Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Short note

A machine learning strategy for computing interface curvature in Front-Tracking methods

Hugo L. França [a], Cassio M. Oishi [b,*]

[a] Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo, São Carlos, Brazil
[b] Departamento de Matemática e Computação, Faculdade de Ciências e Tecnologia, Universidade Estadual Paulista "Júlio de Mesquita Filho", Presidente Prudente, Brazil

ARTICLE INFO

Article history:
Received 12 July 2021
Received in revised form 10 October 2021
Accepted 17 November 2021
Available online 23 November 2021

Keywords:
Machine learning
Front-Tracking
Curvature
Free surface flows
Marker-and-cell

ABSTRACT

In this work we have described the application of a machine learning strategy to compute the interface curvature in the context of a Front-Tracking framework. Based on angular information of normal and tangential vectors between marker points, the interface curvature is predicted using a neural network. The Front-Tracking-Machine-Learning method is validated using a sine wave and then applied in combination with a Marker-And-Cell method for solving a complex free surface flow. Our results indicate that it is feasible to employ machine learning concepts as an alternative approach for computing curvatures in Front-Tracking schemes.

1. Introduction

The Front-Tracking (FT) scheme [5,24] is considered a powerful methodology for solving complex fluid flows involving free surface/interface dynamics. Roughly speaking, the implementations on FT frameworks use a set of discrete marker points (Lagrangian points) to represent the interface, and the surface tension effects can be numerically computed using the geometrical information from this data structure. Important progresses and improvements on FT implementations have been presented in the last decades for multi-phase codes [18,22] as well as for single-phase problems [10,23].

A particularly common strategy in the construction of FT codes is to choose a type of continuous curve (e.g., piecewise parabolic, cubic splines, etc) that will fit the discrete marker points. The analytical expression of this curve can be used to approximate the curvature. In contrast to classical implicit or Eulerian formulations [17], e.g. Volume-of-Fluid (VOF) [6] and Level-Set (LS) [14], the estimation of the interface curvature in FT schemes is generally not considered a computational challenge. Particularly, in order to numerically compute the interface curvature in a FT framework, we can adopt approximation techniques using the Lagrangian markers, e.g., Frenet based methods [4] or curve approximation schemes [10,21]. However, some authors have already pointed out drawbacks on the accuracy [2] and stability [10] of curve approximation schemes. Therefore, new approaches to compute the curvature in FT schemes are still desirable in order to further improve the classical strategies.

Recently, an alternative strategy for computing curvature based on Machine Learning (ML) algorithms has been employed in the context of VOF schemes. The VOF-ML idea was originally presented by Qi et al. [19] to deal with two-dimensional problems while Patel et al. [16] have extended the method for solving 3D fluid flows. In VOF-ML frameworks, a neural

* Corresponding author.
E-mail address: cassio.oishi@unesp.br (C.M. Oishi).

https://doi.org/10.1016/j.jcp.2021.110860

network is modeled which receives local volume fractions as inputs, and then the interface curvature is predicted as an output. This network is trained using a large synthetic dataset composed from circles and spheres. The numerical results of Qi et al. [19] and Patel et al. [16] highlighted that these machine learning strategies can be considered as alternative tools for simulating surface-tension-driven flows. Following the ideas of Qi et al. [19], a Level-Set Deep Learning approach has been developed and used by Larios-Cárdenas and Gibou [7,8] in order to calculate curvature more accurately based on the implicit level-set functions. In addition, a machine learning methodology has been also employed in [9] to improve the simulations of multiphase flows in the context of the particle method.

According to the best of our knowledge, these learning strategies have never been employed for directly calculating the curvature of FT schemes. Therefore, our main motivation in this work is to present a simple adaptation of the VOF-ML scheme by Qi et al. [19] resulting in a FT-ML methodology. In order to further demonstrate the applicability of this strategy for solving a complex fluid flow, we have adopted the FT-ML in our Marker-And-Cell (MAC) in-house code [11,3,13] to investigate the outcomes in binary collisions of droplets.

2. Learning curvature and testing

In this section we describe a new machine learning scheme used to approximate curvature of 2D curves given by the Front-Tracking representation.

In the VOF-ML method [19], the curvature was approximated on a given grid cell (i, j). Analogously, in our Front-Tracking representation, the curvature will be computed on a marker particle p_i. The curvature on the marker p_i is approximated by a function with parameters F_i, each related to one of the $2N$ line segments around p_i, as illustrated in Fig. 1a.

Following this idea, the discrete curvature κ_i is given by

$$h \cdot \kappa_i = C\left(\begin{bmatrix} F_{i-1} & F_{i-2} & \cdots & F_{i-N} \\ F_i & F_{i+1} & \cdots & F_{i+N-1} \end{bmatrix}\right), \tag{1}$$

where C is a selected function. Since the curvature has dimension one over length, we adopt a non dimensional curvature by multiplying by h, where h is the mean length of the $2N$ line segments used.

Each parameter F_i is defined as a set of two values (see Fig. 1b):

- Value 1: the angle between the positive x axis and the outwards normal vector of the segment $[p_{i+1}, p_i]$.
- Value 2: same as value 1 for the oriented tangential vector.

Fig. 1 illustrates the parameters used for a given point p_i using $N = 3$. Note that the function C in (1) is actually a function of $4N$ scalar parameters, since each F_i has two elements.

In the current work, the function C will be represented by a neural network with 4N scalar inputs and one output, as shown in Fig. 2.

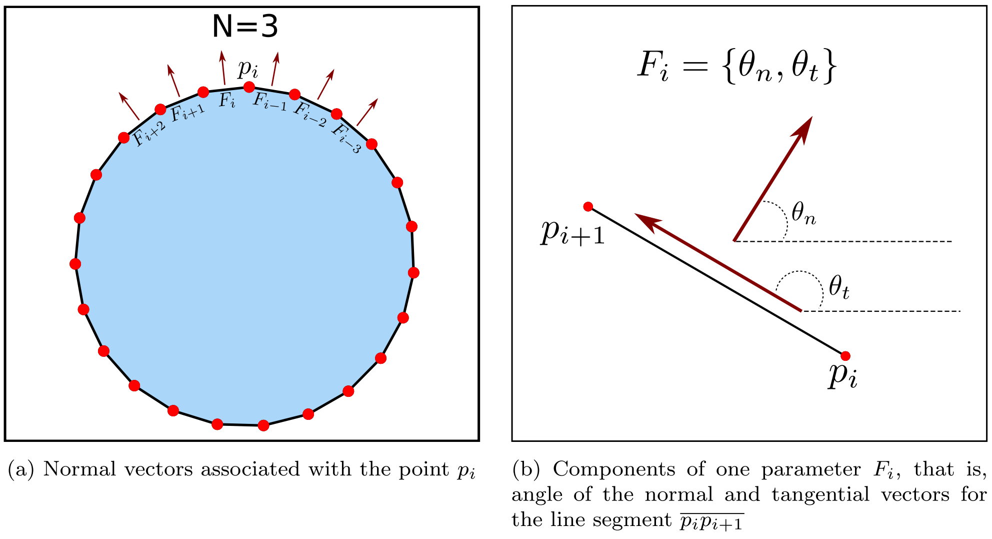
To train this network, we require a dataset with entries associated to curves with known curvature (see [19]). Different types of curves with analytical expressions for the curvature could be used to generate this dataset. The process to generate the dataset can be summarized as follows:

1. We generate 65 circles with radii varying from 0.00225 to 0.475. An exponential function is used to sample points in this interval, such that we have more circles with smaller radii (closer to 0.00225). The curvature of each circle is trivially given by $\kappa = \frac{1}{R}$, where R is the circle radius.
2. All circles are discretized using line segments with the same constant size h, which is a chosen parameter (discussed later). For $h = 0.0004$, the smallest circle will have 35 segments, for example. The marker particles for these segments are initially created in counter-clockwise order.
3. We iterate over each marker particle of each circle and, for each marker, two entries will be generated in the dataset:
   (a) one entry is created with normals pointing outside of the circle and assuming positive curvature $h \cdot \frac{1}{R}$, as illustrated in Fig. 3 (left);
   (b) the second entry is created with normals pointing into the circle and assuming negative curvature $-h \cdot \frac{1}{R}$, as illustrated in Fig. 3 (right).
4. The three steps above are repeated but, in the second step, the markers are generated in clockwise order.

Executing the steps above we obtain a dataset with thousands of entries, depending on the value of h chosen.

From now on we will use a neural network with the following structure and training parameters:

- Number of hidden layers: 3 layers with 200 neurons each;
- Activation function: reLU for hidden layers and linear for output layer;
- Loss function: mean squared error;
- Optimization algorithm: gradient descent with learning rate 0.01;
- Number of training epochs: 3000.


N=3
p_i
F_{i+2} F_{i+1} F_i F_{i-1} F_{i-2} F_{i-3}
$F_i = \{\theta_n, \theta_t\}$
θ_n
θ_t
p_{i+1}
p_i

(a) Normal vectors associated with the point p_i

(b) Components of one parameter F_i, that is, angle of the normal and tangential vectors for the line segment $\overline{p_i p_{i+1}}$

Fig. 1. Description of the parameters F_i used to approximate the curvature at the point p_i.

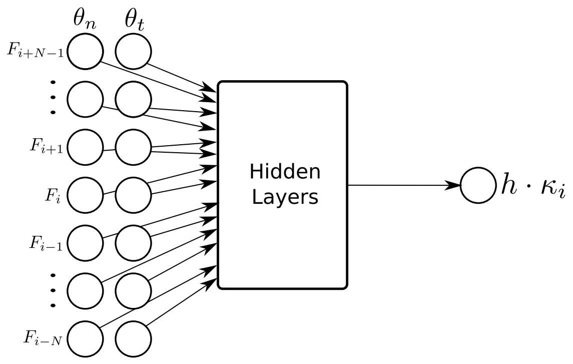


Fig. 2. Neural network used to describe the function C.

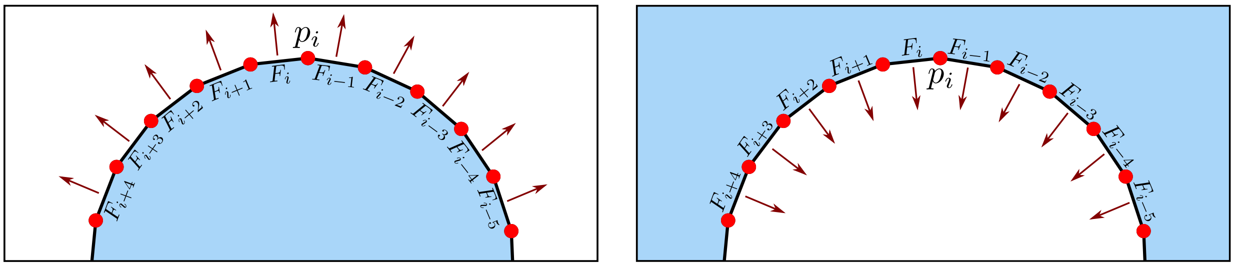

p_i
F_{i+4} F_{i+3} F_{i+2} F_{i+1} F_i F_{i-1} F_{i-2} F_{i-3} F_{i-4} F_{i-5}
p_i
F_{i+4} F_{i+3} F_{i+2} F_{i+1} F_i F_{i-1} F_{i-2} F_{i-3} F_{i-4} F_{i-5}

Fig. 3. Dataset generation for the Front-Tracking method: (Left) normals pointing outside of the circle and (Right) normals pointing into the circle in which negative curvature is assumed.

The training of this network was performed in Python via the Keras-Tensorflow module. The total dataset is split into a training set (80%) and a test set (20%). We did not encounter any significant issues regarding overfitting during our tests, so we chose to simply use a fixed number of epochs instead of using a validation set to measure convergence. After training, we employ the test dataset to evaluate how well the model is predicting curvatures. We note that our model and training set are defined by two hyperparameters: the number of segments around a marker (N) and the length of segments used in the training circles (h). Notice that there is a connection between these parameters since both N and h affect the total length of segments around a marker point p_i. We tested different values for both parameters and we show here results for $N = \{3, 5, 7\}$ and $h = \{0.0004, 0.0001\}$. These results are shown in Fig. 4 as a scatter plot of predicted values versus exact values. If the fit were perfect, all points would lie on the identity line. According to Fig. 4, we can observe better results

Table 1
Mean squared error in the test dataset generated with circles.

	$h = 0.0004$	$h = 0.0001$
$N = 3$	5.03E-04	9.00E-04
$N = 5$	6.41E-04	2.64E-04
$N = 7$	4.30E-03	2.41E-04

Table 2
Mean squared error in the test dataset generated for a sine function.

	$h = 0.0004$	$h = 0.0001$
$N = 3$	2.07E-04	2.50E-03
$N = 5$	6.27E-04	6.05E-04
$N = 7$	1.39E-03	1.70E-03

for $h = 0.0004$ and smaller values of N. It makes sense especially for the smaller circles, in which too many segments would already take a big portion of their perimeter. From a qualitative point-of-view, according to Figs. 4(b) and (e), we can also see an inaccurate fitting for the tuples $(N, h) = (3, 0.0001)$ and $(N, h) = (7,\ 0.0004)$ respectively. In order to further investigate the accuracy of the scheme, we have quantified in Table 1 the relative mean squared error for the same points plotted in Fig. 4. Based on the results described in Table 1, we can note that most combinations give reasonably good results with similar order of magnitude, with the exception of $(N, h) = (3, 0.0001)$ and $(N, h) = (7,\ 0.0004)$.

We can now compute the curvature of other curves using the network that was previously trained only with data from circles. Following the test performed in [19], we have computed the curvature of a sine wave. The wave was discretized on $x \in [-\pi, \pi]$ by 200 line segments, which gives us $h \approx 0.038$. Note that h is not required to be the exact same as the one used in training, since the output curvature was nondimensionalized by h. However, the discretization should be fine enough in order to obtain reasonable results in steep curves. We assume that the normal vectors are pointed upwards from the wave.

In Fig. 5 we show the results obtained by using the model created and trained with $(N, h) = (3,\ 0.0004)$. On the left we plot the analytical and numerical curvatures computed over the x interval, and while slightly worse predictions are seen at the curvature peaks, generally good agreement is observed. On the right, we present the same results as a scatter plot of exact versus numerical curvatures for each point in the sine wave. According to the results described in Fig. 5, we can see a good agreement between the analytical curvature and the predicted one.

In Table 2 we present the relative mean squared error for the sine wave curvature using the same variation of model and training parameters as before. While all variations produce reasonably good results, we note that the lowest error is obtained for $N = 3$ and $h = 0.0004$. For all results shown in the following section, we will assume this combination of parameters. The code used in this Section can be downloaded through the following link: https://github.com/hugo-franca/FrontTrackingCurvature.

3. Application of the scheme for solving a complex free surface flow

Our end goal with this formulation was to use the computed curvature to simulate fluid flows under the effects of surface tension. To test how the machine learning curvature works in these simulations, we decided to incorporate the trained network in our own in-house solver for free surface flows. After the network training is performed using the Python modules, all the network weights are exported into a simple text file, which is loaded into our solver (implemented in language C). The weights can then be easily used to re-evalute the network on our flow interfaces.

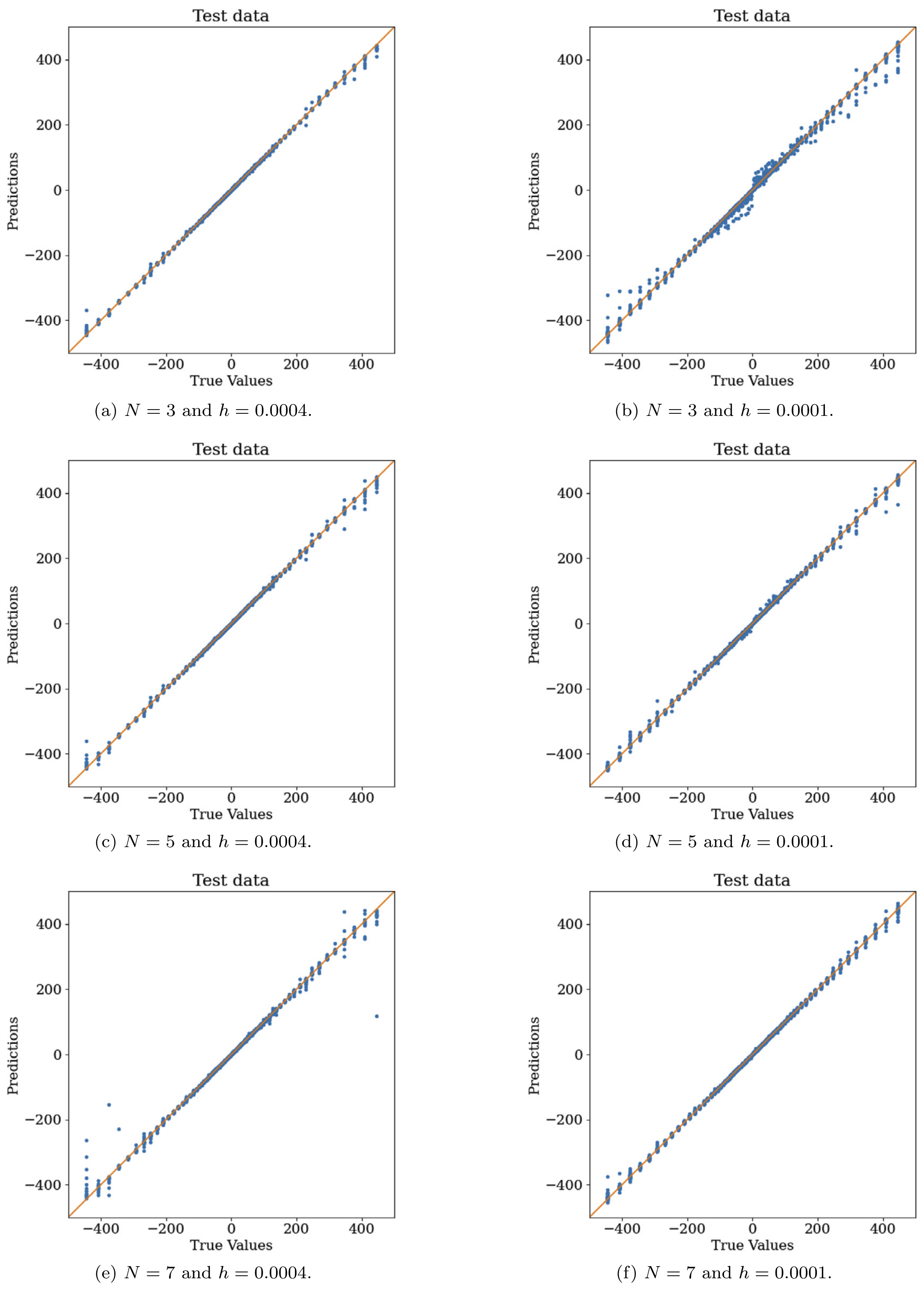
The nondimensional Navier-Stokes equations for a Newtonian fluid are given by

$$\nabla \cdot \mathbf{u} = 0, \quad \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} + \frac{1}{Fr^2}\mathbf{g}. \tag{2}$$

In these equations, the velocity and pressure fields are represented by $\mathbf{u}$ and p, respectively, while $\mathbf{g}$ is the gravitational force. The non-dimensional numbers in the momentum equation are the Reynolds number Re and the Froude number Fr. We use a single-phase formulation, in which the free surface boundary conditions applied at the fluid interface are given by

$$\mathbf{n} \cdot (\boldsymbol{\sigma} \cdot \mathbf{n}) = \frac{1}{We}\kappa, \qquad \mathbf{m} \cdot (\boldsymbol{\sigma} \cdot \mathbf{n}) = 0, \tag{3}$$

where $\boldsymbol{\sigma}$ is the total stress tensor, $\mathbf{n}$ and $\mathbf{m}$ are the unit normal and tangent vectors, respectively, and We is the Weber number.



(a) $N = 3$ and $h = 0.0004$.

(b) $N = 3$ and $h = 0.0001$.

(c) $N = 5$ and $h = 0.0004$.

(d) $N = 5$ and $h = 0.0001$.

(e) $N = 7$ and $h = 0.0004$.

(f) $N = 7$ and $h = 0.0001$.

Fig. 4. Scatter plots of the exact curvature versus the machine learning predicted curvature for each entry in the test dataset (generated from circles). Three values of N and two values of h are used to compare results. In this figure, the curvature values on the axis scaling are given by $\kappa = \pm 1/R$, with R varying from 0.00225 to 0.475.

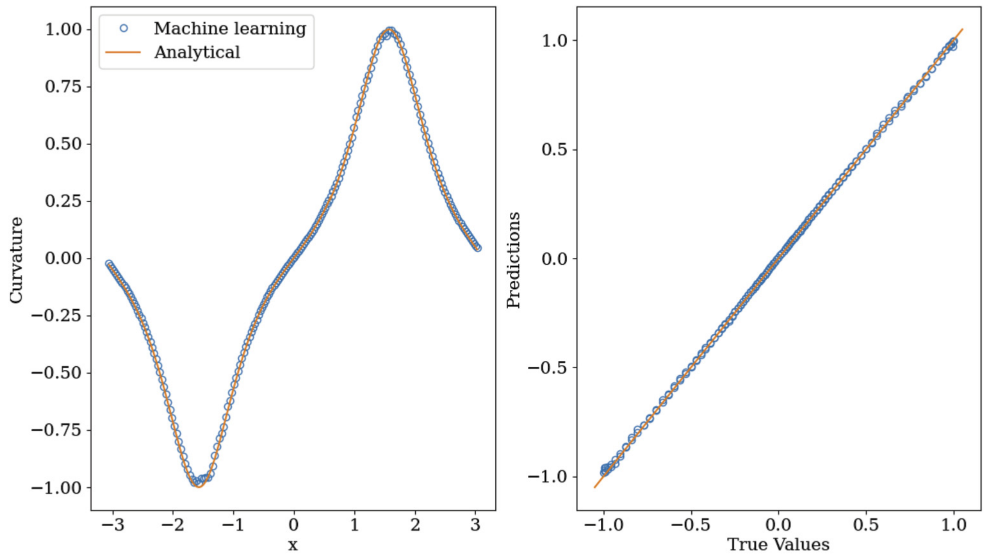


Fig. 5. Results for the sine wave function. Left: plots of the exact and analytical curvature for $x \in [-\pi, \pi]$. Right: Scatter plot of the exact versus machine learning curvature according to the limits of sine wave curvature, e.g. $[-1, 1]$.

Table 3
Fluid and simulation parameters used in the binary droplet collision simulations. These parameters correspond to the 4% HPMC fluid experiments performed in [1].

Nondimensional parameter	Value
Reynolds (Re)	{50.48, 87.45, 112.89, 151.46}
Weber (We)	{10, 30, 50, 70}
Impact parameter (B)	0, 0.2, 0.5, 0.8

Equations (2) are solved numerically through a finite differences scheme based on the MAC formulation [11,13]. The time-stepping is also performed by a simple finite differences Euler scheme, and a projection method is used to decouple the velocity and pressure updates. Finally, the marker particles $\mathbf{x}$ are updated at each time step by solving the advection equation

$$\dot{\mathbf{x}} = \mathbf{u}. \tag{4}$$

The flow chosen here to test the machine learning algorithm is the binary droplet collision [20]. In this flow, two droplets initialized with given velocities $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively, are launched against each other resulting in different outcomes. An important parameter in this simulation, named impact parameter B, is defined as

$$B = \frac{2b}{D_1 + D_2}, \tag{5}$$

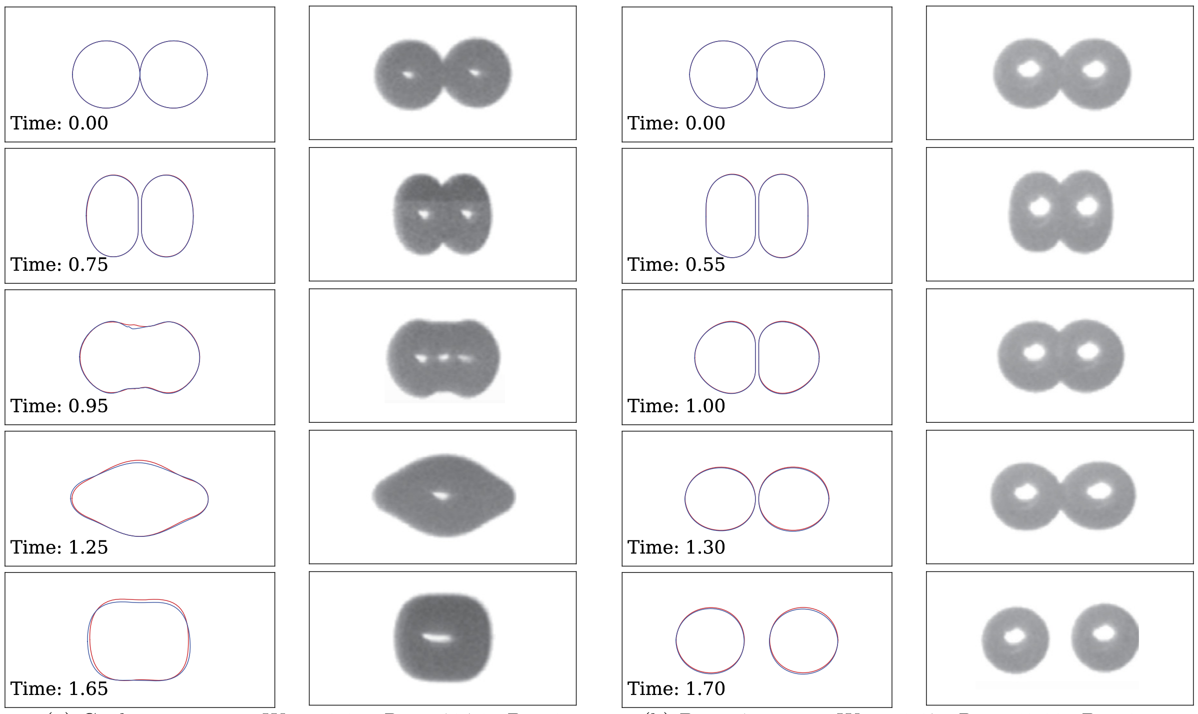
where D_1 and D_2 are the drop diameters and b is the center-to-center relative distance (more details can be seen in [1]).

In the literature there is a lot of interest in studying the post-collision outcomes of these droplets, which can be classified in three main regimes: coalescence, bouncing and separation.

We begin our tests comparing a classical FT and the FT-ML schemes for reproducing two experimental collisions reported in [15]. According to the experiments, in one of these collisions the droplets are expected to coalesce and in the other to bounce. Fig. 6a shows a side-by-side comparison between the numerical and experimental results for the coalescence case, while Fig. 6b presents the same comparison for the bouncing simulation. In both cases we note the machine learning curvature is able to reproduce the experimental results with good qualitative agreement. In addition, in the figures for both numerical simulations we plot the interface obtained using the Machine Learning algorithm (blue) and using a traditional curve approximation scheme (red). We note that machine learning can capture the interface similarly to a traditional approach commonly adopted in FT schemes.

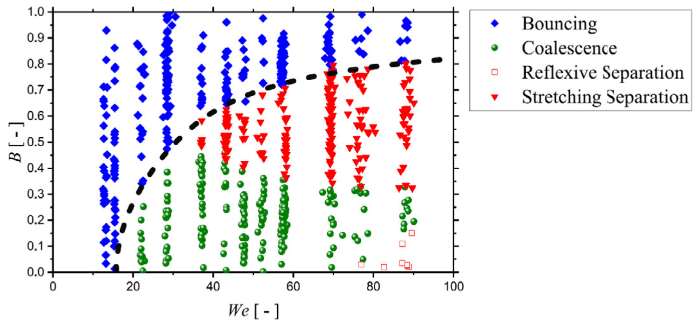
In [1], a regime map is created indicating which outcome happens for different flow parameters, particularly B and We. In order to show our machine learning strategy works for a range of simulation parameters, we reproduced part of one of the maps shown in [1]. Table 3 contains the parameters used in these simulations.

Fig. 7 shows a comparison between the experimental map (taken from [1]) and the numerical map obtained using the machine learning curvature in our solver. We note that the solver is able to correctly capture the expected experimental

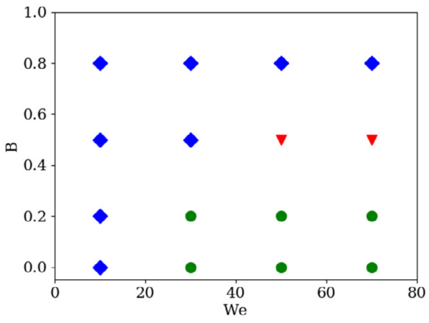


(a) Coalescence case: $We = 2.25$, $Re = 46.79$, $B = 0$. (b) Bouncing case: $We = 2.27$, $Re = 59.32$, $B = 0$.

Fig. 6. Time sequence comparison between numerical and experimental droplet collisions in two cases: coalescence (a) and bouncing (b). In the numerical panels two interfaces are shown: interface obtained from the FT-ML curvature (blue) and from a traditional FT scheme (red). Panels from experimental results are reproduced from [15], with the permission of AIP Publishing. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)



(a) Experimental map reproduced from [1], with the permission of AIP Publishing.



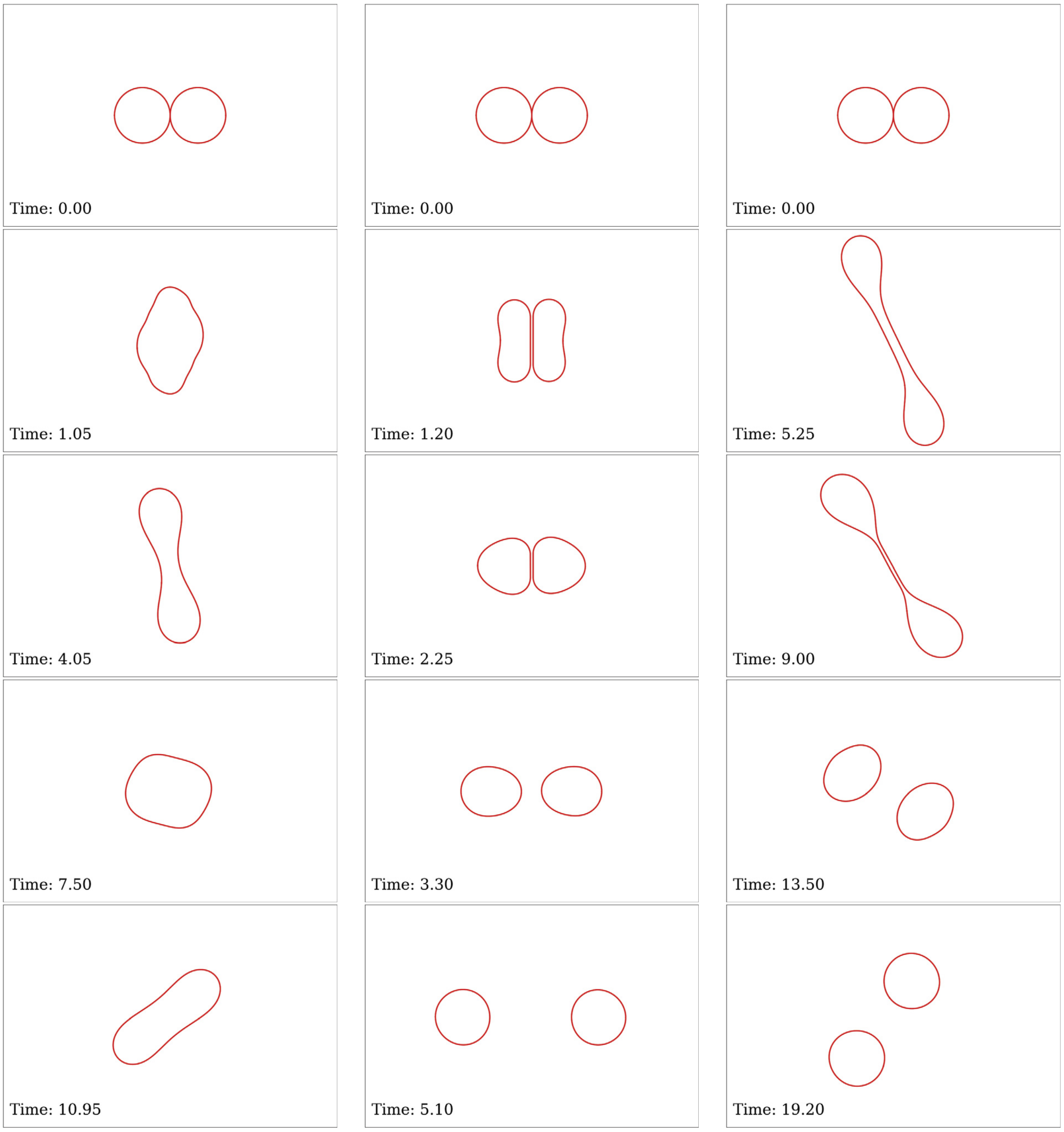
(b) Numerical map obtained from our solver.

Fig. 7. Experimental (a) and numerical (b) regime maps for the Newtonian binary droplet collision.

results. In Fig. 8, we depict selected frames for three different numerical simulations illustrating the possible outcomes: coalescence, separation and bouncing. It is worth to note that the topological changes that merge the droplets together do not happen naturally in an FT representation and need to be enforced. This enforcement was done at a certain simulation time prescribed at the beginning of the simulation, following the strategy proposed in [12].

4. Conclusions

In this work we have shown that it is possible to use machine learning to compute the curvature of Front-Tracking interfaces. Using a very simple model and training strategy based only on circles, we were already capable of creating a function that can even be used to predict curvatures of complex interfaces arising from fluid simulations driven by surface tension. Finally, we highlight the importance of performing the present formulation by training the network with different elementary shapes. In addition, as a future work, we intend to conduct further studies on changing the network structure and including new investigations concerning data preprocessing mechanisms.



(a) Coalescence: $We = 30,\ B = 0.2$ (b) Bouncing: $We = 10,\ B = 0$ (c) Separation: $We = 50,\ B = 0.5$

Fig. 8. Time sequence of three different simulations illustrating the possible outcomes: coalescence, bouncing and separation.

CRediT authorship contribution statement

Hugo L. França: Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Cassio M. Oishi:** Conceptualization, Investigation, Methodology, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank the financial support given by São Paulo Research Foundation (FAPESP) grants #2013/07375-0 and #2019/01811-9, and the National Council for Scientific and Technological Development (CNPq), grant #305383/2019-1.

References

[1] K.H. Al-Dirawi, A.E. Bayly, A new model for the bouncing regime boundary in binary droplet collisions, Phys. Fluids 31 (2) (February 2019) 027105.
[2] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, Y. Li, L. Wu, A simple package for front tracking, J. Comput. Phys. 213 (2) (April 2006) 613–628.
[3] J.D. Evans, H.L. França, C.M. Oishi, Application of the natural stress formulation for solving unsteady viscoelastic contraction flows, J. Comput. Phys. 388 (July 2019) 462–489.
[4] M. Febres, D. Legendre, Enhancement of a 2d front-tracking algorithm with a non-uniform distribution of Lagrangian markers, J. Comput. Phys. 358 (April 2018) 173–200.
[5] J. Glimm, O. McBryan, R. Menikoff, D.H. Sharp, Front tracking applied to Rayleigh–Taylor instability, SIAM J. Sci. Stat. Comput. 7 (1) (January 1986) 230–251.
[6] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1) (January 1981) 201–225.
[7] L.Á. Larios-Cárdenas, F. Gibou, A deep learning approach for the computation of curvature in the level-set method, SIAM J. Sci. Comput. 43 (3) (January 2021) A1754–A1779.
[8] L.Á. Larios-Cárdenas, F. Gibou, A hybrid inference system for improved curvature estimation in the level-set method using machine learning, https://arxiv.org/abs/2104.02951, 2021.
[9] X. Liu, K. Morita, S. Zhang, Machine-learning-based surface tension model for multiphase flow simulation using particle method, Int. J. Numer. Methods Fluids 93 (2) (July 2020) 356–368.
[10] N. Mangiavacchi, A. Castelo, M.F. Tomé, J.A. Cuminato, M.L.B. de Oliveira, S. McKee, An effective implementation of surface tension using the marker and cell method for axisymmetric and planar flows, SIAM J. Sci. Comput. 26 (4) (April 2005) 1340–1368.
[11] F.P. Martins, C.M. Oishi, A.M. Afonso, M.A. Alves, A numerical study of the kernel-conformation transformation for transient viscoelastic fluid flows, J. Comput. Phys. 302 (December 2015) 653–673.
[12] M.R. Nobari, Y.-J. Jan, G. Tryggvason, Head-on collision of drops—a numerical investigation, Phys. Fluids 8 (1) (January 1996) 29–42.
[13] C.M. Oishi, R.L. Thompson, F.P. Martins, Normal and oblique drop impact of yield stress fluids with thixotropic effects, J. Fluid Mech. 876 (August 2019) 642–679.
[14] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, J. Comput. Phys. 79 (1) (November 1988) 12–49.
[15] K.-L. Pan, C.K. Law, B. Zhou, Experimental and mechanistic description of merging and bouncing in head-on binary droplet collision, J. Appl. Phys. 103 (6) (March 2008) 064901.
[16] H.V. Patel, A. Panda, J.A.M. Kuipers, E.A.J.F. Peters, Computing interface curvature from volume fractions: a machine learning approach, Comput. Fluids 193 (October 2019) 104263.
[17] S. Popinet, Numerical models of surface tension, Annu. Rev. Fluid Mech. 50 (1) (January 2018) 49–75.
[18] S. Popinet, S. Zaleski, A front-tracking algorithm for accurate representation of surface tension, Int. J. Numer. Methods Fluids 30 (6) (1999) 775–793.
[19] Y. Qi, J. Lu, R. Scardovelli, S. Zaleski, G. Tryggvason, Computing curvature for volume of fluid methods using machine learning, J. Comput. Phys. 377 (2019) 155–161.
[20] J. Qian, C.K. Law, Regimes of coalescence and separation in droplet collision, J. Fluid Mech. 331 (January 1997) 59–80.
[21] M. Tavares, D.-A. Koffi-Bi, E. Chénier, S. Vincent, A two-dimensional second order conservative front-tracking method with an original marker advection approach based on jump relations, Commun. Comput. Phys. 27 (5) (2020) 1550–1589.
[22] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.-J. Jan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2) (May 2001) 708–759.
[23] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas-Liquid Multiphase Flows, Cambridge University Press, 2011.
[24] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, J. Comput. Phys. 100 (1) (May 1992) 25–37.