid body. The unsteady solution is updated in time using an uncoupled semi-implicit integration method, using primitive variables. The material derivative is discretized using a semi-Lagrangian approach, such that the resulting scheme is unconditionally stable. At each time iteration, the mesh nodes positions are updated with a mesh (ALE) velocity computed using a differential approach using the velocity of the interface as boundary conditions. Additionally, the mesh velocity employs a smoothing strategy to reduce the elements deformations. Selected regions are remeshed in order to remove very distorted elements. The projectile dynamics is computed using the fluid stresses on the wet contact. The computational method is implemented using the Julia programming language, which provides high performance in multi-core computer architectures, using multi-threading and the data parallel paradigm, and the language has proven to be more efficient when compare to other scientific programming languages. The simulations provide estimates of the stresses that the ballistic chamber will undergo, and the dimensions needed for the complete deceleration of specific projectiles and fire guns used, thus providing data for the chamber project. The solver has the potential to be used as a foundation to other applications in the forensic ballistic field that can support police investigators and reduce the gunfire crimes investigations time.*
Keywords: *ballistic chamber, cavitation, free-surface, ALE-FEM*

1. INTRODUCTION

The description of bodies impacts on Newtonian Fluid have been heavily studied since World War II, when the mechanics of torpedo movement in sea waters and several other important applications, came to the spotlight, [2]. In this context, the ballistic chambers are employed by police forensic investigation teams to solve crimes involving fire guns.

This work aims to help Brazil's police solving crimes involving fire guns faster with tools capable of computationally support ballistic forensics investigators, a less terrifying motivation at least.

In Brazil most of the culpable homicides hadn't been solved yet, as [7] has shown, in average only 20% of homicide investigations come to an end in the same year of their occurrence, and in the subsequent years only 10% more of crimes' perpetrators will see themselves inside cages for the next 40 years of their lives. In the light of these statistics, one may become aware of Brasil's police needs of tools capable of increasing these low numbers.

This work proposal is to present simple axis-symmetric 2D meshed FEM-ALE solvers developed in Julia, [1], that reproduces projectiles movement inside ballistics chambers. A case study is developed and its simulated parameters are contrasted with real data, available by others labs and obtained in UERJ's lab, to bring the sense of reliability which is an important attribute for this kind of software.

2. METHODOLOGY

2.1 MODEL

This work describes projects of cylindrical Ballistic Chambers as illustrated by the figure 1. The problem geometry and boundary conditions allow to emply an axis-symmetric strategy which is illustrated by the mesh in the figure below, representing only the right hand region of the section highlighted in the image.

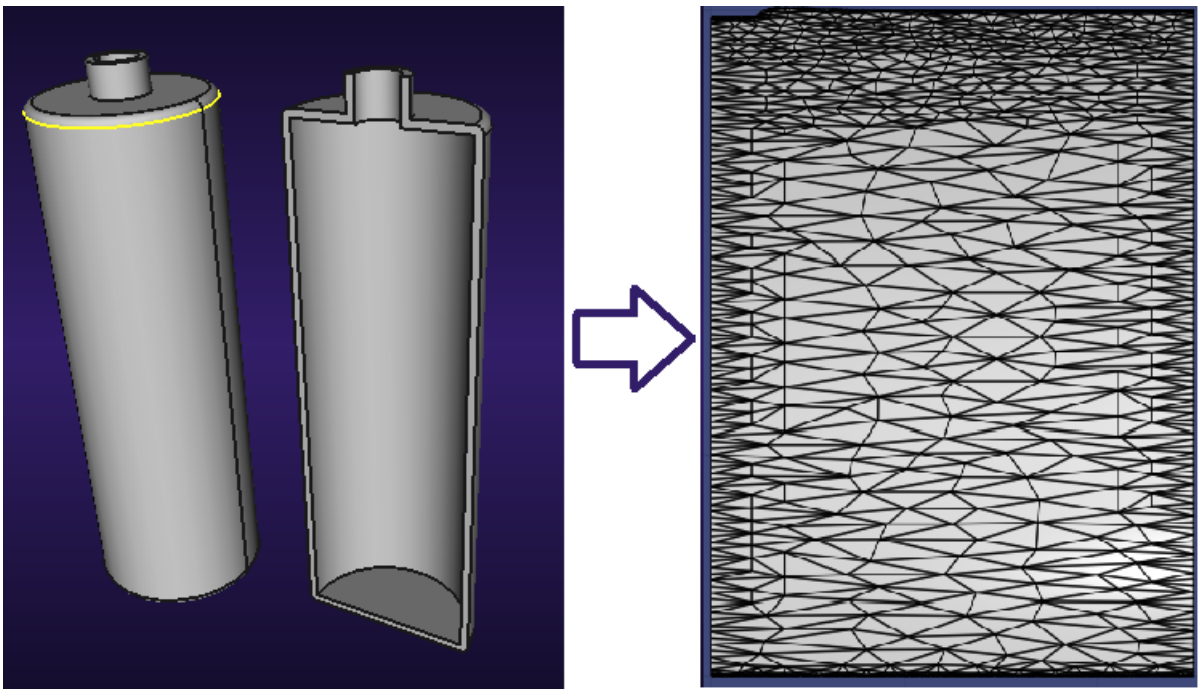

Figure 1. Ballistic chamber model (left) and initial mesh (right)

The mesh boundary regions are key to this work. The figure 2 shows those regions and the nomenclature employed in this work. Neumann and Dirichlet boundary conditions apply to various parts of the boundary and are detailed later in this work.

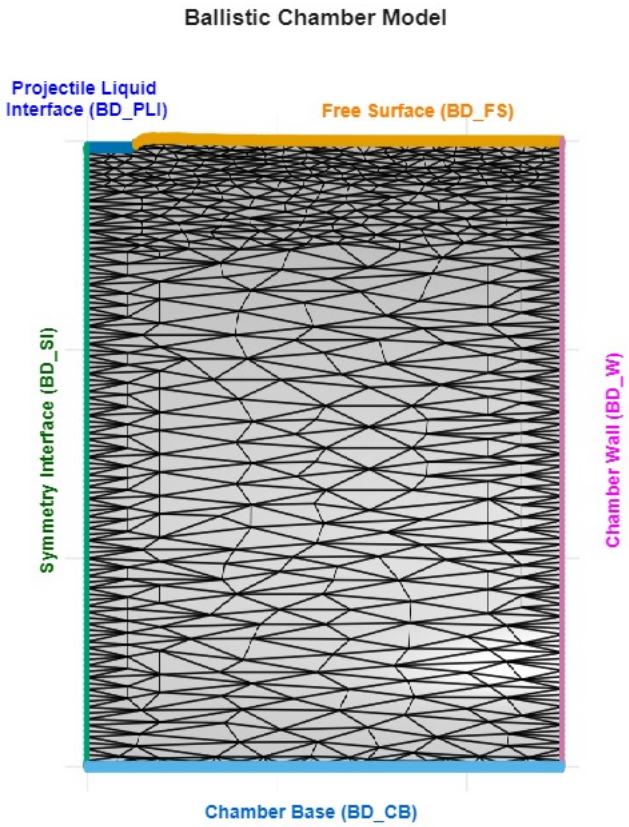

Figure 2. Fluid Domain Mesh Boundary Regions

The system physics is described by the mass and momentum conservation equations for an incompressible flow, and the system is entirely defined by the pressure field (P) and velocity field ($\boldsymbol{U} = u_i$), given the gravitational field ($\boldsymbol{g} = g_i$), in a Cartesian coordinate system which origin is located at intersection between the ballistic chamber bottom and its symmetry axis, as can be observed in Fig. 2.

The conservation equations, using index notation, are listed below, as shown in details in [9].

Conservation of Mass
$$\frac{\partial u_i}{\partial x_i} = 0 \tag{1a}$$

Conservation of Momentum
$$\rho \left(\frac{\partial u_j}{\partial t} + u_i \frac{\partial u_j}{\partial x_i}\right) = -\frac{\partial P}{x_j} + \frac{\partial \tau_{ij}}{\partial x_i} + \rho\, g_j \tag{1b}$$

with the *deviatoric tensor* defined as:

$$\boldsymbol{\tau} = \tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2\delta_{ij}}{3} \frac{\partial u_k}{\partial x_k}\right) \tag{2}$$

and δ_{ij} operation is the Kronecker delta [6].

The simulation starts at the first contact between the axis-symmetric projectile and the fluid surface. The region of contact is refered as BD_PLI, see Fig. 2, at which the solid velocity ($\boldsymbol{U^{pjt}}$) is imposed during the time of simulation.

The boundary regions are shown in Fig. 2 and the corresponding boundary conditions can be seen in detail below:

1. BD_FS (Fig. 2)
   (a) $P(t = 0s) = 0$
   (b) $u_i(t = 0s) = 0$
   (c) $u_1(t > 0s) = 0$
2. BD_W (Fig. 2)

(a) $P(t = 0s) = 0$
(b) $u_i(t \geq 0s) = 0$

3. BD_CB (Fig. 2)
   (a) $P(t = 0s) = 0$
   (b) $u_i(t \geq 0s) = 0$
4. BD_SI (Fig. 2)
   (a) $P(t = 0s) = 0$
   (b) $u_i(t = 0s) = 0$
   (c) $u_1(t > 0s) = 0$
5. BD_PLI (Fig. 2)
   (a) $P(t = 0s) = 0$
   (b) $u_i(t = 0s) = 0$
   (c) $u_2(t \geq 0s) = u_2^{pjt}$
   (d) $u_1(t \geq 0) = 0$

The projectile kinematics is described by its position along the symmetry axis (X_2^{pjt}), which defines the position of the BD_PLI, and is given by

$$\frac{dX_2^{pjt}}{dx_2} = u_2^{pjt} \tag{3}$$

where the vertical velocity (u_2^{pjt}) is computed from the linear momentum balance:

$$m^{pjt}\,\frac{du_2^{pjt}}{dx_2} = \int_{\Gamma_{BD\_PLI}} (P + \tau_{22}) d\Gamma_{BD\_PLI} + m^{pjt}\,g_2 \tag{4}$$

wheere $P + \tau_{22}$ is the normal stress in x_2-axis at the BD_PLI boundary, with wet surface area A^{pjt}, m^{pjt} is the mass of the projectile, and g_2 is the gravity component in the x_2 direction .

2.2 NUMERICAL SOLUTION

Is important to introduce some important notations at this point: $\boldsymbol{\phi} = \phi_n = \phi^n$ where ϕ is a property and n is the index in the vector representing a node in the domain mesh.

The solution is obtained by discretizing the fluid domain in a quadratic triangle 6 nodes mesh, setting up the boundary conditions at the boundary nodes in the Garlekin's FEM Global Matrix ($\boldsymbol{A^g}$), [6], solving a linear system $\boldsymbol{A^g X = B^g}$ for $\boldsymbol{X} = [\boldsymbol{u_1}\ \boldsymbol{u_2}\ \boldsymbol{P}]^T$ at each time step, obtained using the approximation below to compute the time derivatives, and the FEM for the space derivatives.

$$\frac{\partial u_i}{\partial t} \approx \frac{u_i(t + \Delta t) - u_i(t)}{\Delta t} \tag{5}$$

Then the Momentum equation becomes.

$$\rho\,[\frac{u_i}{\Delta t}(t + \Delta t) + u_j(t + \Delta t)\,\frac{\partial u_i}{\partial x_j}(t + \Delta t)] + \frac{\partial P}{x_i}(t + \Delta t) - \frac{\partial \tau_{ij}}{\partial x_j}(t + \Delta t) - \rho\,g_i = \rho\,\frac{u_i}{\Delta t}(t) \tag{6}$$

With equation 6 formulation one may conclude that the vector $\boldsymbol{B}$ is written as $\boldsymbol{B}(t + \Delta t) = \frac{1}{\Delta t}\,[\boldsymbol{u_1}(t)\ \boldsymbol{u_2}(t)\ 0\ \boldsymbol{P}]^T$. Once the $\boldsymbol{X}$ vector is defined the projectile velocity is evaluated by the following approximation of equation 4.

$$u_2^{pjt}(t + \Delta t) = u_2^{pjt}(t) - \Delta t \sum_{n=1}^{n_{BD\_PLI}} \frac{P^n + \tau_{22}^n}{n_{BD\_PLI}} \tag{7}$$

Be aware that n_{BD_PLI} is the number of nodes contained in the BD_PLI boundary region. Then the system boundary moves and a new mesh is generated, where the new nodal properties values are computed by interpolation on the old mesh property fields and then a new iteration starts.

The following sections show some details concerning the iterations steps.

2.2.1 GARLEKIN'S FEM

At each iteration in time domain, the set of equations described in the model section is solved in the system domain as show below.

$$\int_{\Omega} q \frac{\partial u_j}{\partial x_j}\, d\Omega = 0 \tag{8}$$

$$\int_{\Omega} v_j \{\rho\, [\frac{u_j}{\Delta t}(t+\Delta t) + u_i(t+\Delta t)\, \frac{\partial u_j}{\partial x_i}(t+\Delta t)] + \frac{\partial P}{x_j}(t+\Delta t) - \frac{\partial \tau_{ij}}{\partial x_i}(t+\Delta t) - \rho\, g_j\}\, d\Omega = \int_{\Omega} v_j \rho\, \frac{u_i}{\Delta t}(t)\, d\Omega \tag{9}$$

where v_j and q are appropriate test functions.

The integrals are further developed, reducing the second order derivatives $\frac{\partial}{\partial x_i}\frac{\partial u_j}{\partial x_i}$ contained in the $\frac{\partial \tau_{ij}}{\partial x_i}$ formulations of equation 2, by the integration by parts employing Green´s first identity, as described in details in [4].

Thus, employing the Galerkin's method, the resulting problem becomes a linear system of equations, by approximating the property ϕ in the triangle by the trial functions N_i, referred as shape functions, and the nodes values as described in the equation below.

$$\boldsymbol{\phi^n} = \sum_{i=1}^{n=6} N_i(\boldsymbol{X})\, \phi_i = N^i(\boldsymbol{X})\, \phi^i \tag{10}$$

As expressed at the beginning of this section, ϕ^n is the vector of property ϕ in the n^{th} element nodes and ϕ^i is the property value in the element's i^{th} node, with $1 \leq i \leq 6$ in the case of the quadratic triangle element.

The quadratic triangle element formulation of the shape functions implemented by this work can be seen in details by [5]. Using the base functions, the local derivatives in each triangular element, are given by.

$$\frac{\partial \phi}{\partial x_i} = \frac{\partial \mathbf{N^i}}{\partial x_i}\, \phi^{\mathbf{i}} \tag{11}$$

$$\frac{\partial \phi}{\partial x_i} = \sum_{i=1}^{n=6} \frac{\partial N_i(\boldsymbol{X})}{\partial x_i}\, \phi_i = \frac{\partial \mathbf{N^i}}{\partial x_i}\, \phi^{\mathbf{i}} \tag{12}$$

Be aware that $\frac{\partial \phi^i}{\partial x_j} = 0$, since ϕ^i is a constant nodal value to be determined at each iteration by the linear system $AX = B$.

Developing the integrals above using the Garlekin FEM approximations, the global matrix A^g and value vector B^g are assembled as shown in details in [4].

2.3 MESH GENERATOR

The simulator uses meshes representing cylindrical chambers which are characterized by their height and diameter. The mesh generation is done employing the Triangulate.jl, [3], Julia's module which is a wrapper of the Triangulate Generator Algorithm, [8] by keeping track of the movable domain boundary nodes. The coordinate system origin is taken as the lower bound middle point of the chamber section.

The meshes are generated with local refinement, keeping the regions closer to the BD_PLI and BD_FS filled with finer elements than the ones far from it. The refinement criteria is the triangular element barycenters smallest distances (bardist) to these boundary regions as illustrated by the code bellow.

```
if bardist .<= 0.2 then
    triangle_area <= 0.5e-3
elseif (bardist >= 0.2) && (bardist <=0.5))
    triangle_area <= 1e-3
else
    triangle_area >= 0.5e-2
end
```

The mesh refinement results can be seen in Fig. 3. In this example, a new mesh is generated at each iteration employing the above described refinement criteria.

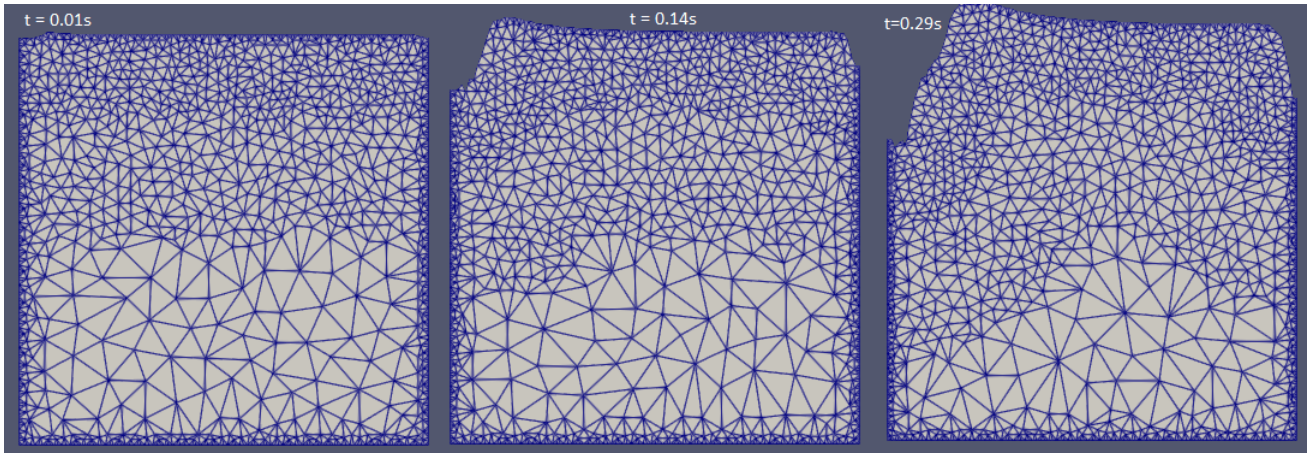

Figure 3. Refinement example, showing three different levels of refinement, according to the distance of the element to the closest surface boundary. Left: initial mesh. Center and right: meshes produced at two later times of the simulation.

2.3.1 FIRST MESH

The first mesh represent the fluid domain as the projectile is about to contact the surface. The domain is, therefore, represented describing a rectangle by its contour nodes, thus a 1d mesh. This 1d mesh is then employed to generate the initial 2d mesh, employing the refinement criteria already mentioned, and the triangulate.jl module. For more details see [3]. Figure 3 (left) illustrates a first mesh example.

2.3.2 SURFACE MOVEMENT

At each iteration the surface nodes only moves in the x_2-axis by the following procedure which is illustrated in the figure 4.

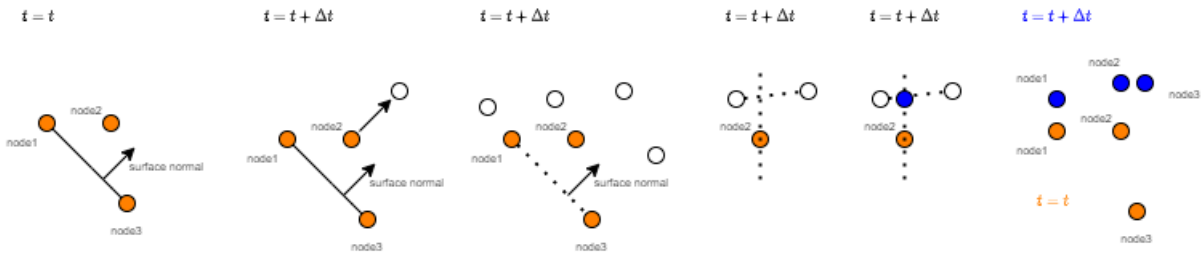

Figure 4. Surface Movement Procedure

For each surface node, the normal direction $\boldsymbol{n}$ is defined as normal to the vector $\boldsymbol{x}$ between the two closer surface nodes. This process is illustrated above in the first schematic and by the equations below.

$$\boldsymbol{x} = \frac{\boldsymbol{x}^{\boldsymbol{node3}} - \boldsymbol{x}^{\boldsymbol{node1}}}{||\boldsymbol{x}^{\boldsymbol{node3}} - \boldsymbol{x}^{\boldsymbol{node1}}||} \tag{13}$$

$$\boldsymbol{n} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \boldsymbol{x} \tag{14}$$

Then the nodes are allowed to move in the normal direction by the equation below as illustrated at the second and third schematics by the blank nodes.

$$\boldsymbol{x}'^{\boldsymbol{node2}}(t + \Delta t) = \boldsymbol{x}^{\boldsymbol{node2}} + \Delta t \, (\boldsymbol{n} \cdot \boldsymbol{u}^{\boldsymbol{node2}}) \, \boldsymbol{n} \tag{15}$$

The last step consist in finding the intersection between the x_2-axis parallel that contains each original node, orange ones, and the vector connecting the closest updated nodes, blank ones, as illustrated by the three last schematics.

The blue nodes represents the new surface nodes. It's important to highlight that the surface nodes in contact with the axis-symmetry interface and the ballistic chamber wall move in the x_2-axis direction with its original u_2 velocities.

2.3.3 MESH AT EACH ITERATION

After performing the surface movement, each internal node in the bulk fluid under the surface has its velocity defined as a linear interpolation between the velocity $(u_2^{\text{surface node}})$ of the surface at node, (see Fig. 5 in red), $(x^{\text{surface node}})$ along a vertical vertical line, and the bottom velocity. This velocity is defined by the shape functions of the surface element as described below.

$$u_2^{\text{surface node}} = N^i(\boldsymbol{x}^{\text{surface node}})\, u_2^i \tag{16}$$

The internal nodes (see Fig . 5 in pink) have it's velocity defined by the following equations.

$$u_2^{node} = \frac{x_2^{node}}{x_2^{\text{surface node}}}\, u_2^{\text{surface node}} \tag{17}$$

$$u_1^{node} = 0 \tag{18}$$

The described steps are illustrated by the figure 5

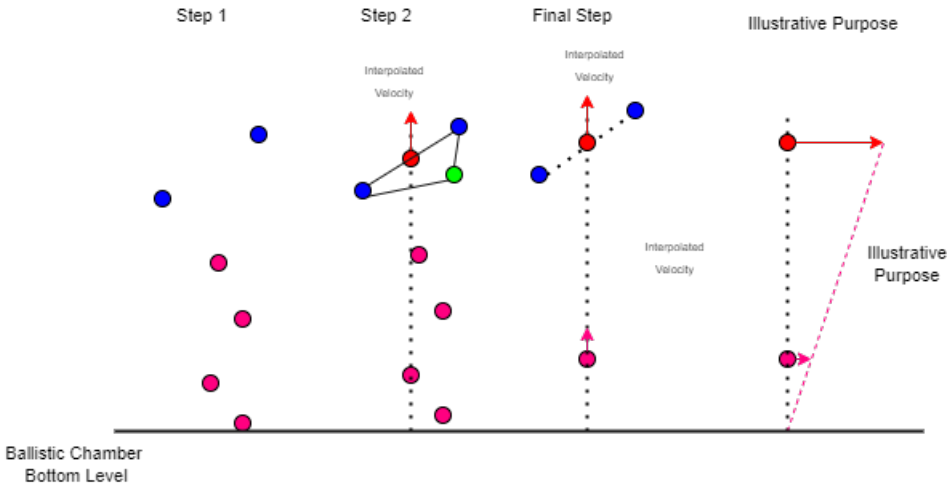

Figure 5. Internal Movement Procedure

Once defining the node velocity, its position is updated by the equation below.

$$\boldsymbol{x}^{\boldsymbol{node}}(t + \Delta t) = \boldsymbol{x}^{\boldsymbol{node}}(t) + \Delta t\, \boldsymbol{u}^{\boldsymbol{node}} \tag{19}$$

This procedure guarantees that no superposition of elements occur. After moving all the nodes, the new geometry has its boundary detected by finding the segments that are present in only one element by a algorithm similar to the one below.

```
for each segment in mesh
    count = 0
    for each triangle in mesh
        if segment in triangle then
            count+=1
        end
    end
    if count == 1 then
        push!(segment_nodes, boundary)
    end
end
```

With the boundary nodes, a new mesh is generated with the local refinement and new elements as in the figure below.

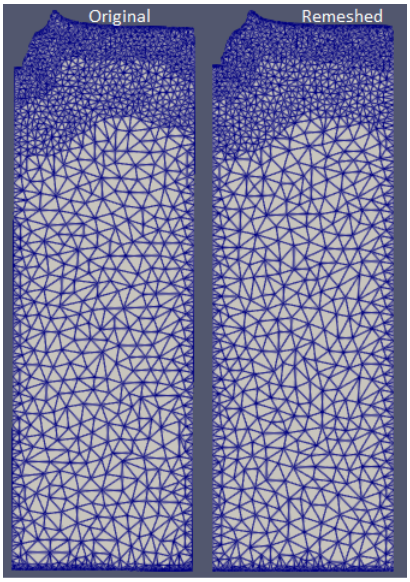

Figure 6. Remesh Example

2.3.4 MESH MAP

A the end of the time step, the properties in the new mesh nodes must be defined based on the the original mesh. This is done by the following simple interpolation algorithm.

```
for each node in new_mesh
    for each triangle in old_mesh
        if node in triangle then
            phi = 0
            for each old_node in triangle
                phi += N_old_node(node)*phi_old_node
            end
        end
    end
end
```

The performance of the algorithm can be verified by the mapping of u_2 in the example shown in Fig. 7, below.

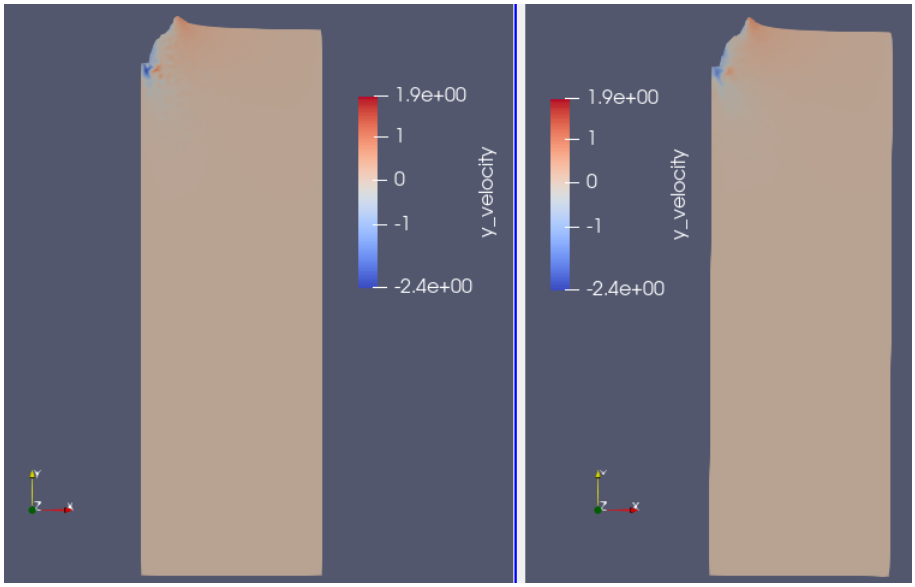

Figure 7. Mapping Performance

2.3.5 CHAMBER UPDATE

The simulation is initiated with data that the program receives as a input from the user, such as the initial chamber geometry height(h^{bch}) and radius(r^{bch}), the projectile radius(r^{ptj}) and its initial velocity, and the chamber height increment step (Δh^{bch}) .

If during the simulations the projectile doesn't get to a percentage of the chamber height from the bottom with a velocity smaller than $u_1 = 0.1mm/s$ the chamber height is incremented by Δh^{bch} and the whole simulation starts over.

3. CASE OF STUDY

For demonstration purposes, a case of study was performed, considering a chamber with $h^{bch} = 1m$ and $r^{bch} = 1m$, a projectile with $r = 0.05m$, velocity $u_2^{pjt} = 10m/s$ and $0.1kg$ of mass, a height increment of $\Delta h^{bch} = 0.2m$ and distance from the bottom criteria as 40% of the total height. The chamber is filled with water at $20°C$, with $\mu = 1.003\,10^{-3}m/s^2$ and $\rho = 998kg/m^3$ ([9]) and with a gravitational field of $g_1 = 0$ and $g_2 = -9.81m/s^2$. The minimum height to decelerate the projectile in the above conditions was evaluated as $h^{bc} = 3.2m$. The results of the simulations were obtained in 2.6h (wall-clock time), running on a Windows 10 PC with Intel(R) Core(TM) i3-6006U (2 core, 4 threads) CPU 2.00 GHz and 8GB of RAM.

4. CONCLUSION

The proposed algorithm is functional, flexible, simple, of easy maintenance and expansion and relatively fast. The Julia language makes possible to create such complex program with simplicity and more human like syntax as python. As a work in progress the program has shown potential to reach additional goals as:

- Chamber radius increment.
- Chamber walls solid mechanics analysis using the Von Mises yield criterion.
- The energy equation implementation.
- Non axis-symmetric analysis.

Some of the steps described in the proposed algorithm are not required to be performed at all iterations, thus providing an economy of computer resources. For instance, the remesh mapping routines can be executed periodically or when a mesh quality measure points that the mesh is becoming too distorted. The Julia language works faster with explicit type declarations, which are not completely implemented in the current version of the code. The next steps to be taken are the evaluation of wall stresses and real data comparisons to validate the developed tool.

5. ACKNOWLEDGEMENTS

The authors thank FAPERJ (Research Support Foundation of the State of Rio de Janeiro), CNPq (National Council for Scientific and Technological Research), and CAPES (Coordination for the Improvement of Higher Education Personnel) for their financial support.

6. REFERENCES

References

[1] Jeff Bezanson et al. "Julia: A fast dynamic language for technical computing". In: *arXiv preprint arXiv:1209.5145* (2012).

[2] Filippo Coletti. "3 1 FLOW, Department of Engineering Mechanics, KTH Royal Institute of Technology, Stockholm, Sweden; email: 2 Department of Energy and Process Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, Norway 3 Department of Mechanical and Process Engineering, ETH Zurich, Zurich". In: *Switzerland Annual Review of Fluid Mechanics* 54 (), pp. 159–189.

[3] Jürgen Fuhrmann. *Triangulate*. `https://github.com/JuliaGeometry/Triangulate.jl.git`. 2020.

[4] Antonio Huerta Jean Donea. *Finite Element Methods for Flow Problems*. 1th. WILEY, 2003.

[5] Peter I. Kattan. "The Quadratic Triangular Element". In: *MATLAB Guide to Finite Elements: An Interactive Approach*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 249–273. ISBN: 978-3-540-70698-4. DOI: 10.1007/978-3-540-70698-4_12. URL: `https://doi.org/10.1007/978-3-540-70698-4_12`.

[6] P. Nithiarasu O.C. Zienkiewicz R.L. Taylor. *The Finite Element Method for Fluid Dynamics*. 7th. Elsevier Ltd, 2014.

[7] Carolina Ricardo et al. “Onde mora a impunidade: Porque o Brasil precisa de um Indicador Nacional de Esclarecimento de Homicídios”. In: (2020). URL: https://soudapaz.org/o-que-fazemos/conhecer/pesquisas/politicas-de-seguranca-publica/controle-de-homicidios/?show=documentos.

[8] Jonathan Richard Shewchuk. “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator”. In: *Applied Computational Geometry: Towards Geometric Engineering*. Ed. by Ming C. Lin and Dinesh Manocha. Vol. 1148. Lecture Notes in Computer Science. From the First ACM Workshop on Applied Computational Geometry. Springer-Verlag, May 1996, pp. 203–222.

[9] Frank M. White. *Fluid Mechanics*. 4th. McGraw-Hill, 2001.

7. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.