UPPSALA
UNIVERSITET

Efficiency and control of split-step methods for complex fluid problems

J. C. Araujo-Cabarcas

Supervisor: Stefan Engblom

Project in Computational Science: Report

February 2012

PROJECT REPORT

Institutionen för informationsteknologi

Efficiency and control of split-step methods for complex fluid problems

J. C. Araujo-Cabarcas

Supervisor: Stefan Engblom
MSc Computational Science, Uppsala University

February 8, 2012

Abstract

This report provides alternative ways to adress nonlinear *partial differential equations* (PDEs) in gradient flow form, particularly a model of surfactant in diffusive interface flow. The 2D problem is first discretized in space with a standard Finite Element procedure and the resulting system of nonlinear ODEs is studied by the use of different numerical schemes for the time discretization. Two split-step-methods are proposed and their respective performance is compared to standard solution methods. The outcome is a gain in performance for low accuracies. This improvement becomes more notable when dealing with large systems of ordinary differential equations (ODEs) result from discretizations on finer space grids. Varying two initial configurations of water and air, the model is tested and the performance of the numerical solution is studied under different initial concentrations of surfactant.

1 Introduction

Surfactants (surface-active agents) are substances that when used in low concentrations, affect the surface characteristics of a two-phase system, for example, liquid-liquid. A surfactant presents a *boundary-tracing* behavior, which means that it tends to trace the juncture of the two phases. Altering in this way the properties of the interface, increasing the contact of materials. For instance, lowering the surface tension of the boundary between fluids.

Soaps, foaming agents, emulsifiers, dispersants and detergents are common examples of surfactants and the study of the surface rheology of surfactant layers, is widely appreciated in fields including detergency, foaming, water repellance (waterproofing), wetting, lubrication, emulsification (coating processes) and plays also roles in the human body, namely in the lung, contributing to its correct functioning.

This report contains a short presentation of modelling surfactant in diffusive interface flow given in Section 1, starting from a brief presentation of the *two-phase flow* problem followed by the *Cahn Hilliard* equation. The inclusion of surfactant is shown and a couple of new chemical potentials presented. After the model is set, a finite element (FEM) discretization is carefully performed and an iterative scheme to solve the resulting set of nonlinear ODEs is suggested in Section 2. Hints on how to track the good behavior of solutions are also given in this section. The main subject in this report, the split-step methods, is introduced in Section 3. Three split-step methods are presented as the time discretization in the solution algorithm and some properties and stability arguments are mentioned as well. Section 4 describes the implementation and the tools utilized in the project. Here, some particular features of the code are explained in detail. The numerical experiments are presented in Section 5 along with the methodology in use. The initial conditions are described and specific arrangements for the solver parameters are explained. In Section 6 the outcome of the work done in the project is discussed and some of the pros and cons of using split-step methods highlighted compared to standard time-discretizations. After this, the smoothness and well-behavior of the solution are shortly presented and discussed. In the final section some general conclusions and future improvements are given.

2 Modelling surfactant in diffusive interface flow

The model starts from the non dimensionalized Cahn-Hilliard equation (1), that is well known for modelling problems in areas such as medicine, metallurgy, new-material development and complex problems in multy-phase fluid dynamics. This equation describes the phase separation dynamics in a binary fluid through the use of the *phase-field* variable $\phi \in$ [-1,1]. Each extreme in ϕ represents a full phase, namely water or oil, and 0 delimits the interface between the fluids. It has to be mentioned that the model describes the dynamics of two incompressible and inmiscible fluids.

The non dimensionalized Cahn-Hilliard equation has the following form

$$\frac{\partial \phi}{\partial t} = \frac{1}{\mathrm{Pe}_\phi} \nabla^2 \left(-\phi + \phi^3 - \frac{\mathrm{Cn}^2}{2} \nabla^2 \phi \right). \tag{1}$$

The Cahn parameter (Cn) represents the thickness of the transition region of the phases in the fluid, and the Peclet number (Pe) expresses the ratio between advection and diffusion of the property observed in a non-dimensional frame.

One of the most important features of equation (1) is the conservation of mass, that can be seen in systems with the form

$$\frac{\partial \phi}{\partial t} = \nabla \cdot \mathbf{J}. \tag{2}$$

The inclusion of the *surfactant volume fraction* $\psi \in [0,1]$ in the Cahn-Hilliard equation is presented as further consideration within the model. The effect of Surfactants in the current study is analyzed in very small quantities, such that its inclussion in the model do not modify the final volume.

The system of equations (3) to be approximated here are deduced under the Landau-Ginzburg free-energy framework of ideas and takes into account the presence of surfactant as an agent modifying[1] the Canh-Hilliard behavior of equation (1). The proposed system is

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= \tfrac{1}{\mathrm{Pe}_\phi}\nabla^2 \mu_\phi \,, \\ \frac{\partial \psi}{\partial t} &= \tfrac{1}{\mathrm{Pe}_\psi}\nabla \cdot [\psi(1-\psi)\nabla \mu_\psi] \,, \end{aligned} \tag{3}$$

where μ_ϕ, μ_ψ are the chemical potentials, namely,

$$\begin{aligned} \mu_\phi &= -\tfrac{\mathrm{Cn}^2}{2}\nabla^2\phi - \phi + \phi^3 + (1-\phi^2)\psi\phi + \tfrac{1}{2\mathrm{Ex}}\psi\phi \,, \\ \mu_\psi &= \mathrm{Pi}\log\frac{\psi}{1-\psi} - \frac{(1-\phi^2)^2}{4} + \tfrac{1}{4\mathrm{Ex}}\phi^2 \,. \end{aligned} \tag{4}$$

Note in this case the distinction of Peclet numbers for ϕ and ψ and the diffusive-like constant Pi describing the diffusive rate of ψ. Clearly, the system (3) belongs to the family of stiff PDEs.

Even though it is hard to set a closed definition of *Stiffness* in ODEs, a common attempt is to classify those ODE with terms that account for different time scales within a solution, requiring extra analysis to set a correct time step for the solver. This is the case presented in equations (6) showed below. There, two important time scales lie within the terms ∇^4 and ∇^2 when the PDE is discretized in space by a standar numerical scheme, for instance FEM. Stiffness means in short, that very small time-steps need to be resolved for, and the use of explicit solvers would not be a suitable strategy for such problems.

[1]For a comprehensive derivation, see [En].

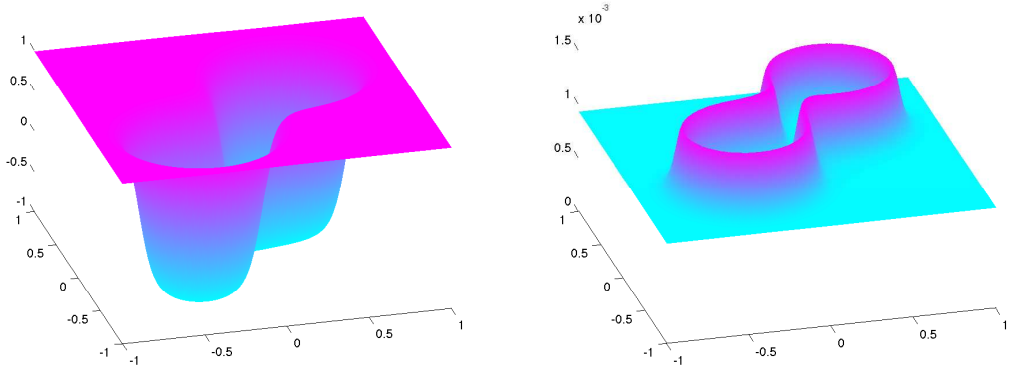

Figure 1: *Plot for a particular solution in the Coalescing bubbles experiment with 20000 triangles showing a typical case for the phase field ϕ (left) and the surfactant volume fraction variable ψ (right).*

Notice the convenient use of the log function when computing μ_ψ by taking advantage of the property

$$\psi(1-\psi)\nabla \log \frac{\psi}{1-\psi} = \nabla\psi\,. \tag{5}$$

From this, it is possible to rename the variables of interest in a more compact form

$$\begin{aligned}
&\Phi = -\tfrac{\mathrm{Cn}^2}{2}\nabla^2\phi - \phi + \phi^3 + (1-\phi^2)\psi\phi + \tfrac{1}{2\mathrm{Ex}}\psi\phi\,,\\
&\frac{\partial\phi}{\partial t} = \tfrac{1}{\mathrm{Pe}_\phi}\nabla^2\Phi\,,\\
&\Psi = -\tfrac{1}{4}(1-\phi^2)^2 + \tfrac{1}{4\mathrm{Ex}}\phi^2\,,\\
&\frac{\partial\psi}{\partial t} = \tfrac{1}{\mathrm{Pe}_\psi}\left(\mathrm{Pi}\nabla^2\psi + \nabla\cdot[\psi(1-\psi)]\nabla\Psi\right)\,.
\end{aligned} \tag{6}$$

The resulting equations are part of the family of *Gradient Systems* that satisfy the criteria given in [Ey]. This type of systems allow the splitting of the energy in *contractive* and *expansive* parts. It is possible to simplify considerably the numerical scheme by accounting for different solution methodologies for contractive or expansive terms. In the particular case treated here, a gradient-stable algorithm is guaranteed under certain discretization conditions. Some of these advantages are explained in detail in [Le]. Later in this report it is shown that for low tolerances, typically up to 10^{-4}, great performance can be obtained thanks to the fact that the equations (6) show a system in gradient form.

3 FEM discretization in space

The first step towards a finite element discretization in space, is to rewrite the model system of equations in variational form. To this end, the system (6) is multiplied by an arbitrary test function χ and then integrated over the entire problem domain (Ω).

$$\begin{aligned}
&\int_\Omega \chi\Phi\, d^3x = -\tfrac{\mathrm{Cn}^2}{2}\int_\Omega \chi\nabla^2\phi\, d^3x + \int_\Omega \chi\left(-\phi+\phi^3+(1-\phi^2)\psi\phi + \tfrac{1}{2\mathrm{Ex}}\psi\phi\right) d^3x\,,\\
&\int_\Omega \chi\frac{\partial\phi}{\partial t}\, d^3x = \tfrac{1}{\mathrm{Pe}_\phi}\int_\Omega \chi\nabla^2\Phi\, d^3x\,,\\
&\int_\Omega \chi\Psi\, d^3x = \int_\Omega \chi\left(-\frac{(1-\phi^2)^2}{4} + \tfrac{1}{4\mathrm{Ex}}\phi^2\right) d^3x\,,\\
&\int_\Omega \chi\frac{\partial\psi}{\partial t}\, d^3x = \tfrac{1}{\mathrm{Pe}_\psi}\int_\Omega \chi\nabla\cdot(\mathrm{Pi}\nabla\psi + \psi(1-\psi)\nabla\Psi)\, d^3x\,.
\end{aligned} \tag{7}$$

Using the Green's first identity and the Divergence theorem

$$\begin{aligned} \nabla \cdot (a\nabla b) &= \nabla a \cdot \nabla b + a\nabla^2 b\,, \\ \int_\Omega \nabla \cdot \mathbf{F}\, d^3x &= \int_{\partial\Omega} \mathbf{F} \cdot \mathbf{n}\, d^2x\,, \end{aligned} \tag{8}$$

and applying homogeneous Neumann boundary conditions for those terms involving the Laplacean operator in the first and second equations in (7), follows

$$\begin{aligned} \int_\Omega \chi\Phi\, d^3x &= \tfrac{\mathrm{Cn}^2}{2} \int_\Omega \nabla\chi \cdot \nabla\phi\, d^3x + \int_\Omega \chi \left(-\phi + \phi^3 + (1-\phi^2)\psi\phi + \tfrac{1}{2\mathrm{Ex}}\psi\phi\right) d^3x\,, \\ \int_\Omega \chi \frac{\partial\phi}{\partial t}\, d^3x &= -\tfrac{1}{\mathrm{Pe}_\phi} \int_\Omega \nabla\chi \cdot \nabla\Phi\, d^3x\,. \end{aligned} \tag{9}$$

Similarly as before, using the vector identity $\nabla \cdot (\chi\mathbf{A}) = \nabla\chi \cdot \mathbf{A} + \chi\nabla \cdot \mathbf{A}$, the Divergence theorem

$$\int_\Omega \chi\nabla \cdot \mathbf{A}\, d^3x = -\int_\Omega \nabla\chi \cdot \mathbf{A}\, d^3x + \int_{\partial\Omega} \chi\mathbf{A} \cdot \mathbf{n}\, d^2x\,, \tag{10}$$

and finally using the boundary condition for the 4th equation in (7),

$$\tfrac{1}{\mathrm{Pe}_\psi}\left(\mathrm{Pi}\nabla\psi + \psi(1-\psi)\nabla\Psi\right) = 0\,. \tag{11}$$

The weak form then reads

$$\begin{aligned} \int_\Omega \chi\Phi\, d^3x &= \tfrac{\mathrm{Cn}^2}{2} \int_\Omega \nabla\chi \cdot \nabla\phi\, d^3x + \int_\Omega \chi \left(-\phi + \phi^3 + (1-\phi^2)\psi\phi + \tfrac{1}{2\mathrm{Ex}}\psi\phi\right) d^3x\,, \\ \int_\Omega \chi \frac{\partial\phi}{\partial t}\, d^3x &= -\tfrac{1}{\mathrm{Pe}_\phi} \int_\Omega \nabla \cdot \chi\nabla\Phi\, d^3x\,, \\ \int_\Omega \chi\Psi\, d^3x &= \int_\Omega \chi \left(-\frac{(1-\phi^2)^2}{4} + \tfrac{1}{4\mathrm{Ex}}\phi^2\right) d^3x\,, \\ \int_\Omega \chi \frac{\partial\psi}{\partial t}\, d^3x &= -\tfrac{\mathrm{Pi}}{\mathrm{Pe}_\psi} \int_\Omega \nabla\chi \cdot \nabla\psi\, d^3x - \tfrac{1}{\mathrm{Pe}_\psi} \int_\Omega \nabla\chi \cdot \psi(1-\psi)\nabla\Psi\, d^3x\,. \end{aligned} \tag{12}$$

3.1 Galerkin method and spatial discretization

Here, take the ansatz

$$\phi(x,y,t) = \sum_j \phi_j(t) N_j(x,y); \quad \chi = N_i \tag{13}$$

and similary for Φ, ψ and Ψ. Then inserting this ansatz into the weak form (12) leads to

$$
\begin{aligned}
\mathbb{M}\boldsymbol{\Phi} &= \tfrac{\mathrm{Cn}^2}{2}\mathbb{K}\boldsymbol{\phi} + \mathbf{f}(\phi,\psi)\,, \\
\mathbb{M}\frac{\partial \boldsymbol{\phi}}{\partial t} &= -\tfrac{1}{\mathrm{Pe}_\phi}\mathbb{K}\boldsymbol{\Phi}\,, \\
\mathbb{M}\boldsymbol{\Psi} &= \mathbf{g}(\phi,\psi)\,, \\
\mathbb{M}\frac{\partial \boldsymbol{\psi}}{\partial t} &= -\tfrac{1}{\mathrm{Pe}_\psi}\left(\mathrm{Pi}\,\mathbb{K}\boldsymbol{\psi} + \mathbb{K}'\boldsymbol{\Psi}\right)\,,
\end{aligned}
\tag{14}
$$

with the standard definitions for the mass, stiffness and other nonlinear entities

$$
\begin{aligned}
M_{ij} &= \int_\Omega N_i N_j\, d^3x\,, \quad K_{ij} = \int_\Omega \nabla N_i \cdot \nabla N_j\, d^3x\,, \\
f_i &= \int_\Omega N_i \left(-\phi + \phi^3 + (1-\phi^2)\psi\phi + \tfrac{1}{2\mathrm{Ex}}\psi\phi\right) d^3x\,, \\
g_i &= \int_\Omega N_i \left(-\frac{(1-\phi^2)^2}{4} + \tfrac{1}{4\mathrm{Ex}}\phi^2\right) d^3x\,, \\
K'_{ij} &= \tfrac{1}{\mathrm{Pe}_\psi}\int_\Omega \nabla N_i \cdot \psi(1-\psi)\nabla N_j\, d^3x\,.
\end{aligned}
\tag{15}
$$

The variables $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are refferred to as the chemical potentials and the reason for having introduced them as variables is due to the fact that the *basis functions* used in linear FEM are only continuous, but not continuously differentiable and there is the need to reduce the order of the PDE. Another reason is to avoid solving for $\boldsymbol{\Phi}$ by using direct methods ($\mathbb{M}^{-1}$), these lead to dense matrix problems that would be unnecessarily expensive. Instead, it is possible as a first approximation to introduce a *lumped* mass matrix $\mathcal{M}$ with diagonal entries[2], whose inverse is straightforward to calculate and its use preserves the original sparsity pattern, avoiding working with full matrices.

3.2 Numerical integration

For the numerical surface integration it is possible to discretize in space using the *Gauss-Legendre quadrature*

$$
\frac{1}{A_k}\int_{\Omega_k} f(x,y)\,dx\,dy \approx \sum_j w_j\, f(x_j,y_j) \tag{16}
$$

where A_k is the area of the triangle Ω_k, (x_j, y_j) the respective quadrature points and w_j the *Gauss-Legendre* weights.

[2]Under some regularity conditions on the mesh the use of lumped matrices give second order in accuracy

3.3 Nonlinear iterations

In general a nonlinear system of the form

$$\tilde{\mathbb{M}}\frac{\partial \mathbf{y}}{\partial t} = \mathcal{F}(\mathbf{y}) \tag{17}$$

can be built for the system (14) by defining the new vector

$$\mathbf{y} = (\phi_1, \phi_2, \cdots, \phi_n, \psi_1, \psi_2, \cdots, \psi_n)^T \,. \tag{18}$$

When discretized in time by first order forward finite differences, it is obtained

$$\tilde{\mathbb{M}}(\mathbf{y}^{n+1} - \mathbf{y}^n) = \Delta t\, \mathcal{F}(\mathbf{y})\,. \tag{19}$$

Notice that the matrix $\tilde{\mathbb{M}}$ is a diagonal block matrix containing $\mathbb{M}$ in each of its two entries. Assuming the splitting $\mathcal{F}(\mathbf{y}^n, \mathbf{y}^{n+1}) = P(\mathbf{y}^n) + Q(\mathbf{y}^{n+1})$ for the explicit and implicit terms, yields

$$\tilde{\mathbb{M}}(\mathbf{y}^{n+1} - \mathbf{y}^n) = \Delta t\, P(\mathbf{y}^n) + \Delta t\, Q(\mathbf{y}^{n+1})\,. \tag{20}$$

Thus, the system to be solved for is

$$\mathcal{N} = \tilde{\mathbb{M}}\mathbf{y}^{n+1} - \tilde{\mathbb{M}}\mathbf{y}^n - \Delta t\, P(\mathbf{y}^n) - \Delta t\, Q(\mathbf{y}^{n+1}) = \mathbf{0}\,. \tag{21}$$

Newton's method to solve nonlinear problems consists of iterating over two steps. First, computing a correction $\delta\mathbf{y}_k$ in the vector system

$$\mathbb{J}\,\delta\mathbf{y}_k = -\mathcal{N}(\mathbf{y}) \tag{22}$$

with Jacobian

$$J_{ij} = \frac{\partial \mathcal{N}_i}{\partial y_j}$$

. After that, the solution is updated by computing $y_{k+1}^{n+1} = y_k^{n+1} + \delta_k$ and repeat. Notice that in the system (21) the variable to solve for is $\mathbf{y}^{n+1}$ and taking the derivatives of the vector $\mathbf{y}^n$ vanishes, meaning that it has no impact on the Jacobian. This is the gain of using explicit methods. However, as it has been said before, the fact that the resulting system is stiff preventing us from using solely explicit methods. Nontheless, there are alternative ways to put more weight into the explicit part to lessen the Jacobian computation.

3.4 Mass conservation

In the experiments shown in this project, there are no external sources or sinks of mass (phase ψ). The evolution of the system is caused by relaxation, which leads to reaccomodation procesess due to the nature of the phenomena. Hence, the total phase of the system must be preserved and its conservation will serve as a measurement of the error in the numerical solution

$$\int_\Omega \phi\, d^3x = \int_\Omega \sum_j \phi_j N_j\, d^3x = \text{constant}\,. \tag{23}$$

Using the property $\sum_i N_i = 1$ for the FEM basis functions and inserting this into the mass conservation integral yields

$$\int_\Omega \sum_i N_i \sum_j \phi_j N_j \, d^3x = \sum_i \left(\sum_j \phi_j \int_\Omega N_i N_j \, d^3x \right) = \sum_{i,j} M_{ij} \, \phi_j \tag{24}$$

which can be easily calculated on each time-step.

4 Split-Step Methods

Usually the discretization in time can be performed by using any of the standard methods, such as the fully implicit Backward Euler (BE) method, or evaluating at an average with the Midpoint Rule (MidP) method, Trapezoidal (TPZ) method or any other clever combination of these. One approach that has been proposed [Ey], is to adopt a compromise between explicit and implicit methods, taking advantage of the resulting form of the equations by weighing the terms as a combination that leads to increasing performance when used along with an adaptative time-stepping scheme. For instance, a term $(\phi^{n+1})^3$ is clearly an non-linear implicit term and hence Jacobian evaluations are needed to solve it implicitly. The same term can be written as $(\phi^n)^2\phi^{n+1}$ transforming the latter term into a linear term for the implicit solver, avoiding Jacobian calls when a Jacobian refactorization scheme is in use[3]. Nonetheless, introducing such artificial terms increases the truncation error in the system.

The way that some terms are passed as explicit and some other as implicit into the solver in (14), dependes on the nature of such terms (expansive or contractive). Linearity of the implicit part means evaluating constant Jacobians, cheaper to calculate. This can be achieved by passing some nonlinear terms in the implicit step to be part of the explicit part. Eyre shows stability for these split methods and propose a set for the Cahn-Hilliard equation (1) that will be used and extended here. There is no unique way to split the problem, but some characteristics of the gradient flows [Ey] have to be taken into account due to stability requirements. See in [Ey] the analysis of expansive and contractive terms in gradient-flow-type problems.

It is aimed, to study the behavior of these ODEs under a combination of explicit and implicit methods, giving different importance to each of the terms in the equation according to the nature of the nonlinearity. *'Split methods'* is the name coined for this technique, leading to different improvements in performance depending on the split used, for instance reducing the number of function calls compared to standard methods. Even though choosing the way that the split is done is not unique, it is possible to obtain valuable information about the solution process and differences in explicit and implicit methods. Some common time discretization methods for the Cahn-Hilliard equation can be obtained by rewritting (1) in

[3] A complete description on Jacobian refactorizations and adaptative time-stepping schemes is given in [So1] and [So2]

the form

$$\begin{aligned} \Phi &= -\phi + \phi^3 - \tfrac{\mathrm{Cn}^2}{2}\nabla^2\phi\,, \\ \frac{\partial\phi}{\partial t} &= \tfrac{1}{\mathrm{Pe}_\phi}\nabla^2\Phi\,, \end{aligned} \tag{25}$$

and then discretizing in time as follows

$$\Phi^n = -\phi^n + (\phi^n)^3 - \tfrac{\mathrm{Cn}^2}{2}\nabla^2\phi^n \tag{26}$$

Next, some time stepping schemes are presented.

Backward Euler

$$\phi_t^n = \tfrac{1}{\mathrm{Pe}_\phi}\nabla^2\Phi^{n+1}\,. \tag{27}$$

Midpoint rule

$$\phi_t^n = \tfrac{1}{\mathrm{Pe}_\phi}\nabla^2\Phi\left(\frac{\phi^n + \phi^{n+1}}{2}\right). \tag{28}$$

Trapezoidal rule

$$\phi_t^n = \tfrac{1}{2\mathrm{Pe}_\phi}\left(\nabla^2\Phi^n + \nabla^2\Phi^{n+1}\right). \tag{29}$$

For the split-step methods presented in [Ey], a similar approach is used by changing the discretization for a combination of the form $\Phi^*(\phi^n, \phi^{n+1})$.

Eyre's split

$$\Phi^* = -\phi^n + (\phi^n)^2\phi^{n+1} - \tfrac{\mathrm{Cn}^2}{2}\nabla^2\phi^{n+1}\,. \tag{30}$$

Semi-implicit

$$\Phi^* = -\phi^n + (\phi^n)^3 - \tfrac{\mathrm{Cn}^2}{2}\nabla^2\phi^{n+1}\,. \tag{31}$$

Non-linearly stabilized

$$\Phi^* = -\phi^n + (\phi^{n+1})^3 - \tfrac{\mathrm{Cn}^2}{2}\nabla^2\phi^{n+1}\,. \tag{32}$$

Linearly stabilized

$$\Phi^* = -3\phi^n + 2\phi^{n+1} + (\phi^n)^3 - \tfrac{\mathrm{Cn}^2}{2}\nabla^2\phi^{n+1}\,, \tag{33}$$

and computing afterwards a simple scheme for the time stepping

$$\phi_t = \frac{\phi^{n+1} - \phi^n}{k} = \tfrac{1}{\mathrm{Pe}_\phi}\nabla^2\Phi^*\,. \tag{34}$$

4.1 Split-step methods for the surfactant

The system of equations in (14) is highly nonlinear, therefore, finding good split methods is slightly more complicated than for the Cahn-Hilliard equation. In particular, the non-constant mobility $M_\psi = \psi(1-\psi)$ makes this task more difficult. However, it is possible to mimic the most evident features of the split-steps presented before. In this study the following three split methods are analyzed.

Semi-Implicit

$$\begin{aligned}
\mu_\phi &= -\phi^n + (\phi^n)^3 + \tfrac{1}{2Ex}\psi^{n+1}\phi^n - \left[1-(\phi^n)^2\right]\phi^n\psi^{n+1} - \tfrac{Cn^2}{2}\nabla^2\phi^{n+1}\,,\\
\mu_\psi &= -\tfrac{1}{4}\left[1-(\phi^n)^2\right]^2 + \tfrac{1}{4Ex}(\phi^n)^2 + \mathrm{Pi}\log\frac{\psi^{n+1}}{1-\psi^{n+1}}\,,\\
M_\psi &= \psi^{n+1}(1-\psi^{n+1})\,.
\end{aligned} \tag{35}$$

Non-linearly stabilized

$$\begin{aligned}
\mu_\phi &= -\phi^n + (\phi^{n+1})^3 + \tfrac{1}{2Ex}\psi^{n+1}\phi^{n+1} - \psi^n\phi^n + \psi^{n+1}(\phi^{n+1})^3 - \tfrac{Cn^2}{2}\nabla^2\phi^{n+1}\,,\\
\mu_\psi &= -\tfrac{1}{4}\left[1-2(\phi^n)^2+(\phi^{n+1})^4\right] + \tfrac{1}{4Ex}(\phi^{n+1})^2 + \mathrm{Pi}\log\frac{\psi^{n+1}}{1-\psi^{n+1}}\,,\\
M_\psi &= \psi^n(1-\psi^n)\,.
\end{aligned} \tag{36}$$

Linearized (Linearly-stabilized)

$$\begin{aligned}
\mu_\phi &= -\phi^n + (\phi^n)^2\phi^{n+1} + \tfrac{1}{2Ex}\psi^n\phi^{n+1} - \psi^n\phi^n + \psi^n(\phi^n)^2\phi^{n+1} - \tfrac{Cn^2}{2}\nabla^2\phi^{n+1}\,,\\
\mu_\psi &= -\tfrac{1}{4}\left[1-(\phi^n)^2\right]^2 + \tfrac{1}{4Ex}(\phi^n)^2 + \mathrm{Pi}\log\frac{\psi^{n+1}}{1-\psi^{n+1}}\,,\\
M_\psi &= \psi^{n+1}(1-\psi^n)\,.
\end{aligned} \tag{37}$$

Compared to the equations (4) it is clear that some nonlinearities have been changed in such a way that there can be a gain in linearity of the implicit terms (ϕ^{n+1}, ψ^{n+1}) or the degree of such nonlinearity is reduced.

5 Code features

The existing code FLOW was provided as the platform for testing split-step methods in the sense of gradient flows (see [Ey]), as an approach to the solution of problems in two phase

flows. The MATLAB code for ODE1S and the FLOW package along with its functionality were explained and the objective of the project was to present suitable solutions to the surfactant in two-phase flows under split-step methods and to evaluate their efficiency in the two-dimensional case.

The plataform for the implementation of this code was developed in MATLAB by using the tools provided within the PDETOOL package, which is a strong tool for solving PDEs with Finite Element Method that contains several built-in definitions for solving standard PDE problems. The Delaunay triangulation is one of the strongest PDETOOL features used in this project, which generates the best set of triangles by maximizing the angles in each triangle (acute angles lower the accuracy of the spatial discretization). PDETOOL also provides the fast assembly of the matricial objects in definitions (15).

As a first approximation, PDETOOL for FEM in MATLAB uses only one point quadrature in the midpoint of each triangle. For a better accuracy more quadrature points need to be considered.

Once the spatial discretization is set, the time stepping is taken care of by ODE1S, a MATLAB implementation for computing the solution of systems of ODEs developed by Stefan Engblom. ODE1S is a solver for stiff nonlinear ODEs that relies on several features as *Jacobian refactorizations*, *Digital filters* and *time step adaptivity*[4].

In order to use ODE1S as a solver, some modifications were implemented. For instance, a counter carrying statistics of the solution, a sparsity pattern used for speeding up the numerical Jacobians, the allowance of passing a mass matrix in the ODE definition, a sparse direct solver (LU) and an interface passing all the numerical objects to ODE1S for the specific problem (14). At first, ODE1S was using the BE method only but the extension to the MidP method, TPZ method and split-step methods was a crucial development in this project.

As mentioned, the Jacobian $\mathbb{J}$ is calculated numerically by ODE1S using the matlab function *numjac*. A sparse pattern $\mathbb{S}$ is a matrix with ones in the nonzero Jacobian arguments, *numjac* is called according to $\mathbb{S}$ avoiding void calculations of zero entries. In ODE1S every new time step comes as a result of an internal automatic comparison of three reference points, where as a result, the new step length to be taken in order to mantain the desired accuracy is returned. Significant understanding can be obtained from plots of number of function calls vs the tolerance reached as a measurement of performance, see for example Figure 4.

6 Numerical experiments

In order to understand the dynamics of the behavior of surfactants in diffusive flows several experiments are presented below. In all the experiments the initial condition ψ_b is set to be constant throughout the domain. This is an ideal approximation to be considered as an academic experiment only. In reality it is impossible to achieve a constant concentration ψ_b everywhere due to the boundary-tracing nature of the surfactants.

[4]For more details see [So1], [So2] and [Gu].

One difficulty that arises when dealing with nonlinear problems is to show that a computed numerical solution is sufficiently good. What makes this task even harder is finding reliable results to make comparisons allowing the possibility to decide whether a solution is well behaved or not. The model in equations (3) is developed in such a way, that the concentration of the surfactant in the solution moves towards the regions where $\phi = 0$. This should be the most important indicator that a solution is following the right path.

There are several ways to track the good behavior of a numerical solution, one of the most important characteristics on gradient flows, is that the energy tends to decrease in the process. In this way, each time step should show a decrement in the total energy of the system following a relaxation process until a steady state phase is reached. Furthermore, the model (3) describes a decrease of the surface tension due to the inclusion of the surfactant. This relaxation would take longer times to complete than without initial surfactant, similarly than when a spring with a small elastic constant would last a longer period to complete a cycle compared to a spring with a higher elastic constant carrying the same weight. It is possible to keep track of this relaxation time in the numerical simulation by setting a common point, say the time when two bubbles first touch forming a 8-shaped figure (see figure 5, right).

Another valuable measurement that is used in this report, is keeping track of the mass conservation within the domain. For this, at each time step the quantity (24) is traced in order to control how good a method performs regarding mass-conservation. Notice that the Neumann-type boundary conditions used to build the weak form, account for the mass-conservation in the system.

6.1 Experiments in 1D

The main interest in the current report lies in the correct implementation of the 2D spatial discretization. For this, it is necessary to have a well proved initial example in 1D. The work presented here is based on a solution provided by Stefan Engblom, approximating the equation (3) by using Legendre-spectral-methods for the discretization in space. The weak form in that case has a similar structure that the one used for FEM. However, the 1D case is built by using Legendre polynomials as test functions. The computational experiment was set for a number of polynomials $np = 100$ and compared with a second run using $np = 200$ allowing to measure performance for two different sizes of the Jacobian matrix.

The initial conditions are $\phi(x, 0) = \cos(2\pi\, x) + \sin(\frac{\pi}{2}\, x)$ and $\psi(x, 0) = \psi_b$. It is observed in Figure (2) that while ϕ tends to get stabilized into two well-behaved phases, ψ starts from a constant value (left), moves to the regions where $\phi = 0$ (right) resulting in a narrow-Gaussian-like peak centered in the final interface (down).Tracking always the regions where $\phi = 0$.

The experiment was run several times varying the required accuracy (TOL) from 10^{-3} to 10^{-7} in order to measure performance. The plot of TOL vs number of function calls (nfun) is presented in Figure 3 and Table 1 shows data for the specific case $TOL = 10^{-4}$.

Method	nStpAccepted	nFailIntTol	nfun	njac	nlinsolve
BE	67	2	2153	8	551
MidPoint	54	0	1617	6	415
TPZ	52	0	1608	6	406
Semi-impl	93	0	1159	2	755
non-linStab	79	3	1934	6	732
Linearized	86	0	1550	4	748

Table 1: *Statistics on the solution of the surfactant system in 1D using* $np = 100$ *with* $TOL = 10^{-4}$.

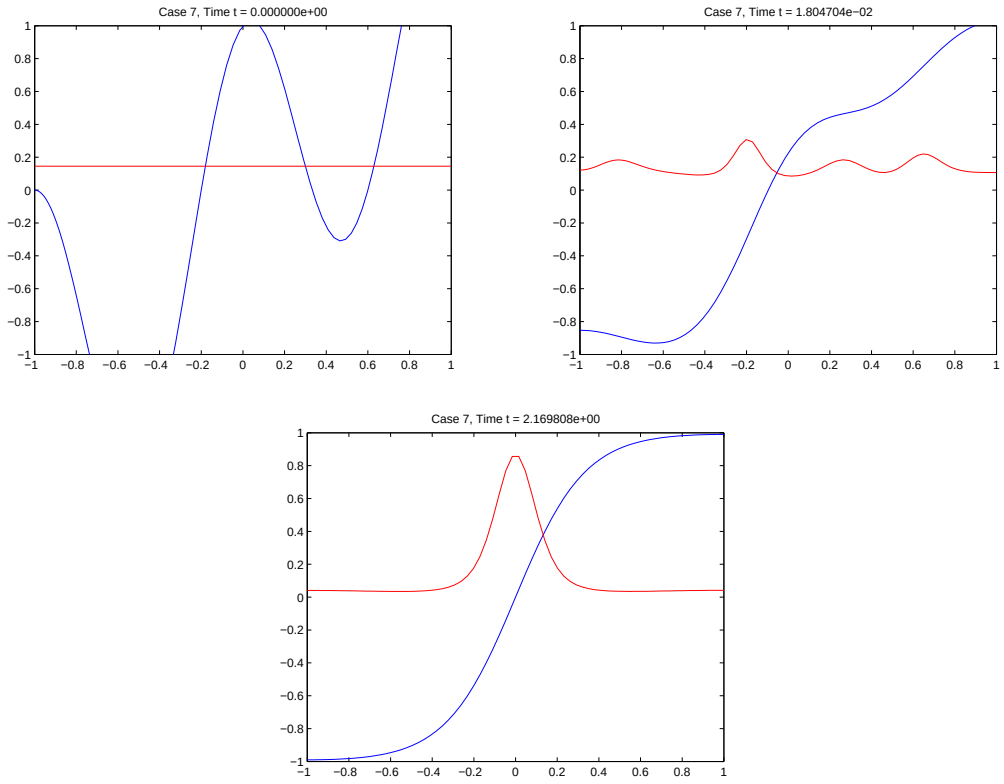

Figure 2: *Evolution of the 1D surfactant problem for* ϕ *in blue and* ψ *in red. Left: initial conditions, right: showing the dynamics, down: stady state solution.*

6.2 Experiments in 2D

The first experiment is the planar version of the sinusoidal initial condition for ϕ given in the 1D case, of the form $\psi(x, y, 0) = \cos(2\pi\, x) + \sin(\frac{\pi}{2}\, x)$. The behavior of the solution remains fairly similar as in the 1D problem.

The second experiment to be considered is the coalescence of two bubbles in a squared domain of dimension $x, y \in [-1, 1]$. The distance between the bubbles is set to be close enough for them to experience coalescence. The expected behavior for this experiment is that after the boundary of the bubbles come in touch, the surface tension then tends to lower the energy by relaxing the surfaces ending up in a bigger bubble. The shape of the solution remains quite similar in all cases for initial surfactant concentrations, but the presence of a higher value of ψ_b impacts on the time scale of the solution, taking a longer time to complete one characteristic period.

To illustrate this fact Table 4 shows some characteristic times taken by running the numerical experiment with approximately 20000 triangles varying ψ_b for a number of choices and registering the time needed for the bubbles to first touch, forming a 8-like boundary (see Figure 5, right).

7 Discussion

The discussion starts by comparing the 1D problem to its 2D planar representation under the same initial conditions. What is immediately apparent from Figure 3 (up), is that the split methods in the 1D case, are cheaper to use up to a limit of $\sim 10^{-4.5}$ in relative error, bounded by the crossing point where all the methods seem to perform similarly. When the number of Legendre polynomials doubles from $np = 100$ to $np = 200$, this crossing point moves to $rTol \approx 10^{-5}$ (Figure 3, down), achieving one order higher in accuracy. When the number of polynomials increases, the Jacobian of the system becomes costly to calculate and the split-methods appear to be more efficient than the standard methods. Take for instance the performance of the Semi-implicit with the lowest function calls $nfun = 1159$ in table 1, for a $TOL = 10^{-4}$ and only two of the Jacobian refactorizations. The split methods performed better than BE method, MidP method and TPZ method in the region where $10^{-4} < TOL < 10^{-3}$.

From Table 1 it is clear that for the split methods the number of required time steps increases. As discussed before, the truncation error is greater for the split methods[5]. This makes ODE1S more suspicious producing shorter time steps, and hence, more of them compared to standard methods.

As a first conclusion, the desired gain in performance is bounded in a region where the split methods are cheaper and more efficient up to a point where the number of steps needed become too many, causing the performance to drop.

The fact that the performance of the split-methods increases when the Jacobian of the

[5]For more a detailed reading see [Ey].

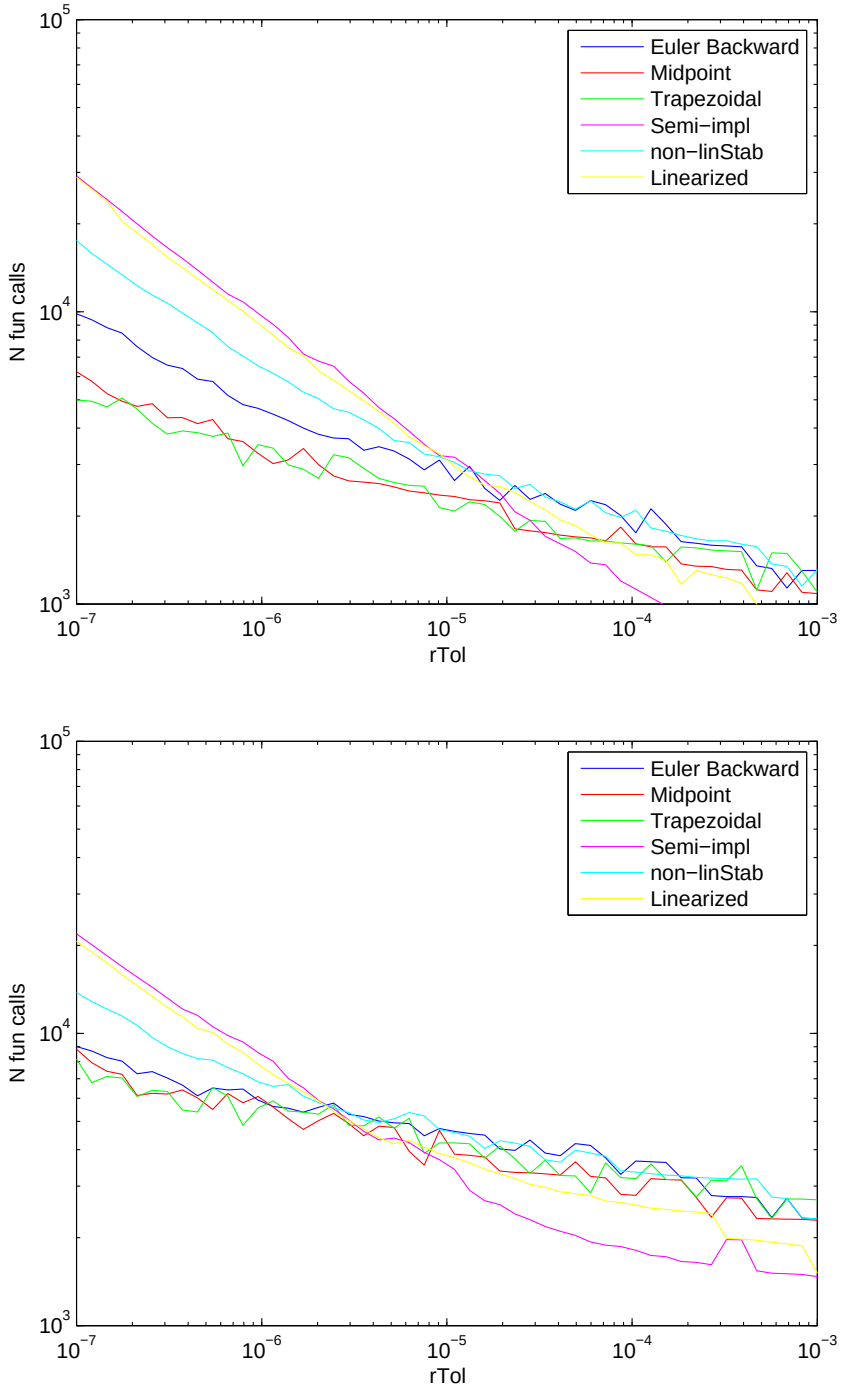

Figure 3: *Plots showing rTol (x-axis) vs nFun (y-axis) in the solutions of the surfactant in 1D (6), for the first 100 Legendre polynomials (up), and 200 polynomials (down).*

Method	nStpAccepted	nFailIntTol	nfun	njac	nNit	nLU	it/stp
BE	179	0	14611	196	1869	398	10.441341
MidPoint	148	2	4003	37	1596	136	10.783784
TPZ	142	7	12841	172	1659	341	11.683099
Semi-impl	208	3	4775	32	2693	118	12.947115
Linearized	195	2	3378	15	2401	88	12.312821

Table 2: *Statistics on the solution of the planar surfactant system in 2D using* 20000 *triangles with* $TOL = 10^{-4}$.

Method	nStpAccepted	nFailIntTol	nfun	njac	nNit	nLU	it/stp
BE	248	9	18043	230	3091	473	12.463710
MidPoint	615	10	9611	19	8374	126	13.616260
TPZ	181	6	18334	243	2537	432	14.016575
Semi-impl	847	8	13365	19	12128	124	14.318772
Linearized	883	15	13386	11	12669	120	14.347678

Table 3: *Statistics on the solution of the planar surfactant in 2D using* 20000 *triangles with* $TOL = 10^{-5}$.

system is twice larger, permits to justify their use in higher dimensions or in cases where Jacobian evaluations are more expensive. This is the main motivation for moving into the 2D.

Looking at Table 2, it is evident the Linearized great performance with only 3378 function calls and only 15 Jacobian refactorizations, producing 195 time-steps for a relative tolerance of 10^{-4} (inside the high performance gap), which is a good accuracy for general purposes. The MidP, as the second preferred, requires 4003 function calls and 37 Jacobian refactorizations resulting in 148 time-steps. Note that even though the Linearized performs better than the MidP it has to take more time-steps as explained before.

The above said is not the case for $TOL = 10^{-5}$ (outside the high performance gap) in Table 3, where the top method is the MidP with 9611 function calls and 19 Jacobian refactorizations, followed by the Semi-Implicit with a higher number of 13365 function calls but the same 19 Jacobian refactorizations. However, the latter performes better than the TPZ, who took 18043 function calls and 243 Jacobian refactorizations, being large numbers compared to the two first methods mentioned before.

Proving performance is not enough evidence to assert whether a method is behaving properly or not. To illustrate this, Figure 6 (down) shows a histogram of the mass of the system defined in (24) along the time steps for 5000 triangles in the 2D problem. What is significantly apparent from this figure is that the TPZ and the Linearized present curves that remained fairly constant throughout each time step. Hence, they conserve the mass of the system in higher accuracy, proving to be more stable than the rest of the considered methods.

Another alternative way to confirm the good behavior of a numerical solution is deduced

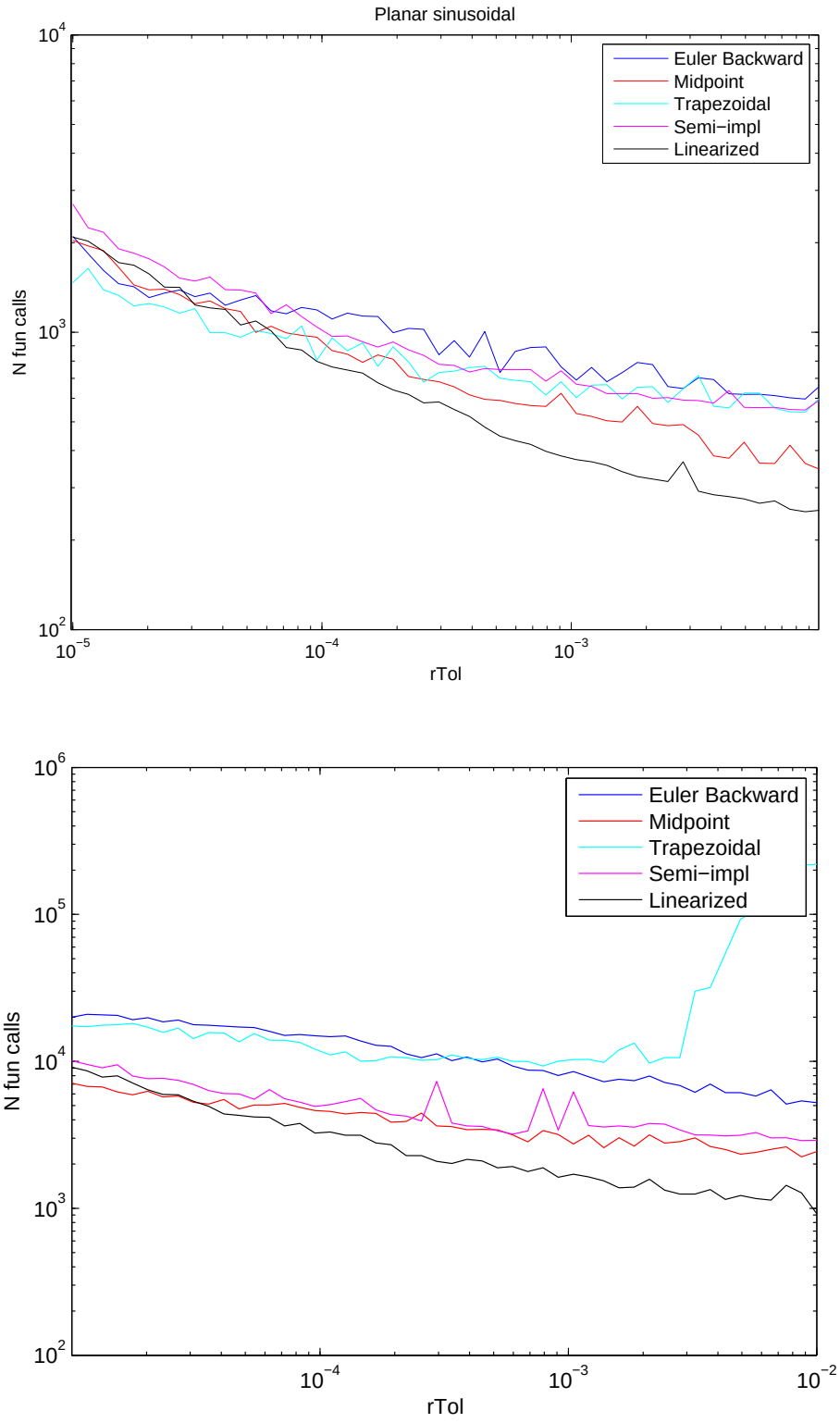

Figure 4: *Performance plots for the 2D surfactant problem. Up: Planar experiment with 5000 triangles. Down: Coalescing bubbles experiment with 20000 triangles.*

by the physics of the phenomena. Figure 6 (up) depicts the relaxation time (Table 4) in which two air bubbles under water, with ψ_b as the initial concentration of surfactant, collide (see Figure 5). It is apparent that for a concentration $0 < \phi_b < 0.25$ the relaxation time appears to have an exponential behavior (approximately linear in the logarithmic Figure 6, down).

8 Conclusions

The split methods as an approach to solve nonlinear problems present an alternative way to reduce the cost per iteration compared to fully implicit solvers. They preserve important characteristics of gradient flows as for example conservation of mass. The main idea is to combine implicit and explicit terms in such a way that the computational cost can be reduced. The lost is in truncation error, becoming evidently for lower relative erros than 10^{-4} in this work. For higher relative errors the split methods perform better than the conventional techniques.

The analysis done in the coupled surfactant-phase field problem showed that the advantages of the split methods lie in the cost per Jacobian evaluations. Greater performance can be seen with the split methods when the cost of Jacobian evaluations is increased, therefore, the improvement seen in the 1D case is not as dramatic as in higher dimensions.

It is shown how the use of split-step methods in 1D leads to a cheaper numerical solution up to a relative tolerance of $10^{-4.5}$. When the number of the polynomials used is doubled, the tendency is to perform even better, that is up to 10^{-5} resulting most suitable for problems with expensive Jacobian calculations. This last fact motivates working with higher dimensions where the Jacobian is far more expensive than in the 1D case.

As expected, the 2D case follows the same trend of ideas as in the 1D case. However, the linearized-split offers in both cases (Figure 4, up with 5000 and down with 20000 triangles) a better performance than the rest of methods. The semi-implicit and the MidP show similar performance than the TPZ or BE with 5000 triangles.

For 20000 triangles, the BE or TPZ perform poorly compared to the other three methods. Justifying the initial motivation for the use of the split-methods in higher dimensions. On the other hand, the MidP presents fairly similar results to the semi-Implicit but showing better stability.

As a final overview of the course, I gained valuable knowledge on solving systems of nonlinear PDEs numerically, and learnt from the correct implementation of the Newton method for implicit solvers among several other details in an automatic schemes based on a posteriori error bounds. The differences between explicit and implicit time discretizations were reinforced along the course and new trends in cutting-edge solvers were introduced. The implementation of a model problem by using finite element methods deepened the knowledge obtained in former courses within the MSc in Computational Science programme.

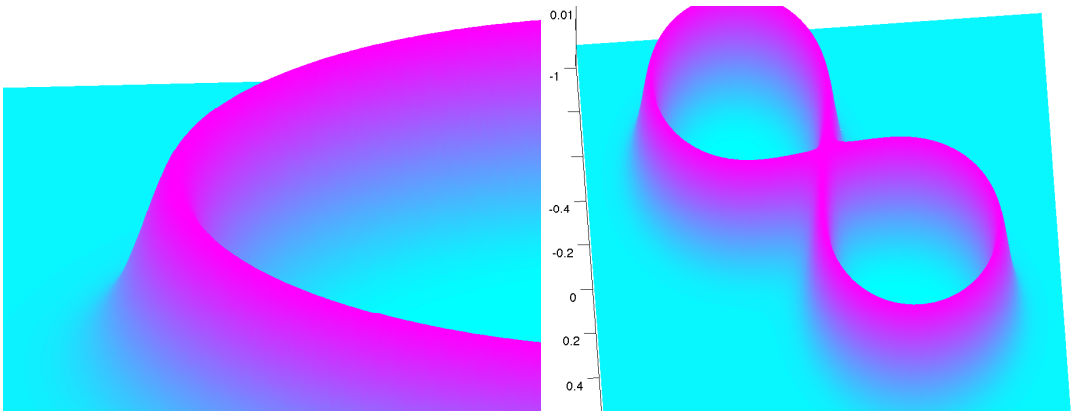

Figure 5: *Diagrams for an arbitrary solution in the Coalescing bubbles experiment with 20000 triangles showing the smoothness of the solution. Right: 8-like shape when the bubbles first touch in the coalescence process.*

ψ_b	t_c	ψ_b	t_c
0	0.3705	0.2250	2.3285
0.0250	0.5328	0.2500	3.4656
0.0500	0.5230	0.2750	5.1274
0.0750	0.7452	0.3000	7.5394
0.1000	0.7559	0.3250	13.5625
0.1250	1.0630	0.3500	24.9005
0.1500	1.0871	0.3750	57.4483
0.1750	1.5510	0.4000	140.2650
0.2000	1.5960	-	-

Table 4: *Relaxation times in the coalescing bubbles under the precence of ψ_b surfactant with a resolution of 5000 triangles and a tolerance of $TOL = 10^{-3}$.*

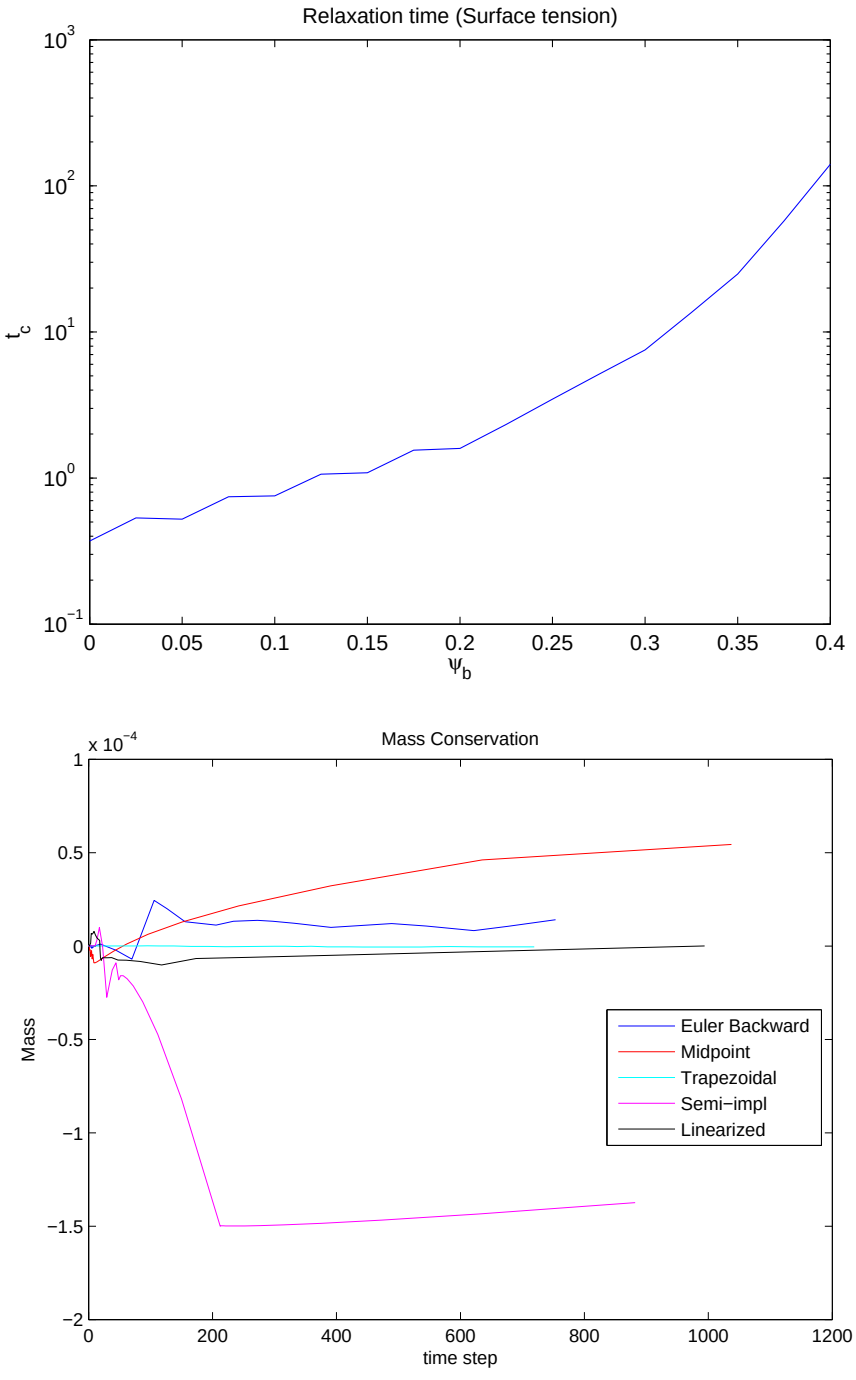

Figure 6: *Up: Relaxation times in the bubbles coalescencing experiment for initial concentrations ψ_b with 5000 triangles and $TOL = 10^{-3}$. Down: Conservation of mass in the system according to the solution-method in use.*

8.1 Future work

The field of possible improvements deimited by this proyect is broad and extensive. Currently, ODE1S solves the linear system (22) by using a sparse implementation of direct solvers provided by MATLAB. Long delays were experienced when calculating the solutions for a large number of triangles. An iterative solver like GMRES would be preferable to the current LU decomposition, since GMRES accounts for the sparsity of the matrices. Nonetheless, a study on suitable initial guesses must follow. For instance, an incomplete LU could serve as a good starting point. Another possibility to decrease the computational cost, is to provide user-computed-Jacobians to ODE1S that serves to replace the current numJac boosting the algorithm.

When controlling the error, ODE1S estimates the size of the new time-step by using L2-norms on the solution ϕ and ψ. A better approach is to consider a energy-norm derived from the physical system.

The paralelization of the code is a non-trivial task to be implemented, for the code in ODE1S needs to be parsed to C or C++ before coding in OMP, PThreads or MPI. However, it is definitely an important point for future improvements.

Another idea to avoid the use of a lumped matrix in the solution of (14) is to use high order basis functions, namely the Lagrange-cubic-basis prevents the system from being split in two stages due to lack of smoothness and continuity of the solution. Other kind of high-order basis functions are the Hermite polynomials, which includes an approximation for the derivatives as degrees of freedom. The use of the latter not only avoids the use of artificial inclusion of new nodal points, like in the higher order Lagrange polynomials, but also increases considerably the accuracy of the method.

As a first approach to the advective problem in fluid flows a static advective matrix can be implemented, meaning that the presence of surfactant nor the behavior of ϕ will be considered as sources altering the velocity field. Here, a comprehensive study should follow in order to guarantee the conservation of mass in the system. A rigorous stability analysis of the system is necessary at this stage since FEM is proven to be unstable for advective problems.

A further step in the latter direction would be to investigate the effects of surfactant on the velocity field. For this research topic, coupling equations (3) to the incompressible Navier Stokes equations is a necessary stage that requires an extensive code development. In an earlier work done in [En] it was proven that the bubbles can bounce away from each other instead of the coalescing experienced in the current project. The resulting bouncing is caused by alterations of the velocity field due to the presence of high concentrations of the surfactant.

Adaptivity is an elaborated solution, which consists in increasing the computational power in the regions where the boundary ($\phi = 0$) is located, making the mesh finner and avoiding extra effort in the domains were the solution is sufficiently smooth.

References

[En] S. Engblom, M. Do-Quang, G. Amberg, A-K. Tornberg. *On modelling and simulation of surfractants in diffuse interface flow*, Dept of Scientific Computing Uppsala Universitet, Jun 2011.

[Ey] David J. Eyre, *Unconditionally gradient stable time marching the Cahn-Hilliard equation*, Dept of Math. University of Utah, May 1998.

[Le] Vladimir Lebedev and Alene Sysoeva, *Unconditionally gradient-stable computational schemes in problems of fast phase transitions*, Physical Review E 83, 026705, 2011.

[Gu] Kjell Gustafsson and Gustaf Söderlind, *Control strategies for the iterative solution of nonlinear equations in ode solvers*, SIAM J. Sci Comput. Vol 18, No 1, Jan 1997.

[So1] Gustaf Söderlind and Lina Wang, *Adaptive time-stepping and computational stability*, Journal of Computational and Applied Mathematics, March 2003.

[So2] Gustaf Söderlind, *Digital filters in Adaptive time-stepping*, ACM Transactions on Mathematical Software, Vol. 29, No. 1, March 2003.