3. Results and discussion

3.1 Simulation in the one-dimensional mesh

In the one-dimensional mesh generated in Section 2.3.1, we ran the simulation of the model implemented in OpenFOAM. The simulation was successful, verifying that there were no errors in the calculations during the process. The time required to simulate 25 hours was 2 hours and 48 minutes, with only 100 cells in the mesh.

Before checking for numerical leaking, we ensured that the obtained values were physically correct, meaning that no values exceeded the maximum range implemented in the simulation, and no negative values were generated. All the model variables were validated, and, for example, the volume fraction variables (alpha and psi) are represented in Fig 1. This figure confirms that there are no negative values or values greater than 1, as alpha and psi range goes from 0 to 1.

Fig 1 also represents the intervals where the interface (0 < α & ψ < 1), gas phase ( α = 0 & ψ = 1) and liquid phase ( α = 1 & ψ = 0) are present in the domain. The respective intervals are x = [0.069, 0.177] m, x = [0, 0.069] m, and x = [0.178, 0.300] m. These intervals allow us to verify whether there is leakage of one phase into another.

Fig FG (a) and (b) illustrates the evolution of α\*SH2 and α\*XPB generated in the simulation. For the former, we used Eq. 9 to determinate the excess percentage (Fig. FGc), while for alpha\*XPB, the excess percentage was calculated using Eq. 6 as a baseline, applied to XPB (Fig FGd). Moreover, the same equation was used as a baseline to compute the excess percentage for all parameters, allowing us to assess wether numerical occurs in any variable.

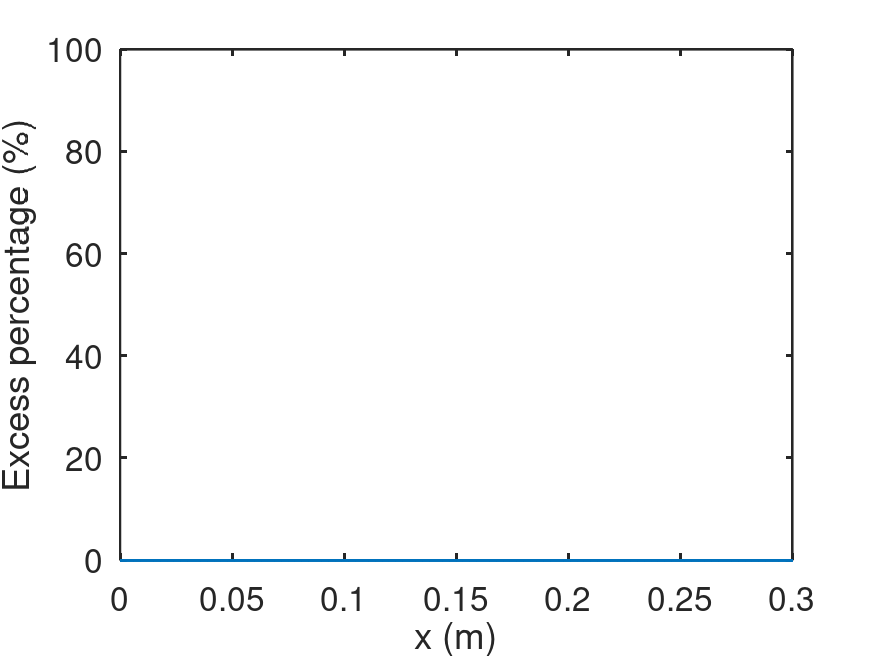
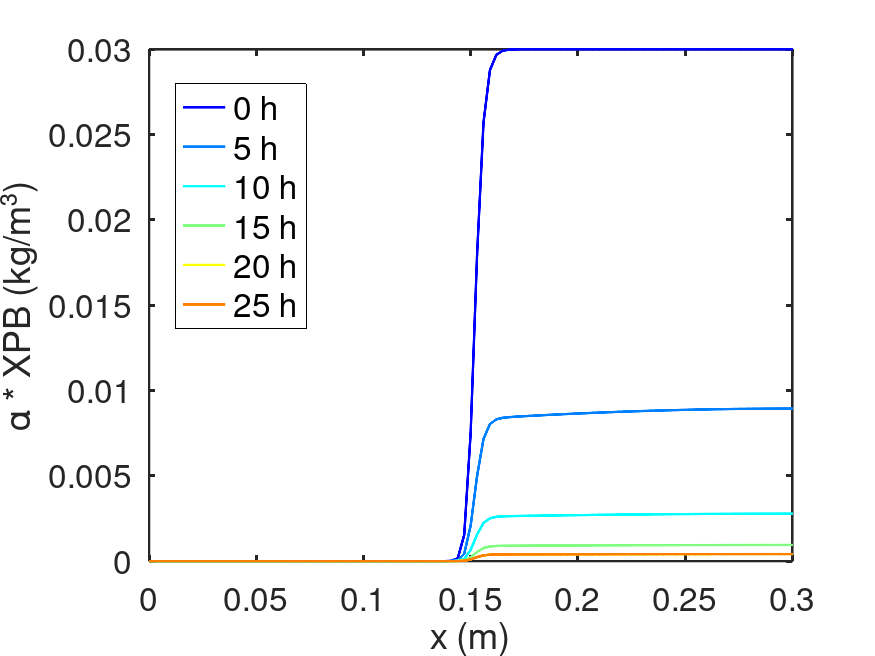
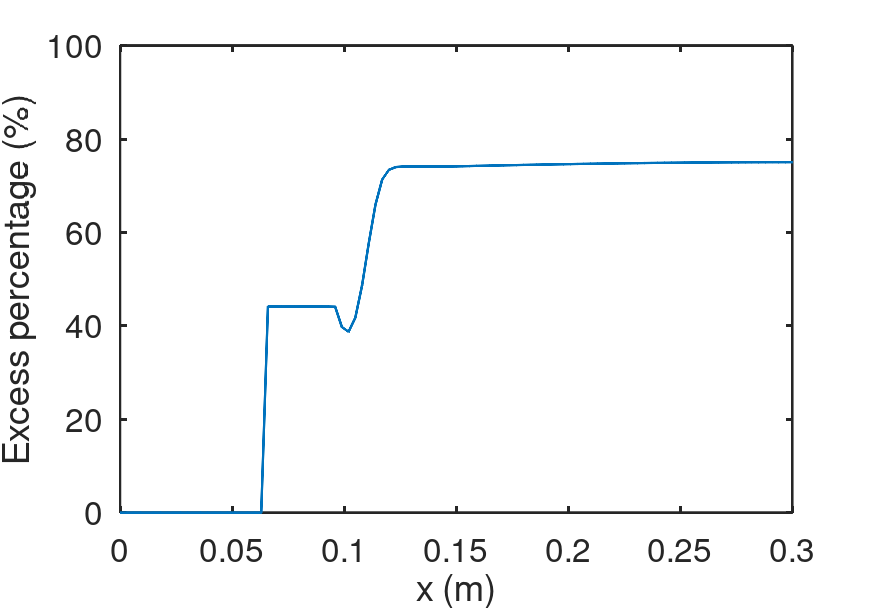
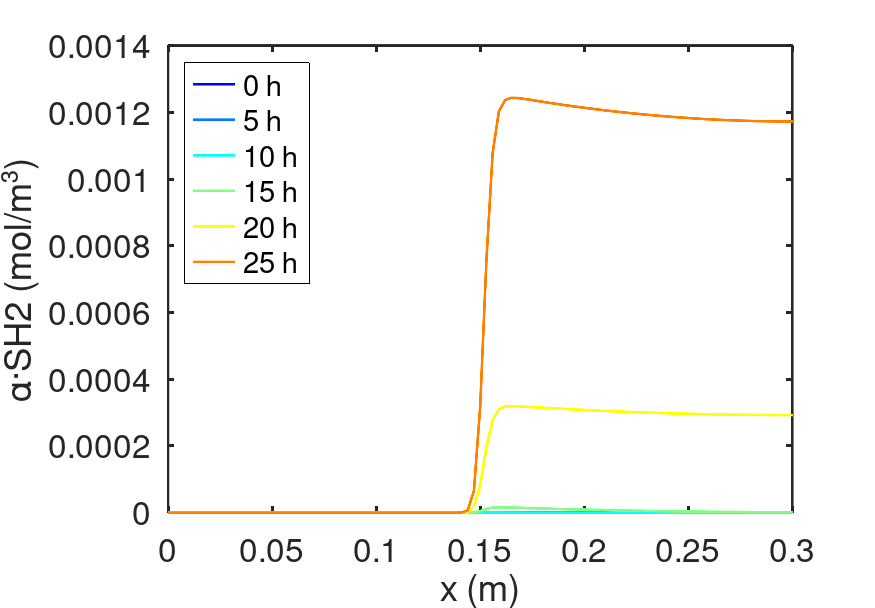


Fig: Evolution of the variables during the simulation of 25 h. (a) alpha\*SH2. (b) alpha\*XPB. Excess percentage. (c) alpha\*SH2. (d) alpha\*XPB.

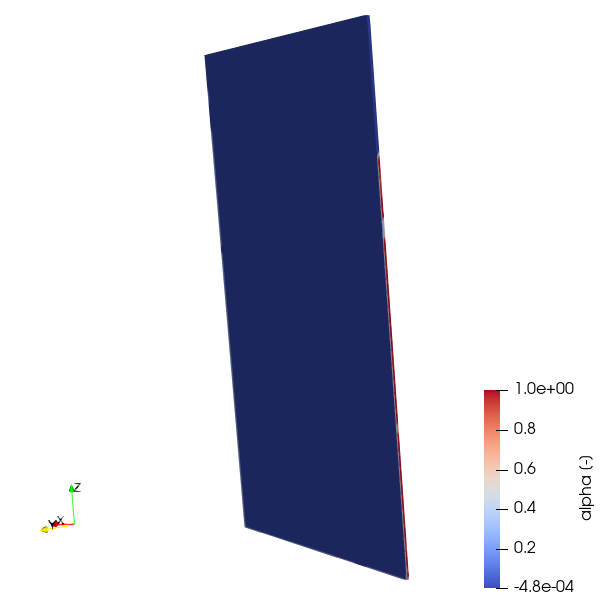
As discussed above, the numerical leakage is represented by the excess percentage. Both alpha\*SH2 and alpha\*XPB can only exist in the liquid phase, if they appear in the gas phase, the scalar is not confined to its designed phase, leading to numerical leakage.

For dissolved hydrogen, there is no numerical leakage due to the excess percentage is not present in the gas phase, even though having some excess in the liquid phase. Meanwhile, for the purple phototrophic bacteria biomass, there is no excess percentage throughout the domain, indicating the absence of numerical leakage.

Similarly, no numerical leakage is observed in the other variables (See Appendix). Therefore, the assumption made in the Section 2.3.1, checking the correctness of calculations, ensuring physically consistent results, and detecting numerical leakage, are confirmed to be valid.

3.2 Simulation in the compartment model

The simulation of the implemented model in the mesh generated in Section 2.3.2, presented some difficulties and complexities. First, we observed that the splitMeshRegions utility leads to the creation of a new boundary condition to separate the newly created section from the original. This boundary condition created does not use the zero-gradient condition, which specifies that the variable in question has no variation in the direction normal to the boundary (Greenshields, 2020). Instead, it applies a calculated boundary with a uniform value of 0 for the phase field variable alpha. As a result, a blue plane is observed in Fig Q, were the values of alpha are equal to 0.



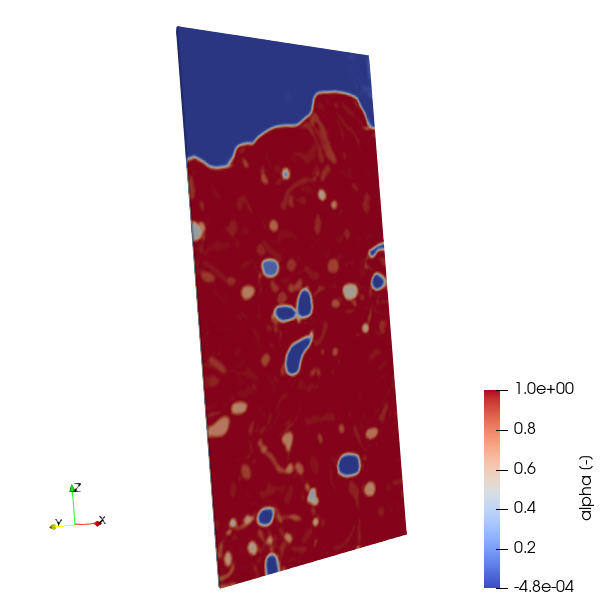


Fig Q: Limitations of the splitMeshRegions utility.

Despite the presence of this blue plane, we ran the simulation despite and observed that it took to long just to obtain results, even for short time steps, due to the high number of cells. Subsequently, a ‘floating point exception’ error emerged, which can be related to an illegal mathematic operation, such as division by zero, the apparition of a negative value, or an issue with a boundary condition.

In order to avoid the ‘floating point exception’ error, we explored several alternatives to solve the problem, which are explained in the following sections.

3.2.1 Change the boundary condition

For a correct simulation, the foamDictionary utility was used to change the boundary condition that generates the blue plane in Fig Q. This tool enables the substitution of the uniform 0 values of alpha with the zero-gradient condition.

This change was successfully applied, and we executed the simulation. The run process was completed properly. Thus, we tested the execution of the model for a few seconds to verify wether the results were logical. However, we observed that the variables SH2, CO2gas and XC had negative values (Fig L) due to the generation of some negatives values and others greater than 1 in alpha, which are physically incorrect and can cause the simulation to explode.

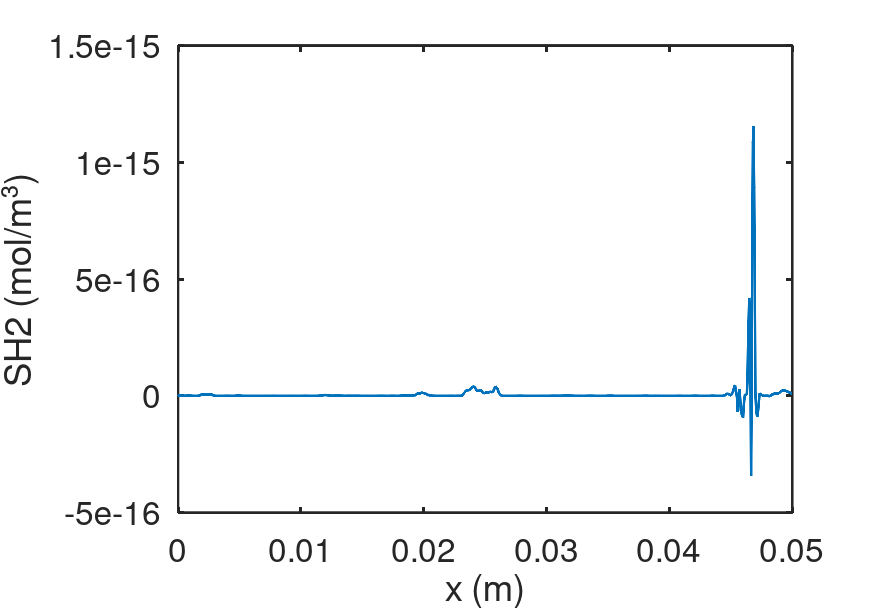
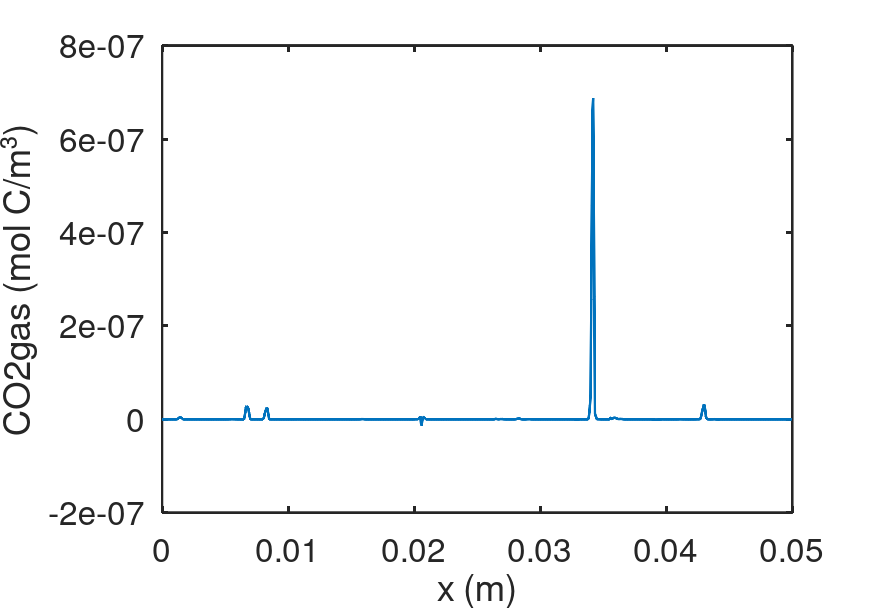
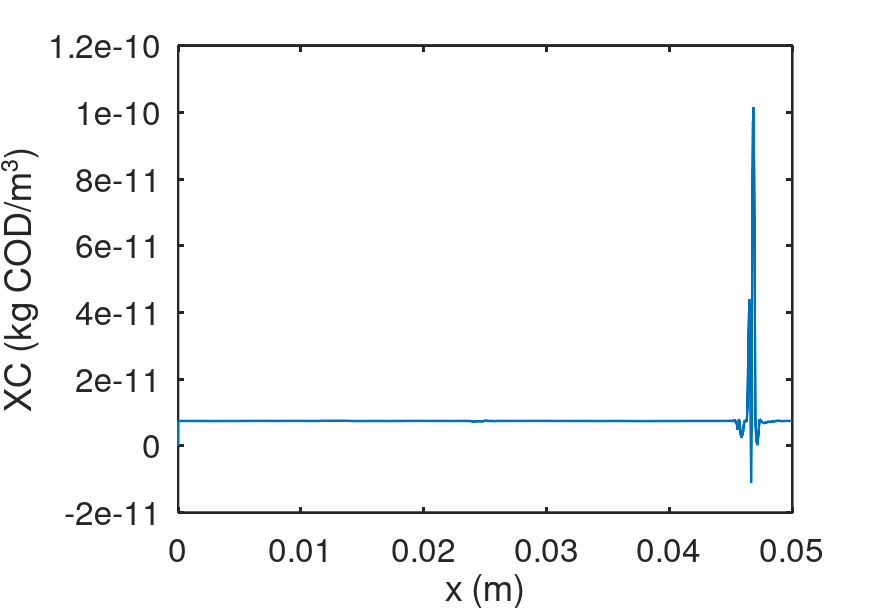


Fig: Negative values after modifying the boundary condition with foamDictionary. (a) XC (b) CO2gas (c) SH2.

This simulation was performed from 0 to 0.01325 seconds, using a time step of 1e-5, and lasted 2 minutes, indicating that reaching 25 hours of simulation (similar to the one-dimensional mesh) would require an extremely long runtime. Using 1e-5 as a time step, the simulation would take approximately three and a half weeks to reach 25 hours.

To reduce computational time, we attempted to use a larger time steps (1e-3). However, this caused the simulation to become unstable and resulted in the ‘floating point exception’ error.

3.2.2 Change format

By default, the alpha values in OpenFOAM are written in binary format, which stores data in a machine-readable way. To modify the negative values and those greater than 1 for alpha, we need to convert the binary format into a human-readable one. This allows us to correct the alpha values to ensure a successful simulation of the model.

Thus, we changed the format from binary to ASCII adjusting values of alpha greater than 1 to 0.9999 and replacing negative values with 1e-6. However, the simulation immediately resulted in a ‘floating point exception’ error. Regardless of the changes made to alpha, the error occurred instantly after converting the format. Even after switching back from ASCII to binary, the simulation still failed, indicating that once the format is changed, the simulation will inevitably lead to a ‘floating point exception’ error.

3.2.3 Data transfer

After the failures to simulate the model in the compartment model, we explored the following alternatives which consist in the data transfer from the full geometry (Fig Da) into a one-dimensional mesh that we will create instead of picking a section of the original mesh, to facilitate the computational cost and time.

#### 3.2.3.1 MapFields

MapFields is an OpenFOAM utility used to transfer data from one geometry into another. In our case, it can be used for numerical transfer of results from the source geometry (quarter-cylinder) to a target geometry (FIG 8), consisting of an axisymmetric wedge with an opening angle of 5°. This type of geometry possesses symmetry around an axis, which implies that the variables do not change in the circumferential direction. Additionally, we ensured the plane alignment between the source and target geometry to perfectly match, as well as the correct assignment of the boundary conditions to guarantee the proper execution of the simulation.



Fig 8: Representation of the axisymmetric wedge created using blockMesh.

The functionality of this tool is represented in Fig 9. It can be used with identical geometries when the meshes characteristics match perfectly. We use the option “- consistent” to simplify the mapping of the fields, or it can be applied to different geometries. In the latter, it is necessary to specify the boundary conditions through a MapFieldDict dictionary, located in the source geometry case. In MapFieldsDict, two elements can be specified: patchMap and cuttingPatches. The former defines identical boundary conditions between the source and target geometries, ensuring that data is correctly transferred for these matching areas. The latter specifies the parts of the contour that require cutting or spatial adjustment to properly align the transferred fields between the two geometries.

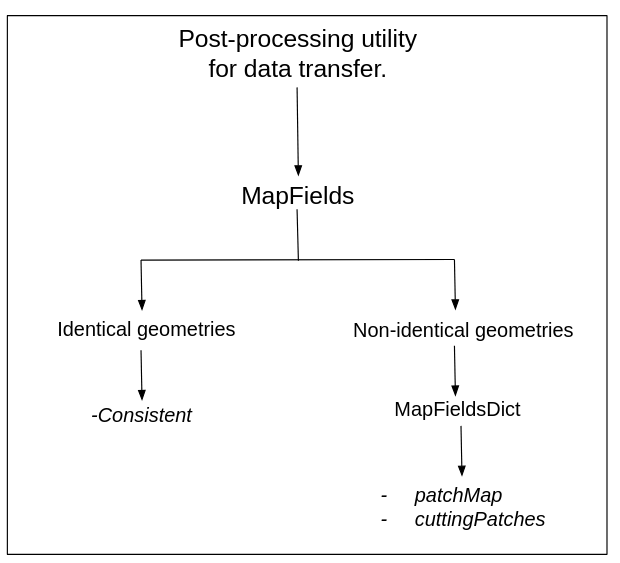
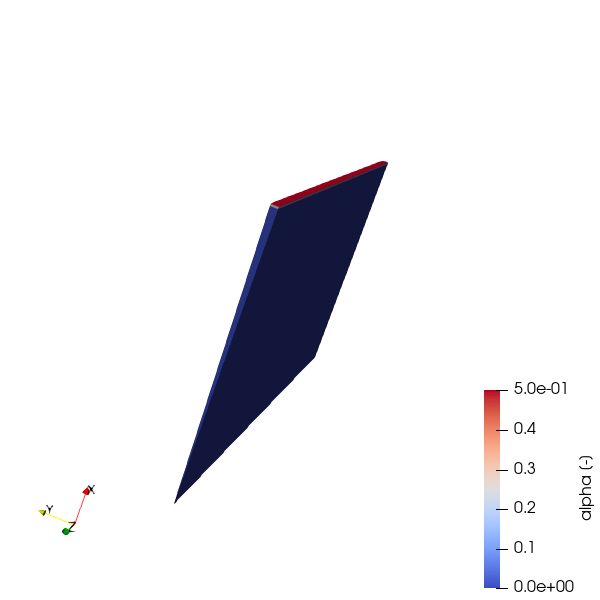
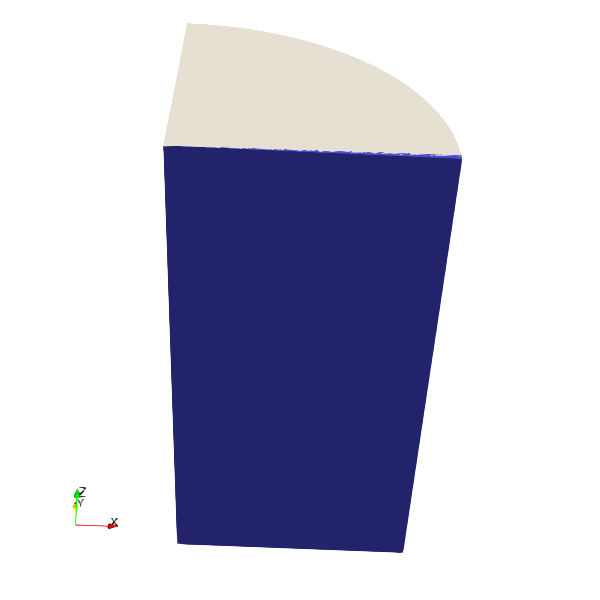


Fig 9. Functionality of the MapFields utility.

Once patchMap and cuttingPatches are specified, the MapFields utility interpolates the configured variables in the original mesh between the two geometries, ensuring that the axisymmetric wedge maintains consistency with the results of the full geometry simulation. As a result, the following figure illustrates the execution of MapFields:



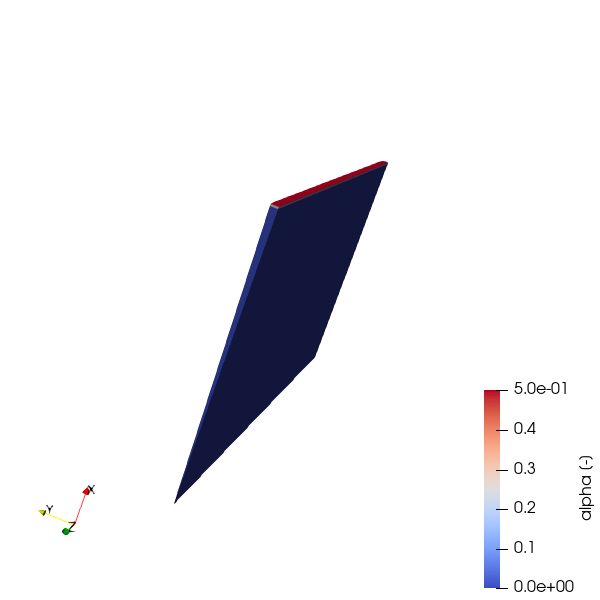
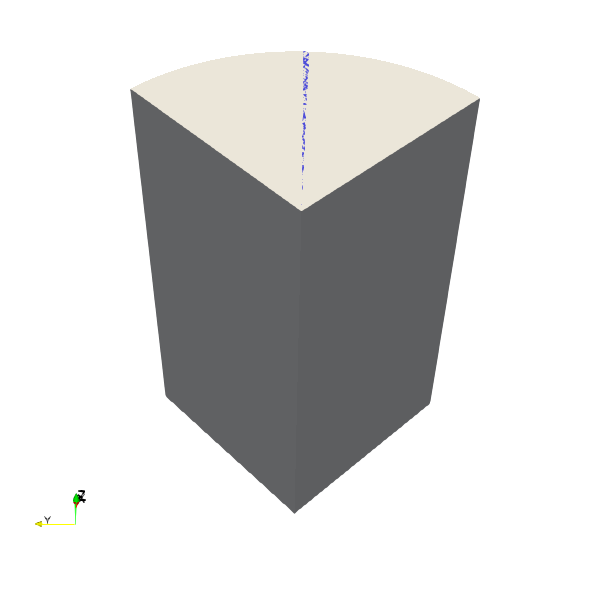


Fig 10: Representation of the axisymmetric wedge location within the quarter-cylinder. (a) At the end, (b) Along the bisector. Bubble distribution results using MapFields. (c) At the end, (d) Along the bisector.

The figures above shows that the data transfer between the geometries has been deficient, and the representation of the phase-field variable alpha is incorrect when compared to the data of the quarter-cylinder in Fig l. Once these inconsistencies were identified, adjustments were made to the contours in patchMap and cuttingPatches. However, the results obtained remained the same as in Fig 10. Hence, we concluded that the MapFields utility is not suitable for this geometric transformation. This could be due to the high number of cells in the angular direction of the quarter-cylinder. Its a possibility that this utility calculates the average of these values and transfer them into the wedge.

#### 3.2.3.1 Slice

In this section, the data transfer is performed manually instead of using an OpenFOAM post-processing utility. The goal is to copy the velocity and alpha values from the quarter-cylinder and map them into a mesh created. To reduce the number of cells used in the quarter-cylinder, we applied the Slice filter in ParaView to extract a cross-section of the full geometry (Fig 11). The resulting slice contains 88200 cells.

 Fig 11. Cross-section obtained using the Slice filter in ParaView.

After extracting the slice, we transferred the alpha and velocity values to a new mesh generated with blockMesh, with the same geometry, number of cells, and dimension (0.05 m x 0.0005 m x 0.1 m) as the slice. However, despite transferring the values correctly, OpenFOAM did not preserve the expected order (Fig 12), leading to a disordered bubble distribution.

Even tough, the model was executed and resulted in a ‘floating point exception’ error dirung the simulation.

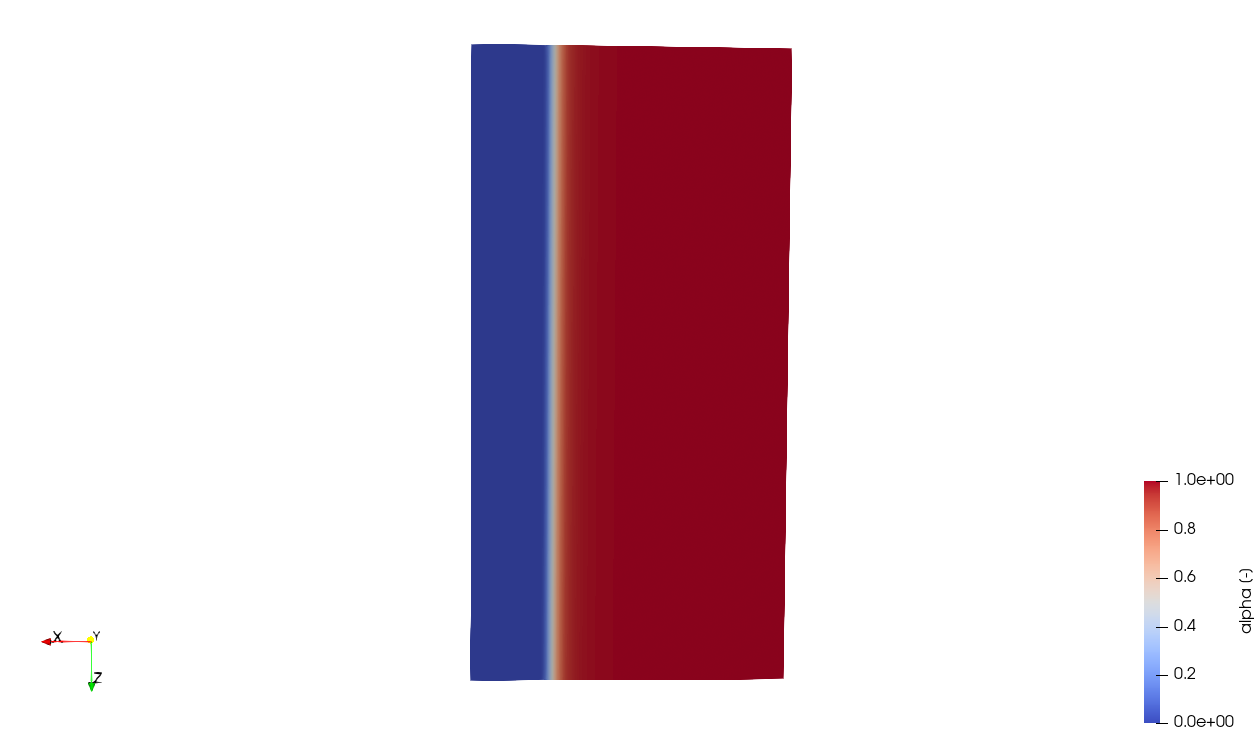


Fig 12: Bubble distribution after transferring data from the cross-section to the new mesh generated.

3.3 Development of the graphical user interface

To simplify the execution of the implemented model in OpenFOAM, a graphical user interface (GUI) was created using guiEditor, a GNU Octave utility that enables visual editing of interfaces by generating scripts from a native function calls, which can be reproduced (Burgos and Adam, 2020).

The developed GUI is designed for use with any OpenFOAM case, as at includes an option to specify the case directory path and another to define the output file name (Fig 13a), where the simulation data is stored during the execution. Once the path and output file are set, clicking in the ‘Next’ button opens the main interface (Fig 13b), which contains four buttons: Physical Parameters, Time, Run and Plot results.

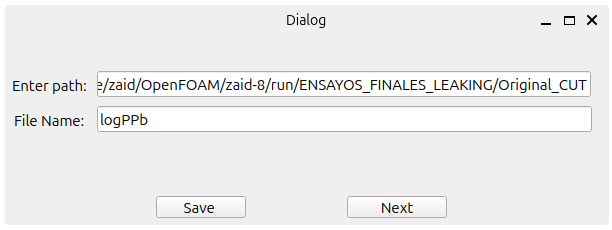
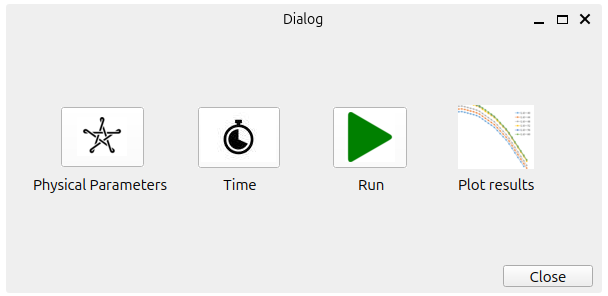


Fig 13: Graphical user interface for OpenFOAM case execution. (a) Interface for entering the case path and output file name. (b) Main interface with primary function buttons.

Our developed GUI allows users to modify all relevant physical parameters with a single click, including stoichiometric, kinetic, inhibition, pH and gas transfer parameters, directly within the OpenFOAM case specified in the selected path.

A similar approach is used for time settings (Fig 14), where users can define the start time, end time, time step, and write interval for the execution of the implemented model.

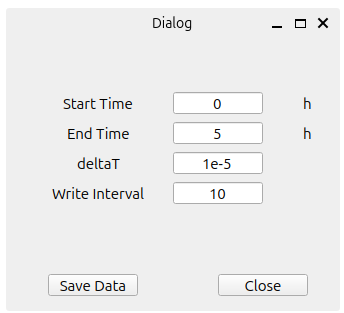


Fig 14: Interface for configuring time parameters.

Once the time and physical parameters are configured, users can click in the ‘Run’ button to execute the implemented model directly.

Additionally, to visualize the results a ‘Plot results’ button is available, allowing users to display any variable from their corresponding case. This button, illustrated in Fig 15, enables users to select the time range for plotting results. It also includes, three input fields, where users must specify the .csv file containing the results exported from ParaView. Furthermore, users need to input the column numbers corresponding to x, alpha, and the variable of interest to generate an accurate plot.

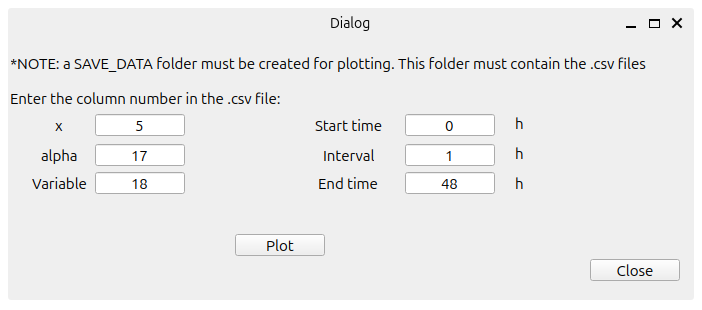


Fig 15: Interface for selecting time range and variables for result plotting.

All the interfaces are contained within a guiEditor project. The latter, can be saved as a package and then installed directly in GNU Octave.

The GNU Octave scripts used to run the model directly, modify the time and physical parameters, and plot the results, contained within guiEditor, can be found in GitHub (<https://github.com/TEMSAMANI8Z/guiEditor_graphINTERFACE/tree/main>)

REFERENCES

Janke, D., Caiazzo, A., Ahmed, N., Alia, N., Knoth, O., Moreau, B., Wilbrandt, U., Willink, D., Amon, T., John, V., 2020. On the feasibility of using open source solvers for the simulation of a turbulent air flow in a dairy barn. Computers and Electronics in Agriculture 175, 105546. <https://doi.org/10.1016/j.compag.2020.105546>

(Janke et al., 2020)

Islas, A., Rodríguez-Fernández, A., Betegón, C., Martínez-Pañeda, E., Pandal, A., 2022. CFD simulations of turbulent dust dispersion in the 20 L vessel using OpenFOAM. Powder Technology 397, 117033. <https://doi.org/10.1016/j.powtec.2021.117033>

(Islas et al., 2022)

He, Z., You, J., Kang, D., Zou, Q., Zhang, W., Zhang, Z., 2024. Overall numerical simulation of chemical-thermal-electric conversion for an all-in-one thermoelectric generator based on micro scale combustion. Energy 292, 130307. <https://doi.org/10.1016/j.energy.2024.130307>

(He et al., 2024)

Capson-Tojo, G., Batstone, D.J., Hülsen, T., 2023. Expanding mechanistic models to represent purple phototrophic bacteria enriched cultures growing outdoors. Water Research 229, 119401. <https://doi.org/10.1016/j.watres.2022.119401>

(Capson-Tojo et al., 2023)

Puyol, D., Barry, E.M., Hülsen, T., Batstone, D.J., 2017. A mechanistic model for anaerobic phototrophs in domestic wastewater applications: Photo-anaerobic model (PAnM). Water Research 116, 241–253. <https://doi.org/10.1016/j.watres.2017.03.022>

(Puyol et al., 2017)

Greenshields, C., 2020. OpenFOAM v8 User Guide [WWW Document]. CFD Direct. URL <https://doc.cfd.direct/openfoam/user-guide-v8/index/> (accessed 3.7.25).

(Greenshields, 2020)

Qin, R., Bhadeshia, H., 2010. Phase field method. Materials Science and Technology 26, 803–811. <https://doi.org/10.1179/174328409X453190>

(Qin and Bhadeshia, 2010)

Jain, S.S., Mani, A., 2023. A computational model for transport of immiscible scalars in two-phase flows. Journal of Computational Physics 476, 111843. <https://doi.org/10.1016/j.jcp.2022.111843>

(Jain and Mani, 2023)

Burgos, E.S., Adam, E.J., 2020. Graphical user interface editor for Octave applications. Engineering Reports 2, e12269. https://doi.org/10.1002/eng2.12269

(Burgos and Adam, 2020)