Time-Splitting Procedure for the Resolution of the Volume Fraction for Unsteady Flows

Yoann Robert, Alban Leroyer, Jeroen Wackers, Michel Visonneau, Patrick Queutey

Time-Splitting Procedure for the Resolution of the Volume Fraction for Unsteady Flows

Yoann Robert, Alban Leroyer, Jeroen Wackers, Michel Visonneau and Patrick Queutey

LHEEA Lab., École Centrale de Nantes / CNRS-UMR 6598, 1 rue de la Noë, Nantes, France
yoann.robert@ec-nantes.fr

1 Introduction

Numerical simulations of multifluid flows using interface capturing schemes are highly constrained by the Courant-Friedrichs-Lewy (CFL) condition. Even with implicit codes, the Courant number Co is limited to values lower than unity in order to benefit from compressive properties of the schemes and to restrict the diffusion of the interface [6, 9]. Therefore the usable time steps become rapidly redhibitory since they are linked to the grid by the definition of Co and since, if the grid is refined, the time step must be reduced accordingly. This is particularly true when the free surface undergoes large and violent deformations, as in the case of a rowing stroke in water [3, 8] or a prism impacting a free surface [2, 10]. The equation subject to this condition is the convection equation of the volume fraction of water c.

The goal of this work is to elaborate a strategy to accelerate numerical simulations of unsteady multifluid flows while formally respecting the CFL condition. The procedure presented here is an extension of the one developed for flows reaching a steady state [4]. Manzke *et al.* have likewise worked on this topic [5] but only a few details were given for the treatment of unsteady computations. In short our method allows the use of a larger global time step for each temporal iteration and the resolution of the convection equation for c is carried out in several successive steps in which the associated time step is smaller. This compromise enables to diminish the total computation time, a priori without loss of accuracy. This approach should be even more efficient for parallel computations and especially for the fine meshes using adaptive grid refinement (AGR) [10]. In the AGR procedure, a number of layers of refined cells around the free surface, or buffer layers, can be set. This is done to ensure that the free surface stays in the refined grid area, until the next step of grid refinement. Indeed each refinement causes an extra cost in CPU time, that is why it is not performed at each temporal iteration. The combination of the AGR with the time-splitting procedure enables to carry out an indirect form of parallelisation of the temporal resolution: the use of a larger time step leads to increase the number of buffer layers, and thus the number of cells, which forces to employ more cores. Parallel computations have a problem of scaling efficiency, as there is an issue of communication between cores, in other words a maximum number of cells per core. So the use of the time-splitting procedure with a reasonable number of extra cores, keeping the same ratio of cells per core as for a regular computation, should make the number of temporal iterations decrease and will therefore reduce the total CPU time.

In the article, some explanations of the method will be given in section 2 and first results showing benefits will be exposed in section 3. Finally, section 4 will draw some conclusions and perspectives. In the remainder of the document, the term *time-splitting procedure* will be employed to indicate the described method.

2 The time-splitting procedure

2.1 Principle

The time-splitting procedure consists of cutting each original time step into N successive intervals (see figure 1), also called temporal splits or subcycles, and solving c on each of them. Hence a fraction of the original time step is associated with each temporal split and adjusted to have a Courant number satisfying the CFL condition. In this way, the global time step can be increased and the total CPU time will therefore be reduced. The global time step can in principle be set to a critical value, representing the limit not to cross in order to capture the physics with a second-order time discretisation scheme, regardless of the grid fineness. The cost of the extra resolutions for the volume fraction remains quite low because this resolution is not the most time-consuming part in RANS codes.

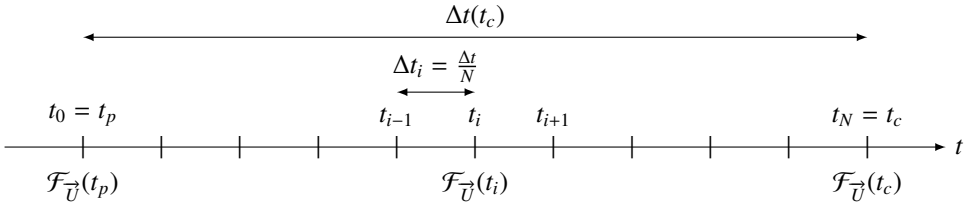


Figure 1: Temporal splitting for the equation (2).

2.2 Time-integration method

The equation to solve for the interface-capturing methods using an ALE approach is the convection equation for the volume fraction c:

$$\frac{\delta}{\delta t}\int_{\mathcal{V}} c \,\mathrm{d}\mathcal{V} + \oint_{\mathcal{S}} c\left(\overrightarrow{U} - \overrightarrow{U_d}\right)\cdot\overrightarrow{n}\,\mathrm{d}\mathcal{S} = 0\,, \tag{1}$$

where $\mathcal{V}$ is the control volume limited by a closed surface $\mathcal{S}$ of normal vector $\overrightarrow{n}$ outwardly directed and moving at a velocity $\overrightarrow{U_d}$. The time derivative following the moving grid is written $\frac{\delta}{\delta t}$.

The finite volume discretisation with the first-order backward differentiation formula leads to the following discretised equation:

$$\frac{c(t_c)\,V(t_c) - c(t_p)\,V(t_p)}{\Delta t} + \sum_{\text{faces } \mathcal{S}_f} c_f(t_c)\left(\mathcal{F}_{\overrightarrow{U}}(t_c) - \mathcal{F}_{\overrightarrow{U_d}}(t_c)\right) = 0\,, \tag{2}$$

where t_p and t_c indicate respectively the previous and current instants, $\Delta t = t_c - t_p$ the current global time step and V the volume of the cells. The reconstruction of the volume fraction at the face centre is noted c_f. The vectors $\mathcal{F}_{\overrightarrow{U}}$ and $\mathcal{F}_{\overrightarrow{U_d}}$ are respectively the flux of velocity and the flux of displacement velocity of the grid through the face $\mathcal{S}_f$. These three quantities are expressed at the current instant t_c. To shorten thereafter the equations, the variable $\mathcal{F}'$ is introduced to gather both fluxes and is defined by $\mathcal{F}' = \mathcal{F}_{\overrightarrow{U}} - \mathcal{F}_{\overrightarrow{U_d}}$. The first-order time discretisation scheme described here is not able to accurately capture unsteady phenomena for reasonable time steps because it is too diffusive. However, for computations reaching a steady state, an accurate temporal resolution is not sought, which means that this scheme can be used. On the other hand, the temporal accuracy is essential for unsteady computations. At the end of the time step, the second-order backward differentiation formula (BDF2) has to be verified:

$$e_c\,c(t_c)\,V(t_c) + e_p\,c(t_p)\,V(t_p) + e_q\,c(t_q)\,V(t_q) + \sum_{\text{faces } \mathcal{S}_f} c_f(t_c)\,\mathcal{F}'(t_c) = 0\,, \tag{3}$$

where e_c, e_p and e_q are the coefficients of the aforementioned scheme for the time derivative and t_q is the instant preceding t_p. This causes new difficulties compared to the steady case explained in [4], especially to reconstruct c_f on the global time step (detailled in section 2.2.2).

2.2.1 Resolution of the cell-centered volume fraction on a subcycle

As previously mentioned, the resolution of (1) is carried out by a succession of temporal splits on intervals $\Delta t_i = \frac{\Delta t}{N}$ (see figure 1). The fluxes $\mathcal{F}_{\overrightarrow{U}}$ and $\mathcal{F}_{\overrightarrow{U_d}}$, as well as the volumes, are interpolated between their values at the instants t_p and t_c, knowing that quantities at t_c are updated at each non linear iteration within the convergence loop. By linear combination, $\mathcal{F}'$ is calculated in the same way:

$$\mathcal{F}'(t_i) = (1-\alpha_i)\,\mathcal{F}'(t_p) + \alpha_i\,\mathcal{F}'(t_c) \quad \text{with} \quad 0 < i \leq N\,, \quad \alpha_i = \frac{i}{N} \quad \text{and} \quad t_i = t_p + \alpha_i\,\Delta t(t_c)\,. \tag{4}$$

The Crank-Nicolson scheme (5) has been chosen to maintain a second-order accuracy with a compact two-point stencil for the time discretisation on each subcycle. The discretised equation of convection for the volume fraction within each temporal split is then written:

$$\frac{c(t_i)\,V(t_i) - c(t_{i-1})\,V(t_{i-1})}{\Delta t(t_i)} + \frac{1}{2}\sum_{\text{faces } \mathcal{S}_f} c_f(t_i)\,\mathcal{F}'(t_i) = -\frac{1}{2}\sum_{\text{faces } \mathcal{S}_f} c_f(t_{i-1})\,\mathcal{F}'(t_{i-1})\,, \tag{5}$$

where the right-hand side is the source term.

The value of c at t_c is directly equal to its value at the last subcycle.

2.2.2 Reconstruction of the interfacial volume fraction at the global level

The interfacial volume fraction c_f, expressed at the time t_c, is required to compute the term $\rho\,\mathcal{F}'$ of the momentum equation. It has to be reconstructed while verifying the second-order time discretisation of (3). To do so, it is mandatory to take into account the evolution of the cell-centered value of c during the subcycles.

The sum of the N equations (5) between t_p and t_c leads to:

$$\frac{c(t_c)\,V(t_c) - c(t_p)\,V(t_p)}{\Delta t(t_c)} + \sum_{\text{faces } \mathcal{S}_f} \underbrace{\frac{1}{2\,N(t_c)}\left(\sum_{i=1}^{N(t_c)} c_f(t_i)\,\mathcal{F}'(t_i) + \sum_{i=1}^{N(t_c)} c_f(t_{i-1})\,\mathcal{F}'(t_{i-1})\right)}_{=B_{t_p\to t_c}} = 0\,. \tag{6}$$

This first step enables to have an equation where the time-derivative part is identical to (2), hence only a first order. The term $B_{t_p\to t_c}$ can be seen as an interfacial quantity equal to the average of $c_f\,\mathcal{F}'$ over all the temporal splits between t_p and t_q. The same sum between t_q and t_p gives:

$$\frac{c(t_p)\,V(t_p) - c(t_q)\,V(t_q)}{\Delta t(t_p)} + \sum_{\text{faces } \mathcal{S}_f} \underbrace{\frac{1}{2\,N(t_p)}\left(\sum_{i=1}^{N(t_p)} c_f(t_i)\,\mathcal{F}'(t_i) + \sum_{i=1}^{N(t_p)} c_f(t_{i-1})\,\mathcal{F}'(t_{i-1})\right)}_{=B_{t_q\to t_p}} = 0\,. \tag{7}$$

From the equations (6) and (7), it is possible to retrieve a BDF2 for (1). The second-order time discretisation of the derivative of a quantity ϕ at the time t_c, in function of t_c, t_p and t_q is:

$$\left(\frac{\delta\phi}{\delta t}\right)_{t=t_c} = e_c\,\phi_c + e_p\,\phi_p + e_q\,\phi_q\,, \tag{8}$$

with

$$e_c = \frac{2\,\Delta t(t_c) + \Delta t(t_p)}{\Delta t(t_c)\left[\Delta t(t_c) + \Delta t(t_p)\right]}\,,\quad e_p = -\frac{\Delta t(t_c) + \Delta t(t_p)}{\Delta t(t_c)\,\Delta t(t_p)} \quad\text{and}\quad e_q = \frac{\Delta t(t_c)}{\Delta t(t_p)\left[\Delta t(t_c) + \Delta t(t_p)\right]}\,. \tag{9}$$

Applying the linear combination

$$(11) \leftarrow (6)\cdot e_c\cdot\Delta t(t_c) - (7)\cdot e_q\cdot\Delta t(t_p)\,, \tag{10}$$

leads to, after simplifications, an equation with a second-order time derivative:

$$e_c\,c(t_c)\,V(t_c) + e_p\,c(t_p)\,V(t_p) + e_q\,c(t_q)\,V(t_q) + \underbrace{\sum_{\text{faces } \mathcal{S}_f}\left[e_c\,\Delta t(t_c)\,B_{t_p\to t_c} - e_q\,\Delta t(t_p)\,B_{t_q\to t_p}\right]}_{F} = 0\,. \tag{11}$$

The term F contains $B_{t_q\to t_p}$ and $B_{t_p\to t_c}$, respectively computed in the subcycles of the intervals $[t_q, t_p]$ and $[t_p, t_c]$. To fulfill (3), F has to be written under the form (12) where $\tilde{c}_f$ stands for the reconstructed value of c_f.

$$F = \sum_{\text{faces } \mathcal{S}_f} \tilde{c}_f\,\mathcal{F}'(t_c)\,. \tag{12}$$

This gives the following arbitrary expression for $\tilde{c}_f$:

$$\tilde{c}_f = \frac{1}{\mathcal{F}'(t_c)}\left[\Delta t(t_c)\,e_c\,B_{t_p\to t_c} - \Delta t(t_p)\,e_q\,B_{t_q\to t_p}\right] \tag{13}$$

where an obvious problem arises in the case $\mathcal{F}' = 0$. Until now, the division by $\mathcal{F}' + \epsilon$ (with ϵ equal to 10 times the machine precision), proved to be a functional workaround.

3 Results for the dam break with obstacle

The time-splitting procedure has been implemented in the *ISIS-CFD* code [7], developed by the DSPM (Dynamics of Marine Propulsion Systems) team of the LHEEA Lab. and available as part of the FINE™/Marine computing suite. Then it has been validated on several bidimensional test cases, for flows reaching a steady state and purely unsteady flows: simple advection of a square, submerged foil (as described in [1, 4]), dam break with an obstacle or propagation of waves. Solutions are very close (or even identical) to the ones obtained without the time-splitting procedure. In these first computations, the expected reduction of the total CPU time is confirmed: a reduction factor up to 4 has been reached. In this article, the results related to the dam break with obstacle are presented.

3.1 Test case description

One of the most interesting examples used for the validation of the simulation of free surface flows is the dam break with obstacle [9]. In the initial state (see figure 2), a rectangular water column (146 cm wide, 292 cm high) is located in the lower left corner of a square domain (side of 584 cm) whose edges are solid walls except for the top one which is in open air. The gravity makes the water column fall, the latter hits a rectangular obstacle (24 cm wide, 48 cm high).

3.2 Numerical procedure

For these simulations, a slip condition is applied to all solid walls. Since the case is two-dimensional, a mirror condition is used for the front and back faces. A Dirichlet condition is applied to the upper face: the hydrostatic pressure is imposed.

The mesh generation is performed with HEXPRESS™. Two kinds of grid have been used. In the first configuration, the grid is fixed and quite fine. In the lower half of the domain, there are 192 cells in the horizontal direction and 96 in the vertical direction. The grid becomes coarser in the upper part. The dimensions of the cells gradually get larger as the vertical coordinate increases and reach values they would have if the domain would be entirely discretised in 24 per 24 cells. In the second configuration, the AGR is used. The initial mesh is quite coarse since the domain contains only 58 cells in each direction, except in a more refined area around the obstacle. The cells around the free surface are refined during the computation every three time steps, with a target size of 1,25 mm (normal distance to the free surface) and with a given number of buffer layers of cells N_l.

The computations have been done on the same computer and with only one core. Cases with or without time-splitting procedure have been launched and different numbers N of temporal splits (as defined in section 2) have been tested. If $N = 1$, it means that the time-splitting procedure has not been used. Increasing N should make the CPU time t_{CPU} decrease although it has to be verified that the results do not degrade. For instance, the maximum value of the horizontal component of the force on the obstacle Fx_{max} is examined. Since any experimental value are available, the value of the case 4 (see table 1) is arbitrarily taken as reference. When the time-splitting procedure is used, it is necessary to increase N_l to have a sufficient safety margin. Thus the mean number of cells $N_{c,\,mean}$ is higher.

The global time step is adapted at each temporal iteration in order to have a global Courant number Co lower than a limit value. For the computations not using the time-splitting procedure, this limit is equal to 0.3 and this value is multiplied by N in the other cases.

3.3 Results

The results of six different cases are summed up in the table 1, all reaching a physical time of simulation of 0.2 s. The evolution of the simulated flow, together with the grid, between the initial instant and $t = 0.6\,s$ is represented in figure 3. The images come from the case 5, somewhat modified to avoid at best undesirable effects relative to the AGR, i.e. unnecessary and costly generations of cells after $t = 0.2\,s$ (images 3c to 3f) and to prevent the water from hitting the top boundary. So the action of the AGR is limited to a acceptable vertical coordinate and the domain is enlarged in the vertical direction. The evolution of the solution as described in [9] is reproduced. A fine layer is coming from the water column and impacts the obstacle. A tongue of water is created at the top of the obstacle, then it is directed to the right and hits the opposite wall of the domain, trapping air below itself. A secondary tongue is then created at the top right of the obstacle, imprisoning in its turn an area of air.

3.3.1 Computations with fixed grids

The first three cases use the fixed-grid configuration. As seen in figure 4, the contour lines of c at $t = 0.2\,s$, with or without using the time-splitting procedure, are very similar. The main difference is that a bigger drop is breaking away from the tongue of water in the case 3. The use of the time-splitting procedure enables to decrease t_{CPU}: a factor up to 3 has been reached in the case where $N = 10$. The cost C_{tot} of each computation can also be observed. It shows the mean total CPU time per cell. Since the number of cells does not vary for this grid, C_{tot} has the same ratio than t_{CPU}. However, Fx_{max} is slightly reduced. The global time step may have been chosen too large relatively to the physics of the flow. The use of time steps too large also has the tendency to trap air at the impact on the obstacle, which consequently reduces the force.

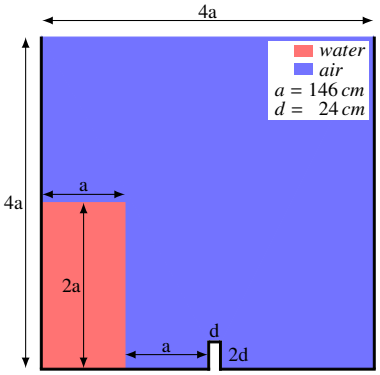
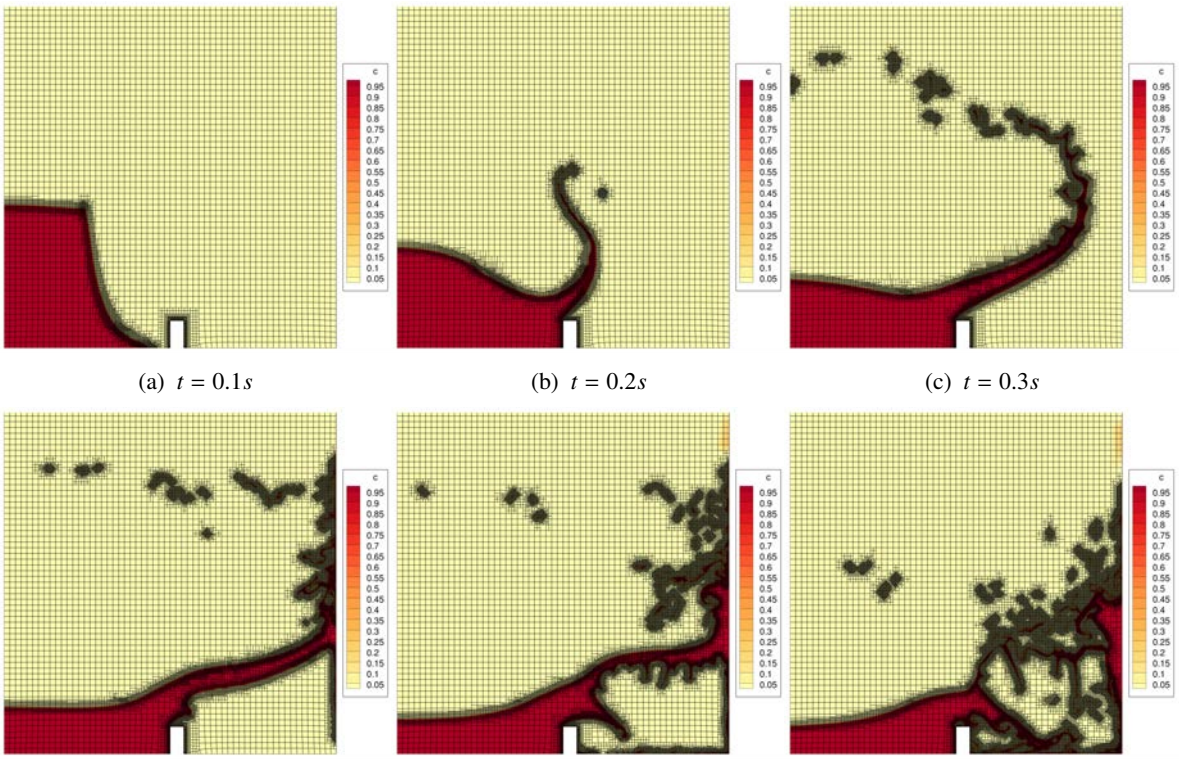


Figure 2: Initial configuration of the dam break with obstacle test case.

Case	N	AGR	N_l	t_{CPU} (s)	ratio t_{CPU}	$N_{c,\,mean}$	ratio $N_{c,\,mean}$	C_{tot}(ms/pt)	ratio C_{tot}	Fx_{max} (N)	Error Fx_{max} (%)
1	1	×	×	1905	-	20210	-	94.3	-	183	0.5
2	5	×	×	824	2.31	20210	1	40.8	2.31	179	2.7
3	10	×	×	646	2.95	20210	1	32.0	2.95	175	4.9
4	1	✓	1	3514	-	5774	-	540.9	-	184	-
5	10	✓	5	2545	1.38	8039	1.39	280.1	1.93	178	3.3
6	10	✓	10	3285	1.07	10590	1.83	263.9	2.05	179	2.7

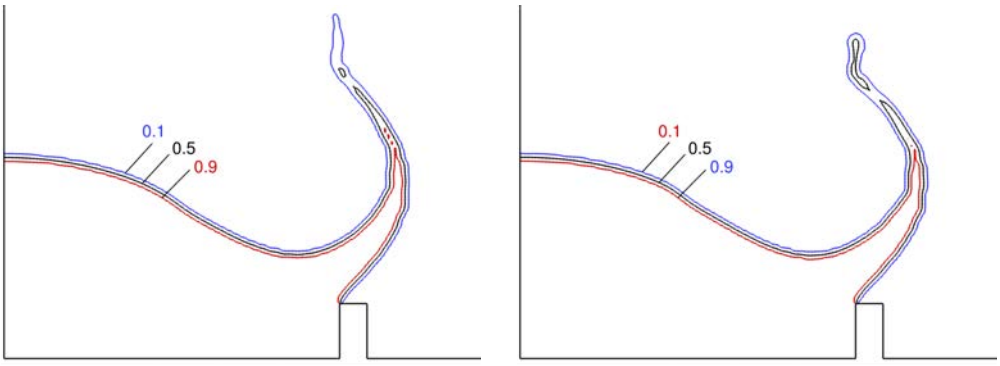
Table 1: Summary of the results obtained for a physical time of simulation of 0.2 s.



(a) $t = 0.1s$ (b) $t = 0.2s$ (c) $t = 0.3s$

(d) $t = 0.4s$ (e) $t = 0.5s$ (f) $t = 0.6s$

Figure 3: Evolution of the simulated flow for the case 5. Contours of the volume fraction on the refined grid.



(a) $N = 1$, case 1 (b) $N = 10$, case 3

Figure 4: Contour lines of the volume fraction at $t = 0.2\,s$ with or without the time-splitting procedure.

3.3.2 Computations with adaptive grid refinement

For cases with AGR, the gains in CPU time are much weaker. This is explained by a higher $N_{c,\,mean}$ (up to 1.8 times higher for the case 6). However, for the same case, C_{tot} is 2 times smaller than the case 4, which does not use the time-splitting procedure. Given the target size specified for the AGR, which is half the size of the cells in the fixed mesh, the accuracy of Fx_{max} is increased. Even with $N_{c,\,mean}$ lower than with fixed grids, the CPU times are higher. This lack of gain is explained by the small target cell size and the law for the adaptation of the global time step to a target Courant number. The combination of these two choices causes the limitation of the usable time step to small values.

4 Conclusion and perspectives

The first results are encouraging and confirm the expectations of the time-splitting procedure. On fixed grids, the CPU time for the simulation of the dam break with obstacle test case has been reduced with a factor 3. In the case of computations using a single core and the adaptive grid refinement, the gain in CPU time is not very large. This is explained by the need to keep large safety margin of layers of cells around the free surface. However, the cost of a computation, while looking at the mean total time per cell, is halved. Thus it appears that the time-splitting procedure should be very efficient on parallel computations, even if it remains to be proved.

Acknowledgements

This study was supported by a grant from the Région Pays de la Loire (project ANOPACy). The authors gratefully acknowledge GENCI (Grand Equipement National de Calcul Intensif, Grant 2014-21308) for the HPC resources.

References

[1] J. H. Duncan. The breaking and non-breaking wave resistance of a two-dimensional hydrofoil. *Journal of Fluid Mechanics*, 126(-1):507–520, 1983.

[2] A. Hay, A. Leroyer, and M. Visonneau. H-adaptive navier–stokes simulations of free-surface flows around moving bodies. *Journal of Marine Science and Technology*, 11(1):1–18, 2006.

[3] A. Leroyer, S. Barré, J.-M. Kobus, and M. Visonneau. Influence of free surface, unsteadiness and viscous effects on oar blade hydrodynamic loads. *Journal of Sports Sciences*, 28(12):1287–1298, 2010.

[4] A. Leroyer, J. Wackers, P. Queutey, and E. Guilmineau. Numerical strategies to speed up CFD computations with free surface - application to the dynamic equilibrium of hulls. *Ocean Engineering*, 38(17–18):2070–2076, 2011.

[5] M. Manzke, J.-P. Voss, and T. Rung. Sub-cycling strategies for maritime two-phase flow simulations. In R. Ansorge, H. Bijl, A. Meister, and T. Sonar, editors, *Recent Developments in the Numerics of Nonlinear Hyperbolic Conservation Laws*, volume 120 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 237–251. Springer Berlin Heidelberg, 2013.

[6] F. Moukalled and M. Darwish. Transient schemes for capturing interfaces of free–surface flows. *Numerical Heat Transfer, Part B: Fundamentals*, 61(3):171–203, 2012.

[7] P. Queutey and M. Visonneau. An interface capturing method for free–surface hydrodynamic flows. *Computers & fluids*, 36(9):1481–1510, 2007.

[8] Y. Robert, A. Leroyer, S. Barré, F. Rongère, P. Queutey, and M. Visonneau. Fluid mechanics in rowing : the case of the flow around the blades. In *The 2014 conference of the International Sports Engineering Association*, 2014.

[9] O. Ubbink. *Numerical prediction of two fluid systems with sharp interfaces*. PhD thesis, Imperial College, University of London, 1997.

[10] J. Wackers, G. Deng, A. Leroyer, P. Queutey, and M. Visonneau. Adaptive grid refinement for hydrodynamic flows. *Computers & Fluids*, 55:85–100, 2012.