Application of the Laminar Navier-Stokes Equations for Solving 2D and 3D Pathfinding Problems with Static and Dynamic Spatial Constraints. Implementation and validation in Comsol Multiphysics.

Benjamin Ivorra
MOMAT research group & Instituto de Matemática Interdisciplinar,
Departamento de Matemática Aplicada,
Universidad Complutense de Madrid,
Plaza de Ciencias 3, 28040, Madrid, Spain Tel.: +34-91-3944415
Fax: +34-91-3944613
email:ivorra@mat.ucm.es

Abstract

Pathfinding problems consist in determining the optimal shortest path, or at least one path, between two points in the space. In this paper, we propose a particular approach, based on methods used in Computational Fluid Dynamics, that intends to solve such problems. In particular, we reformulate pathfinding problems as the motion of a viscous fluid via the use of the laminar Navier-Stokes equations completed with suitable boundary conditions corresponding to some characteristics of the considered problem: position of the initial and final points, a-priori information of the terrain, One-way routes and dynamic spatial configuration. Then, we propose and validate a numerical implementation of this methodology by using Comsol Multiphysics (i.e., a Finite Element Methods software) and by considering various experiments. We compare the obtained results with those returned by a classical pathfinding algorithm. Finally, we perform a sensitivity analysis of the proposed algorithms with respect to some key parameters.

Keywords: Pathfinding, Computational Fluid Dynamics, Comsol Multiphysics, Spatial Constraints, Artificial Intelligence

1. Introduction

Pathfinding problems are based on the determination of the shortest route between two points in the space, or at least a reasonable route [47]. This kind of problem can be found in a wide range of disciplines such as: Artificial Intelligences (eg., movements of robots [8]), Navigation systems (e.g., GPS [2]) or Leisure activities (e.g., Video games [52]).

Currently there exist various algorithms which intend to solve this kind of problems and their variations. For instance, the **A*** algorithm (e.g., see [53]) or the Any-angle path planning algorithms ([13]) which are both based on a graph representation of the space. Those algorithms can deal with a variety of pathfinding problems such as: single or multi-agents cases, limited knowledge of the spatial domain or dynamic environments. One of the main drawbacks of those classical algorithms is the fact that they require abstraction methods to transform the considered spatial area into a compatible domain (e.g., a graph in the case of the **A*** or the Dijkstra's algorithm, see [16]), making their implementation not straightforward.

One alternative to those algorithms can be found in the use of differential equations, and in particular those coming from the Computational Fluid Dynamics (CFD) field [4, 10, 22, 28, 29]. Regarding the existing literature, a pioneer work introducing this idea was presented in [32]. In this article, the author propose a method based on the use of an artificial potential function, determined by solving a partial differential equation, to determine the path taken by a robot to avoid obstacles in real time. This methodology was then applied (see, for instance, [12, 43]) to develop general robot navigation algorithms. In [11], 2D and 3D maze problems with static geometries are solved by using the Laplace's equation. Later, in [37], a method based on the incompressible Navier-Stokes equations for handling 2D pathfinding problems with restriction on the vehicle's ground friction has been described. This method was also validated experimentally by using real liquid systems in [19, 46]. Recently, in [36, 40], some CFD equations were applied to determine the optimal route of a swarm of ground or underwater vehicles in 2D domains with restrictions (e.g., power consumption, avoiding collisions or forbidden routes). Moreover, in [1, 50], a wave equation is used to build a cognitive map of the spatial environment and guide a robot in real-time. From a numerical point of view, in all those works authors have used their own numerical implementations by considering finite element, finite difference or level set methods. Finally, we note that Fluid Dynamics equations

have been also applied to solve problems in other area of transportation, such as: the modeling of traffic car density [6] or the optimization of road networks [38].

The objective of this work is threefold. Firstly, we extend the use of CFD methods for the resolution of pathfinding problems that include:

- Incomplete information of the spatial domain;
- Dynamical evolution of the environment;
- One-way routes;
- and 3D environments for ground or nonholonomic flying vehicles [44].

Secondly, we present and validate a particular implementation of the proposed algorithms by using Comsol Multiphysics 5.0 [48] coupled with Matlab 2014a [24]. Indeed, Comsol Multiphysics is a modeling software based on Finite Element Methods (FEM) [21, 42]. It allows obtaining easily robust numerical version of Fluids Dynamics models [22, 26, 27, 28, 29]. In its last versions (4.0 and later), this program produces accurate solutions of CFD equations in a reasonable computational time when compared to other commercial software (e.g., ANSYS Fluent) [25]. This software is also used as a formative tool in order to illustrate the possible applications of FEM to industrial problems [34, 35]. Thus, one of our aim is to propose easy-to-implement pathfinding algorithms to Comsol Multiphysics users. We note that the use of Comsol Multiphysics is not mandatory to implement the methods proposed here. Indeed, any other FEM toolbox can be used as an alternative implementation framework. Thirdly, we perform a sensitivity analysis of the proposed algorithms in order to study the impact on the solutions of changes in the values of some key parameters.

To do so, in Section 2, we introduce the continuous equations and algorithms used to model and solve different pathfinding problems that include some of the characteristics presented previously. Next, in Section 3, we propose a particular implementation of our approach with Comsol Multiphysics 5.0 and Matlab 2014a software. Finally, in Section 4, we validate our approach by considering various benchmark pathfinding problems and by analyzing the obtained results. When possible, we compare those outputs with the ones returned by a classical **A*** algorithm implemented in Matlab. We note that this **A*** is only considered as a reference value to be compared

with other pathfinding methods found in the literature [7, 33, 45]. We also provide a sensitivity analysis of the considered algorithms with respect to some parameters.

2. Mathematical models and algorithms

In this Section, we introduce the equations and algorithms used to model and solve the different classes of pathfinding problems studied during this work.

2.1. Basic problem

First, we focus on the problem of finding a path between a starting point S and a final point F in a 2D space delimited by walls. To do so, let $\Omega \in \mathbb{R}^2$ be the geometrical representation of the space. We assume that S and F are included in $\Omega \backslash \partial\Omega$, where $\partial\Omega$ is the boundary of Ω. Following [37, 19, 46], we want to model the motion of a fluid entering by a small area including S and exiting by a small area including F, when this physical system is at equilibrium. To do so, let $B_S(\epsilon)$ and $B_F(\epsilon)$ be the balls of radius $\epsilon > 0$ centered at S and F, respectively, and included in Ω. A schematic representation of Ω, $B_S(\epsilon)$ and $B_F(\epsilon)$ is given by Figure 1. Then, we consider the following steady configurations of the laminar incompressible Navier-Stokes equations [4, 10]:

$$\begin{cases} -\nabla \cdot (\eta(\nabla \mathbf{u} + (\nabla \mathbf{u})^\top) - p\mathbf{I}) + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = 0 & \text{in } \Omega \backslash \{B_S(\epsilon) \cup B_F(\epsilon)\}, \\ \nabla \cdot \rho\mathbf{u} = 0 & \text{in } \Omega \backslash \{B_S(\epsilon) \cup B_F(\epsilon)\}, \end{cases} \tag{1}$$

where $\mathbf{u}$ is the fluid flow velocity vector (m s^{-1}), p is the pressure field (Pa), η is the fluid dynamic viscosity (kg m^{-1} s^{-1}) and ρ is the fluid solution density (kg m^{-3}).

Furthermore, System (1) is completed with those boundary conditions:

- the effect of the walls on the fluid is modeled by considering

$$\mathbf{u} = 0 \tag{2}$$

on $\partial\Omega$,

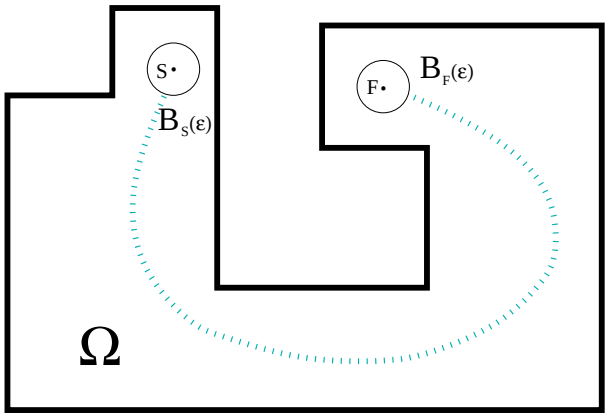

Figure 1: Schematic representation of the spatial domain Ω, $B_S(\epsilon)$ and $B_F(\epsilon)$ associated to the pathfinding problem presented in Section 2.1. The trajectory obtained by using the **BPA** is represented by a dotted line.

- the fluid entering through the boundary of $B_S(\epsilon)$ (called inlet boundary) is described by
$$\mathbf{u} = u_0, \tag{3}$$
on $\partial B_S(\epsilon)$, where $u_0 \in \mathbb{R}$ being the fluid injection velocity,
- the fluid exiting through the boundary of $B_F(\epsilon)$ (called outlet boundary) is characterized by
$$\Big(\eta\big(\nabla\mathbf{u} + (\nabla\mathbf{u})^\top\big) - p\mathbf{I}\Big)\mathbf{n} = 0, \tag{4}$$
on $\partial B_F(\epsilon)$, see [23].

We note that the boundary conditions imposed here to $B_S(\epsilon)$ and $\partial B_F(\epsilon)$ are different from the ones proposed in [37] (based on a difference of pressure between the starting and final points). This choice is motivated by the fact that they generally allow to obtain faster numerical results when using commercial CFD software, such as Comsol Multiphysics [39].

Once the solution of System (1) is computed, we determine a path linking S and F by considering one of the streamlines associated to $\mathbf{u}$ of shortest length starting from $\partial B_S(\epsilon)$ and ending at $\partial B_F(\epsilon)$ [4]. To determine such a streamline, we solve the following minimization problem:
$$\min_{x \in \partial B_S(\epsilon)} \int_{l_x} ds \tag{5}$$

where l_x is the streamline obtained by solving

$$\frac{dl_x(t)}{dt} = \mathbf{u}(l_x(t)), \tag{6}$$

with $l_x(0) = x$. Again, here, our approach is different from the ones proposed in [37, 40, 36] in which authors aim generating paths with restrictions (e.g., minimum ground friction, collision avoidance, etc.).

The process described above can be summarized by the following algorithm, called Basic Pathfinding Algorithm and denoted by **BPA**:

- **Step 1**- Let Ω, S and F be the spatial domain, the starting point and the final point, respectively.
- **Step 2**- Build $B_S(\epsilon) \subset \Omega$ and $B_F(\epsilon) \subset \Omega$, with $\epsilon > 0$ given.
- **Step 3**- Solve the fluid flow problem by considering Equations (1)-(4).
- **Step 4**- Solve the minimization Problem (5). The solution is denoted by xo.
- **Step 5**- Return l_{xo}.

Remark 1. *We note that, despite the fact that the authors in [46] have assumed the optimality of the routes generated with Equations* (1)-(4)*, the solution of Problem 5 does not always provide the shortest path between S and F [37]. This is due to the fact that the curvature and distribution of the streamlines associated to the Navier-Stokes velocity field depend on the pressure distribution of the fluid [18]. Thus, in function of the geometry of the domain boundary, the trajectories does not generally follow straight lines. Furthermore, due to apparition of secondary flows, not all the possible combinations of routes between S and F are explored (e.g., streamlines cannot cross each other [15]). This drawback is illustrated in Section 4.1.*

2.2. Incomplete spatial information

Here, we deal with the same pathfinding problem as the one introduced in Section 2.1. However, we now contemplate the case for which the spatial information between S and F is incomplete. For instance, we can consider a robot which is able to analyze its immediate spatial environment by using sensors and convert the collected information into a 2D or 3D geometrical

domain [8]. In this case, to solve the problem of determining a route between S and F, we propose to use the following try-and-error algorithm, called Dynamic Gathering Information Pathfinding Algorithm and denoted by **DyGIPA**:

- **Step 1**- Let $i = 1$, Ω_I, S and F be the known information on the spatial domain, the staring point and the final point, respectively.
- **Step 2**- Define Ω a compact spatial domain (e.g., a rectangle) that contains Ω_I, S and F. We consider $\mathbf{u} = 0$ on $\partial\Omega$.
- **Step 3**- Compute $\gamma_i \subset \Omega$, the solution of the **BPA**, defined in Section 2.1, obtained by considering Ω, S and F.
- **Step 4**- Follows γ_i. We have three possibilities:
  - **Step 4.1**- If when following γ_i a wall is crossed, we consider the point W which is at a safety distance $d_s \geq 0$ from this wall and run:
    - **Step 4.1.1**- Add the additional spatial information gathered during the trip from S to W to Ω_I.
    - **Step 4.1.2**- Limit γ_i only to the trajectory from S to W.
    - **Step 4.1.3**- Set $S = W$ and $i = i + 1$.
    - **Step 4.1.4**- Return to **Step 2**.
  - **Step 4.2**- If when following γ_i, we exit Ω_I at point W, we run **Steps 4.1.1** to **4.1.4**.
  - **Step 4.3**- If we reach F: go to **Step 5**.
  - **Step 5**- Return $\gamma = \bigcup_{k=1}^{i} \gamma_k$.

In Figure 2, we illustrate one particular run of the **DyGIPA**.

We note that, as shown in Section 4, obtaining streamlines is a quite computationally expensive process. Thus, we are interested in reducing the number of times that CFD equations are evaluated. To do so, **Step 4.2** can be omitted.

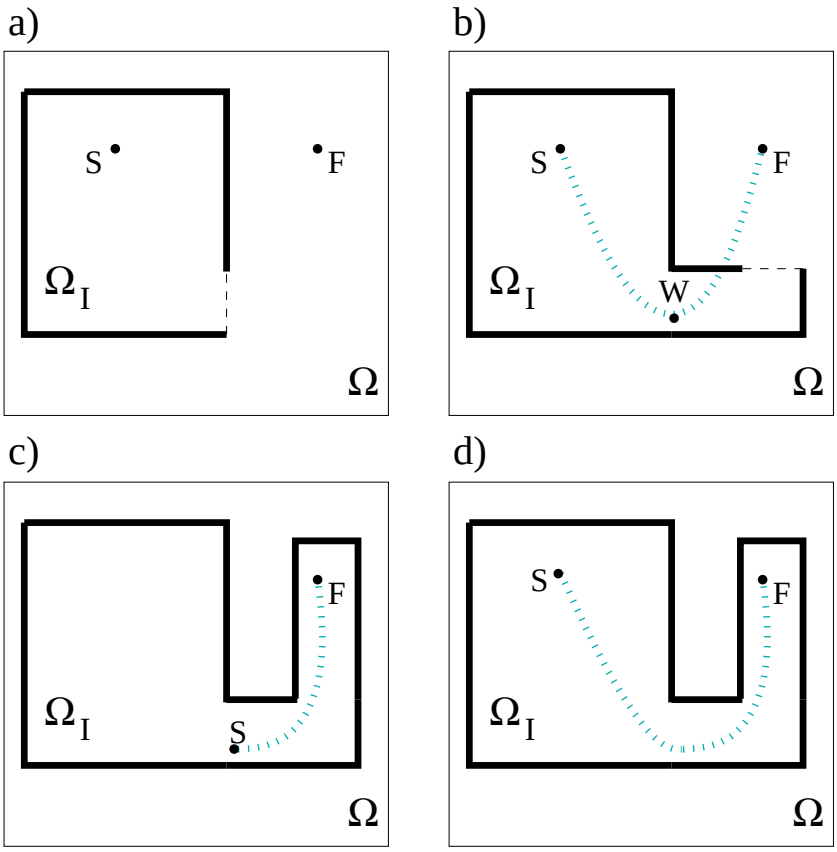

Figure 2: Schematic representation of one particular run of the **DyGIPA** described in Section 2.2: **a)** A rectangular domain Ω including S, W and Ω_I is built. **b)** the **BPA** defined in Section 2.1 is solved. The obtained trajectory (dotted line) left Ω_I at point W. We add the new spatial information to Ω_I. **c)** We set $S = W$ and solve again the **BPA**. The obtained trajectory reaches F. **d)** The algorithm returns the union of trajectories built during previous Steps **b)** and **c)**.

2.3. Dynamic spatial environments

Considering the pathfinding problem introduced in Section 2.1, we now introduce the case for which the spatial configuration of Ω vary in time. One way to solve such a problem could be to use the time-dependent versions of Equations (1)-(4) and techniques from the Fluid-Structure Interaction field to solve those equations for dynamic spatial domains [5]. However, this method is quite time expensive and should be applied only for continuous changes in the spatial configuration of Ω.

In this section, we focus on the particular context in which the modifications of Ω can be approximated by discrete events. More precisely, we assume that at time $t_n \in \mathbb{R}^+$, with $n \in \mathbb{N}$, $\Omega = \Omega_n$ such that S and $F \in \Omega_n$. In that case, the problem of finding a route from S to F can be solved by using the following try-and-error algorithm, called Dynamic Spatial Configuration Pathfinding Algorithm and denoted by **DySCoPA**:

- **Step 1**- Let $i = 1$, Ω_0, S and F be the initial spatial domain, the staring point and the final point, respectively. Set $\Omega = \Omega_0$.
- **Step 2**- Compute $\gamma_i \subset \Omega$, the solution of the **BPA**, defined in Section 2.1.
- **Step 3**- Follows γ_i. We have two sub-cases:
  - **Step 3.1**- If when following γ_i a wall is crossed, we consider the point W which is at a safety distance $d_s \geq 0$ from this wall and run:
    - **Step 3.1.1**- Determine the new spatial configuration $\Omega = \Omega_m$, with $m \in \mathbb{N}$.
    - **Step 3.1.2**- Limit γ_i only to the trajectory from S to W.
    - **Step 3.1.3**- Set $S = W$ and $i = i + 1$.
    - **Step 3.1.4**- Return to **Step 2**.
  - **Step 3.2**- If we reach point F: go to **Step 4**.
  - **Step 4**- Return $\gamma = \bigcup_{k=1}^{i} \gamma_k$.

In Figure 3, we illustrate one particular run of **DySCoPA**.

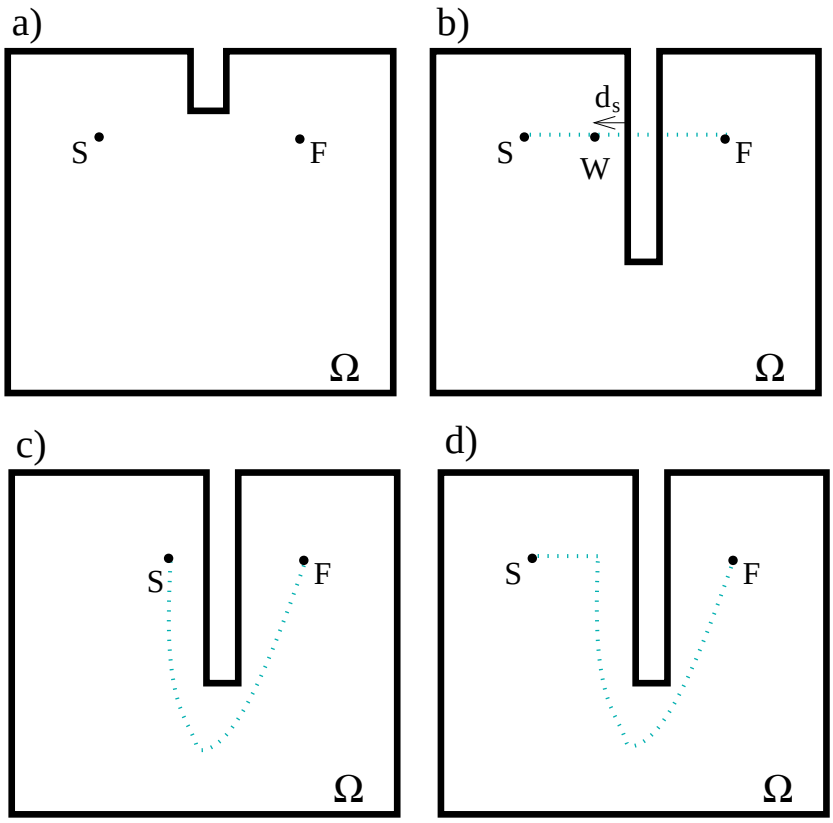

Figure 3: Schematic representation of one particular run of the **DySCoPA** described in Section 2.3: **a)** Initial problem. **b)** The **BPA**, defined in Section 2.1, is solved. The obtained trajectory (dotted line) cross a wall. Thus, we consider the point W which is at a safety distance d_s from this wall. We update the information of Ω. **c)** We set $S = W$ and solve the **BPA**. The obtained trajectory reaches F. **d)** The **DySCoPA** returns the union of trajectories built during previous Steps **b)** and **c)**.

Remark 2. *Here, we focus on the description of pathfinding methods easy to implement in Comsol Multiphysics. Thus,* ***DyGIPA*** *and* ***DySCoPA*** *are based on basic approaches of path replanning to prevent wall collision scenarios. They present some limitations. For instance,* ***DySCoPA*** *can be trapped in an infinite loop during* ***Step 3.1*** *in the case of periodic spatial environment reconfigurations. To avoid those drawbacks, both algorithms can be replaced by more efficient replanning methods in dynamic environments such as the one proposed in [9].*

2.4. One-way routes

Here, we add new restrictions to the basic pathfinding problem proposed previously by considering that some paths of the spatial domain Ω are of the type One-way route (i.e., the direction when using this path is fixed). This kind of restriction is used, for instance, in the design of road networks in city [51].

To model such a restriction, we propose to use a fan boundary condition (see, e.g., [30]) on interior boundaries added at the beginning and at the end of each One-way path, denoted by $\partial\Omega_{\text{fan}}$. For each One-way path, we assume that the normal vectors of $\partial\Omega_{\text{fan}}$ have the same direction than the one imposed to the path. More precisely, we consider the following equations on $\partial\Omega_{\text{fan}}$

$$\begin{cases} \Big[p - \mathbf{n}^\top[\eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^\top)]\mathbf{n} + \rho(\mathbf{u}\cdot\mathbf{n})^2\Big]^+ = f(p_f, v_f), \\ \Big[\rho\mathbf{u}\cdot\mathbf{n}\Big]^- = 0 \end{cases} \tag{7}$$

where $\mathbf{n}$ is the normal vector, $[\mathbf{v}]^+$ and $[\mathbf{v}]^-$ denote the positive and negative parts of a vector $\mathbf{v}$, respectively; f is the linear function modeling the fan static pressure evolution (Pa) and described in [49]; p_f is the fan static pressure at no flow (Pa); and v_f is the fan free delivery flow rate ($\text{m}^3.\ \text{s}^{-1}$).

Then, considering those additional boundary conditions, we solve the path-finding problem by using the **BPA** detailed previously. A schematic representation of this process is presented in Figure 4.

2.5. 3D spatial environments

Finally, we study the case of a 3D spatial domain Ω. Here, we consider the following two sub-problems:

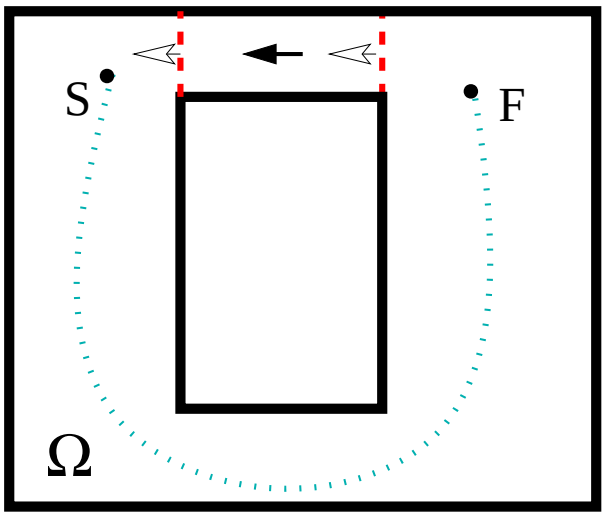

Figure 4: Schematic representation of the One-way routes pathfinding problem presented in Section 2.4. The direction of the One-way path is represented by a black arrow. $\partial\Omega_{\text{fan}}$ are described by dashed lines and their normal directions by white arrows. The trajectory obtained by using the **BPA**, defined in Section 2.1, is represented by a dotted line.

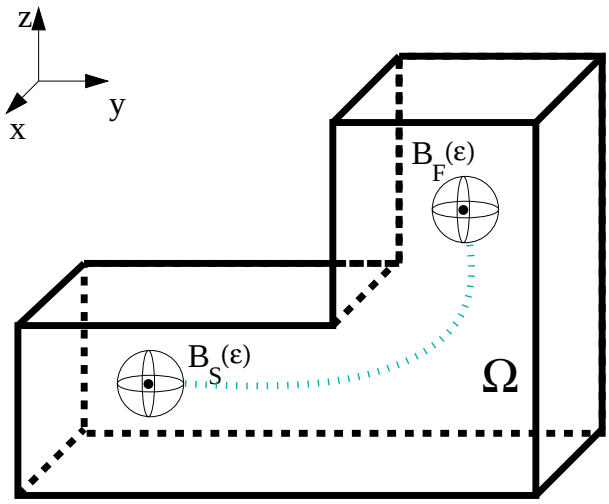

Figure 5: Schematic representation of the Problem **3DPN** presented in Section 2.5. The trajectory obtained by using the **BPA**, defined in Section 2.1, is represented by a dotted line.

The first one, denoted by **3DPN**, consists in finding a route from S to F without restriction on the height (i.e., the z-axis) of the path (e.g., when determining the trajectory of a nonholonomic flying vehicles). A particular example of such a case is depicted in Figure 5. In [44], a similar problem was solved by using a feedback approach. Here, we are interested in proposing a direct approach based on CFD equations. To do so, we can directly extend the methods proposed in Sections 2.1-2.4, by redefining $B_S(\epsilon)$ and $B_F(\epsilon)$ as spheres centered at S and F, respectively, of radius ϵ and included in Ω. Furthermore, we use the 3D versions of Equations (1)-(4) and (7). According to the pathfinding problem characteristics, the **BPA**, the **DyGIPA** or the **DySCoPA** described previously can be applied directly.

For the second sub-problem, denoted by **3DPR**, the height of the path between S and F must be equal to the height of the ground of Ω (e.g., when determining the trajectory of a ground vehicle). Additionally:

- If Ω does not include superposed floors (e.g., bridges or tunnels): the 3D problem can be reformulated as a 2D pathfinding problem by using projection techniques (see, e.g., [17]) and solved by using the algorithms presented in Sections 2.1-2.4 (but taking into account the height of the

ground when computing the length of the streamlines in Problem (5)). This case is denoted by **3DPR-1**.

- If Ω includes superposed floors: one possibility is to solve the Problem **3DPR-1** described above. However, when solving the minimization Problem (5), streamlines are orthogonally projected on the ground of Ω. Additionally, a process that avoid non-admissible streamlines (e.g., discontinuous trajectories such as the one presented in Figure 6), should be performed. This case is denoted by denoted by **3DPR-2**. A particular example representing this situation is shown in Figure 7.

3. Numerical implementation

In this section, we give some details about the numerical implementation of the models and algorithms described in Section 2.

The considered pathfinding problems are modeled by using Comsol Multiphysics 5.0 software (`www.comsol.com`, see [48]). More precisely, to compute a numerical solution of Equations (1)-(4) and (7), we use Finite Element Methods with Lagrange P2-P1 elements to stabilize the pressure and to satisfy the Ladyzhenskaya, Babouska and Brezzi stability condition. The 2nd-order Lagrange elements model the velocity component, while linear elements represent the pressure. A discrete version of the spatial domain Ω is obtained by applying a Delaunay mesh. The Navier-Stokes equations are solved by considering Galerkin Least Square streamline and crosswind diffusion methods in order to prevent numerical oscillations. Streamlines are estimated by using a Runge-Kutta method for solving (6). To reduce the apparition of secondary flows such as Dean vortices[14], parameters ρ, η and u_0 in Equation (1) are set such that the Reynolds number $Re = \rho u_s L/\eta$, where L (m) is a typical length of the domain Ω, is less than 1 [22, 28, 29]. Other model and solver parameters are fixed to their default values in Comsol [48]. A complete description of those techniques can be found in [20, 21].

The **BPA**, **DyGIPA** or **DySCoPA** are implemented in Matlab script (`www.mathworks.com`, see [24]). They use the outputs of the Comsol models detailed previously to compute an approximation of the solution of the studied pathfinding problem. More precisely, we apply the following techniques:

- The solution of the minimization Problem (5) is approximated by returning the shortest streamline starting from a discrete set of $N_{\text{bound}} =$

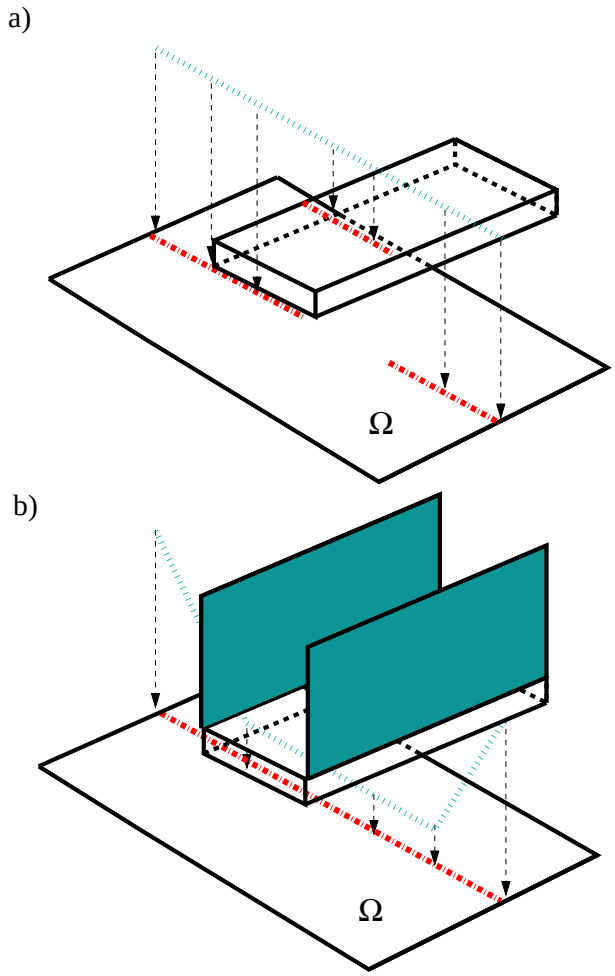

Figure 6: Example of a non-admissible trajectory for the Problem **3DPR-2**. Ω is compound by a floor and an upper bridge. **a)** With a dotted line, we represent a part of a trajectory obtained when solving the Problem **3DPR-1**. With a dash-dotted line, we represent the orthogonal projection on Ω of the previous trajectory. We observe a discontinuity when the trajectory enters and leaves the bridge. **b)** Wall boundary conditions (in solid gray) are added to the vertical borders of Ω. In this case, the orthogonal projection on Ω of the trajectory obtained when solving the Problem **3DPR-1** is admissible.

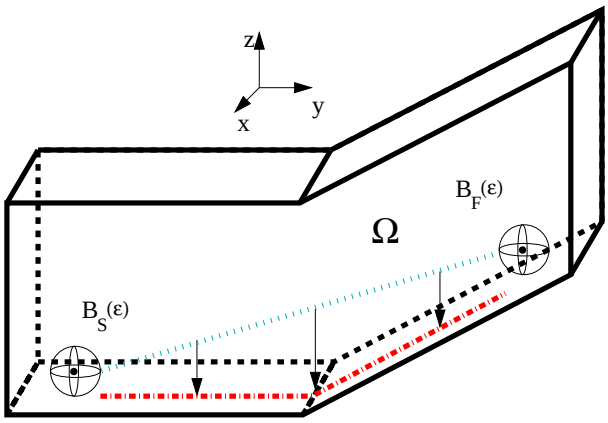

Figure 7: Schematic representation of the Problem **3DPR-2** presented in Section 2.5. The trajectory obtained by using the **BPA**, defined in Section 2.1, is represented by a dotted line and its orthogonal projection on the ground by a dash dotted line.

50 points uniformly distributed on $\partial B_S(\epsilon)$. Furthermore, integrals are estimated with a trapezoidal approximation formula.

- When using the **DyGIPA** or **DySCoPA**, we verify if, at time $t \in \mathbb{R}^+$, a vehicle with known kinetic characteristics and following γ_i is blocked by a wall of $\partial\Omega_t$, the spatial configuration of the domain at time t, by using the Matlab command *Inpolygon*, see:

  `www.mathworks.com/help/matlab/`

- The safety distance between the considered vehicle and a wall can be computed by using a ray tracing algorithm (see, e.g., [3]) in the tangent direction of the path.

- When solving a problem of the form **3DPR-2**, artificial wall boundary conditions are added to the vertical borders of Ω to avoid discontinuous trajectories. A particular example is given in Figure 6.

A file containing the **BPA** code used during **Example 1** (detailed below) can be downloaded at

`http://www.mat.ucm.es/~ivorra/BPA.zip`

4. Numerical Experiments

We now focus on the validation of the algorithms proposed previously by considering benchmark numerical experiments corresponding to each specific case detailed in Sections 2.1-2.5. Additionally, in Section 4.6, we propose a sensitivity analysis of the **BPA** according to the values of some of its parameters.

Those simulations are carried out in a 3.6Ghz Intel i7-4790 64bits computer with 32GB of RAM.

4.1. Basic problem

For this first experiment, denoted by **Example 1**, we consider a 2D maze Ω, depicted in Figure 8-**a**, a starting point $S = (7, 109)$ (m) and a final point $F = (10, 125)$ (m). We set $\epsilon = 1$ (m). The mesh associated to Ω is presented in Figure 8-**b** and is composed by N_{mesh} =4538 elements. The fluid coefficients are fixed to those of the water, which are $\rho = 1010$ (kg m^{-3}) and $\eta = 9.8e-4$ (kg m^{-1} s^{-1}). To obtain a Reynolds number Re lower than 1 and considering $L = 11$ (m), which is the length of the shortest wall of $\partial\Omega$, we use u_0=8e-8 (m s^{-1}). The impact on the solutions returned by the **BPA** of the values of those parameters is discussed in Section 4.6.

In order to compare the efficiency of our approach with another pathfinding method, **Example 1** is also solved by considering the basic **A*** algorithm implemented in Matlab script available at [41]. For this algorithm, we consider 4791 points, obtained by building the maze geometry with Comsol and converting it to a black and white image of size 100x100 pixels, to represent the spatial domain. Other algorithm parameters are set to their default values.

Remark 3. *The particular **A*** considered here is only presented as a reference algorithm and cannot be considered as a fast pathfinding method. Thus, comparisons with more efficient pathfinding algorithms, such as the **D*** algorithm [33], can be recovered from the literature (see, e.g., [7, 45]) and we can deduce that our methodology is not the fastest one. This result should be expected as, in general, CFD numerical implementations are generally computationally intensive [39]. However, one of the objectives of this article, is not to propose computationally cheap pathfinding methods, but easy-to-implement algorithms based on the use of Comsol Multiphysics.*

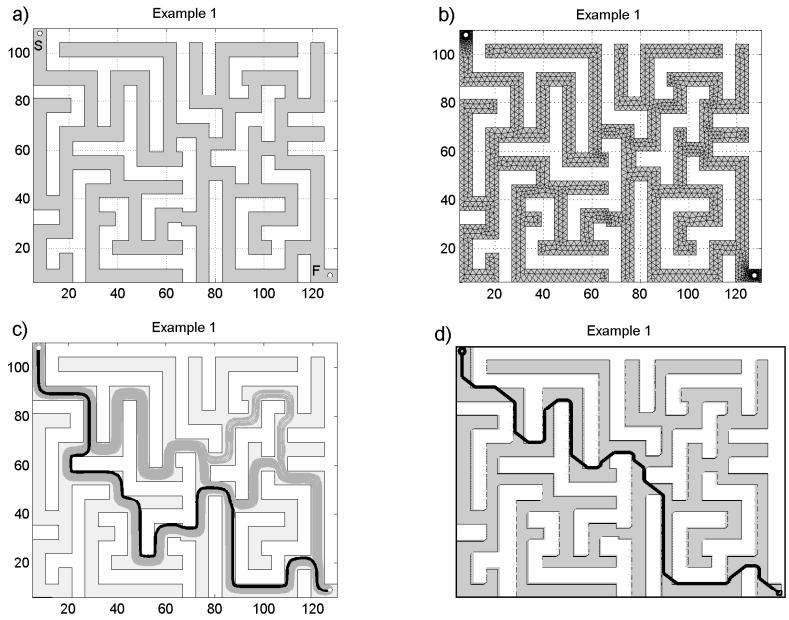

Figure 8: **Example 1**: **a)** Spatial domain Ω and position of $B_S(\epsilon)$ and $B_F(\epsilon)$. **b)** Mesh associated to Ω. **c)** Solution (black line) and streamlines (gray line) returned by the **BPA**. **d)** Solution returned by the considered **A*** algorithm.

For this particular problem, both **BPA** and **A*** require around 3.5 (s). For the **BSA**, this computational time is distributed on the following way: 0.6 (s) to build Ω and its mesh, 2.4 (s) to compute the solution of the model and 0.5 (s) to compute and return the best trajectory. For the **A***, we obtain: 0.8 (s) to build the points representing Ω and 2.7 (s) to estimate the trajectory. This result tends to show that, when considering similar numbers of mesh elements and points for representing Ω in the **BPA** and the **A***, respectively, both algorithms require a comparable amount of computational time to return the solution. In addition, we point out the fact that in [46] the authors obtain the numerical solution of a similar problem with a finite difference approach in 10 (s) (however, in this particular reference, the used computer is not described).

Solutions found by the **BPA** and the **A*** are depicted in Figures 8-**c** and **d**. Both methods success in finding a route between points S and F. The length of the best trajectories returned by the **BPA** and the **A*** are 284.5m and 267.4m, respectively. Thus, for this particular example, the **BPA** provides a path 6.4% larger than the one given by the **A***. Indeed, due to the reasons explained in Remark 1, the **BPA** streamlines do not explores all the possible combinations of routes and, thus, the shortest path may be a non-admissible path for this algorithm.

However, as we can observe in Figure 8-**c**, one advantage of using the **BPA** instead of the **A*** is that, considering the whole set of streamlines, we can directly obtain alternative routes between S and F without requiring additional computation. This could be useful, for instance, in case of problems for which different trajectories should be returned (e.g., healthcare exploration devices [31], moving a swarm of vehicles [36, 40]). Another interest is that the union of all the streamlines corresponds to all possible direct routes (i.e., avoiding useless detour) between S and F. Thus, this information could be used to reduce the complexity of the pathfinding problem (by removing Dead-End routes). For example, we can consider the **BPA*** hybrid algorithm which consists in: (i) Run the **BPA**, (ii) plot all computed streamlines and convert them into a bitmap image and (iii) solve the **A*** by using the previous image as the spatial domain. Applying the **BPA*** to **Example 1** with the same parameters as used before, we obtain the solution in 5.7 (s): 3.5 (s) for running Steps (i) and (ii) and 1.5 (s) for Step (iii). The set of streamlines computed by the **BPA** and the solution returned by the **A*** are depicted in Figures 9-**a** and **b**, respectively. By applying this method, the spatial domain for the **A*** is reduced to 1976 points. The **BPA*** should be

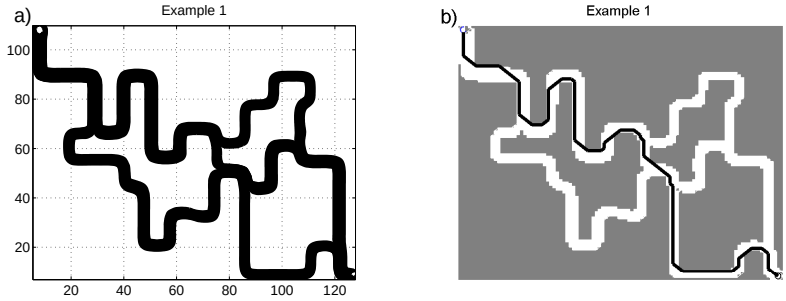

Figure 9: **Example 1**: **a)** Streamlines (black) returned by the **BPA**. **b)** Solution returned by the considered **BPA*** algorithm.

used for problems for which the **A*** requires high computational times.

From those previous results, the **BPA** seems to be a reasonable alternative to the considered **A***.

A file containing the **BPA** code used during this experiment can be downloaded at

`http://www.mat.ucm.es/~ivorra/BPA.zip`

4.2. Incomplete spatial information

For this experiment, denoted by **Example 2**, we consider a vehicle moving at a constant velocity of 0.5 (m.s^{-1}) in a 2D plane and which is able to obtain visual data from its spatial environment. A safety distance $d_s = 1$ (m) between the vehicle and crossed walls is used. The aim of this vehicle is to move from $S = (30, 150)$ (m) to $F = (150, 30)$ (m). The full spatial domain Ω is depicted in Figure 10-**a**. On the same figure we also present the solution returned by the **BPA** in 3.5 (s) with the parameters from Section 4.1.

However, at time $t_0 = 0$ (s), only an incomplete information of the spatial domain is known. It is denoted by Ω_0 and is presented in Figure 10-**b**. Starting from this space we run the **DyGIPA**, without **Step 4.2** to accelerate the computational time, with the same parameters as the ones used in Section 4.1. A graphical representation of each **DyGIPA** step is depicted in Figures 10-**b** to **e**. More precisely:

- At Step 1, solved in 3.4 (s), the algorithm determines a first trajectory which crosses a wall of $\partial\Omega$. The vehicle reaches the safety distance d_s

from this wall at point $W0 = (98.5, 39.5)$ and at time $t_1 = 348$ (s). We set $S = W0$. The new spatial information gathered by the vehicle during this first trip is added to the pathfinding problem and we now consider $\Omega = \Omega_1$, shown in Figure 10-**c**, as the spatial domain.

- At Step 2, solved in 3.7 (s), the algorithm returns a trajectory which is blocked by a wall of $\partial\Omega$. The vehicles attains the safety distance at point $W1 = (120.5, 81.5)$ and at time $t_2 = 466$ (s). We set $S = W1$ and consider a new spatial domain $\Omega = \Omega_2$ (see Figure 10-**d**).

- At Step 3, solved in 3.8 (s), the algorithm generates a trajectory which is blocked by a wall of $\partial\Omega$. The vehicles attains the safety distance at point $W2 = (203, 64.5)$ and at time $t_2 = 466$ (s). We set $S = W2$ and consider a new spatial domain $\Omega = \Omega_3$ (see Figure 10-**e**).

- At Step 4, solved in 3.8 (s), the algorithm determines a trajectory that reaches F at time $t_3 = 998$ (s). The algorithm finally returns the union of all trajectories found previously (see Figure 10-**f**).

The final path returned by the **DyGIPA** is generated in 14.7 (s) and measures 499 (m). For this particular problem, we observe that this solution is shorter than the one returned by the **BPA** starting from the complete spatial information, which has a length of 522 (m) and is obtained in 4.2 (s). This can be explained by the high curvatures of the **BPA** streamlines produced by the sharp boundaries of Ω [18]. Additionally, we also solve the problem starting from the complete spatial information with the **A*** and considering the parameters presented in Section 4.1. The obtained path, depicted in Figure 10-**a**, has a distance of 381 (m) and is returned after 3.8 (s). Thus, for this example, the solution returned by the **BPA** is 37% larger than the one given by the **A***.

This result tends to show that using the **DyGIPA** helps generate suitable solutions for problems with incomplete information of the spatial domain.

4.3. Dynamic spatial environments

To illustrate the interest of the **DySCoPA**, we consider the case of a spatial environment which varies in time. This experiment is denoted by **Example 3**. At time $t = 0$ (s), $\Omega = \Omega_0$. Then, each 300 (s), the spatial configuration alternates between Ω_1 and Ω_0 (e.g., simulating the fact that two doors are periodically opened and closed at the same time), both depicted in

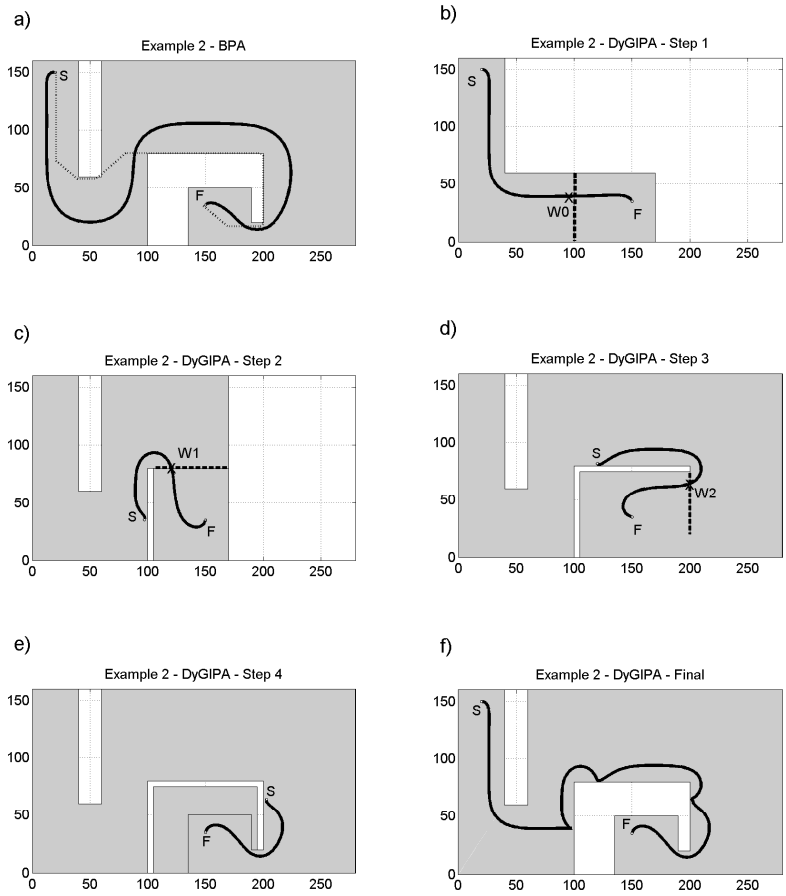

Figure 10: **Example 2**: **a)** Spatial domain Ω, position of $B_S(\epsilon)$ and $B_F(\epsilon)$ and solutions returned by the **BPA** (black continuous line) and the A* (black dotted line) **b)** Domain Ω_0, position of $B_S(\epsilon)$, $B_F(\epsilon)$ and $W0$ (X) and solution (black line) returned by the **DyGIPA** at Step 1. In dashed line, we represent the wall of $\partial\Omega$ that blocks the trajectory. **c)** Domain Ω_1, position of $B_S(\epsilon)$, $B_F(\epsilon)$ and $W1$ and solution returned by the **DyGIPA** at Step 2. **d)** Domain Ω_2, position of $B_S(\epsilon)$, $B_F(\epsilon)$ and $W2$ and solution returned by the **DyGIPA** at Step 3. **e)** Domain Ω_3, position of $B_S(\epsilon)$ and $B_F(\epsilon)$ and solution returned by the **DyGIPA** at Step 4. **f)** Domain Ω, position of $B_S(\epsilon)$ and $B_F(\epsilon)$ and final solution returned by the **DyGIPA**.

Figures 11-**a** and **b**. Considering a vehicle moving at a constant velocity of 0.5 (m.s^{-1}) and $d_s = 1$ (m), we want to determine a path from $S = (65, 160)$ (m) to $F = (220, 145)$ (m).

First, assuming an invariant spatial domain Ω_0 or Ω_1, the pathfinding problem is solved with the **BPA**, using the values of the parameters given in Section 4.1. Both solutions are found in 3.7 (s) and are presented in Figures 11-**a** and **b**. For Ω_0, the length of the path is 329 (m) whereas, for Ω_1, it is 498 (m). Both problems are also solved by considering the **A*** with the parameters described in Section 4.1. The returned paths, depicted in Figure 11-**a** and -**b**, measure 294 (m) and 496 (m) when considering Ω_0 and Ω_1, respectively. They are obtained in 2.1 (s) and 2.8 (s), respectively. For those examples, the **BPA** generates paths 12% and 0.4% larger than the ones given by the **A***, respectively.

Starting from Ω_0 at time 0 (s), we run the **DySCoPA** with the parameters introduced in Section 4.1. A graphical representation of each **DySCoPA** step is depicted in Figures 11-**c** and **d**. More precisely:

- At Step 1, solved in 3.4 (s), the algorithm returns a trajectory which is blocked by a wall of $\partial\Omega_1$. The vehicle reaches the safety distance d_s from this wall at point $W0 = (127, 119)$ and at time $t_1 = 404$ (s) (see Figure 11-**c**). We set $S = W0$ and $\Omega = \Omega_1$.

- At Step 2, solved in 3.4 (s), the algorithm determines a trajectory that reaches F at time $t_3 = 758$ (s). The algorithm finally returns the union of all trajectories found previously (see Figure 11-**e**).

The final route returned by the **DySCoPA** is generated in 6.8 (s), measures 379 (m) and is depicted in Figure 11-**e**.

This example seems to indicate that the **DySCoPA** is a reasonable tool for solving problems with dynamic environments.

4.4. One-way routes

Here, we consider the same maze as the one introduced in Section 4.1. The starting point is $S = (7, 109)$ (m) and final point is $F = (10, 125)$ (m). We impose the direction of 3 routes of the maze. The spatial domain and the One-way routes are presented in Figure 12-**a**. We set $p_f = 100$ (Pa) and $v_f = 1e - 5$ (m^3. s^{-1}). This case is denoted by **Example 4**.

The **BPA** is solved with the parameters defined in Section 4.1. The final solution is returned in around 8.4 (s), has a length of 292 (m) and is depicted

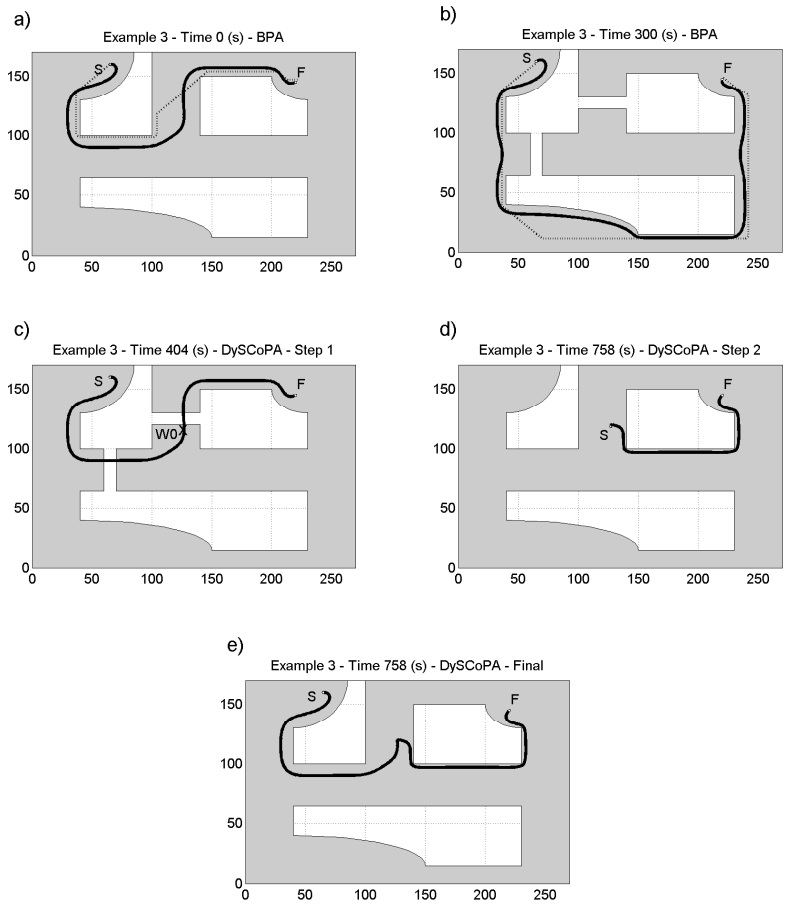

Figure 11: **Example 3**: **a)** Spatial domain Ω_0, position of $B_S(\epsilon)$ and $B_F(\epsilon)$ and solutions returned by the **BPA** (black continuous line) and the A* (black dotted line) considering Ω_0. **b)** Spatial domain Ω_1, position of $B_S(\epsilon)$ and $B_F(\epsilon)$ and solutions returned by the **BPA** (black continuous line) and the A* (black dotted line) considering Ω_1. **c)** Domain Ω_1, position of $B_S(\epsilon)$, $B_F(\epsilon)$ and $W0$ (X) and solution (black line) returned by the **DySCoPA** at Step 1. **d)** Domain Ω_0, position of $B_S(\epsilon)$ and $B_F(\epsilon)$ and solution returned by the **DySCoPA** at Step 2. **e)** Domain Ω_0, position of $B_S(\epsilon)$ and $B_F(\epsilon)$ and final solution returned by the **DySCoPA**.

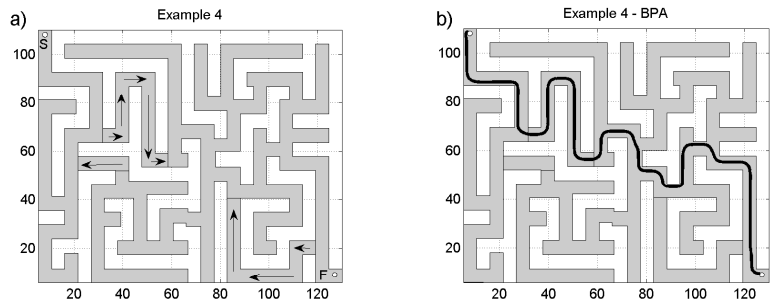

Figure 12: **Example 4**: **a)** Spatial domain Ω, position of $B_S(\epsilon)$ and $B_F(\epsilon)$ and direction of the One-way routes (arrows). **b)** Solution (black line) returned by the **BPA**.

in Figure 12-**b**. We can observe on this figure that the obtained trajectory respects the directions imposed to the One-way routes.

This result tends to show the efficiency of modeling One-way routes by using Equation (7). However, in this case, the computational time of the **BPA** is increased by more than twice.

4.5. 3D spatial environments

Finally, we focus on the application of the **BPA** to 3D pathfinding problems of the type **3DPN** and **3DPR-2** (as, problems **3DPR-1** are equivalent to 2D problems).

First, we study the case of a nonholonomic flying vehicle moving into the 3D space Ω, depicted in Figure 13-**a**, from $S = (0.5, 0.5, 0.5)$ (m) to $F = (2.5, 0.5, 4.5)$ (m). This case is denoted by **Example 5**. We consider the same parameters as the ones introduced in Section 4.1, except ϵ=0.1 (m). The 3D mesh is compound by N_{mesh} =10361 elements (see Figure 13-**b**). In this case, the **BPA** needs around 5.8 (s) to solve the pathfinding problem. Obtained streamlines are shown in Figure 13-**c**. The length of the best path found by the **BPA** is 11 (m) and is depicted in Figure 13-**d**. Those results seems to indicate that the **BPA** proposes a computationally affordable method to solve 3D problems of the form **3DPN**.

Now, we study the case of a ground vehicle moving into the 3D space Ω presented in Figures 13-**a** to **d** from $S = (0.5, 4.7, 0.2)$ (m) to $F = (6.5, 4.5, 2.2)$ (m). We can observe that, due to the presence of superposed floors, this problem cannot be easily simplified to a 2D domain. For this particular case, one can build a bijective application from the ground of Ω

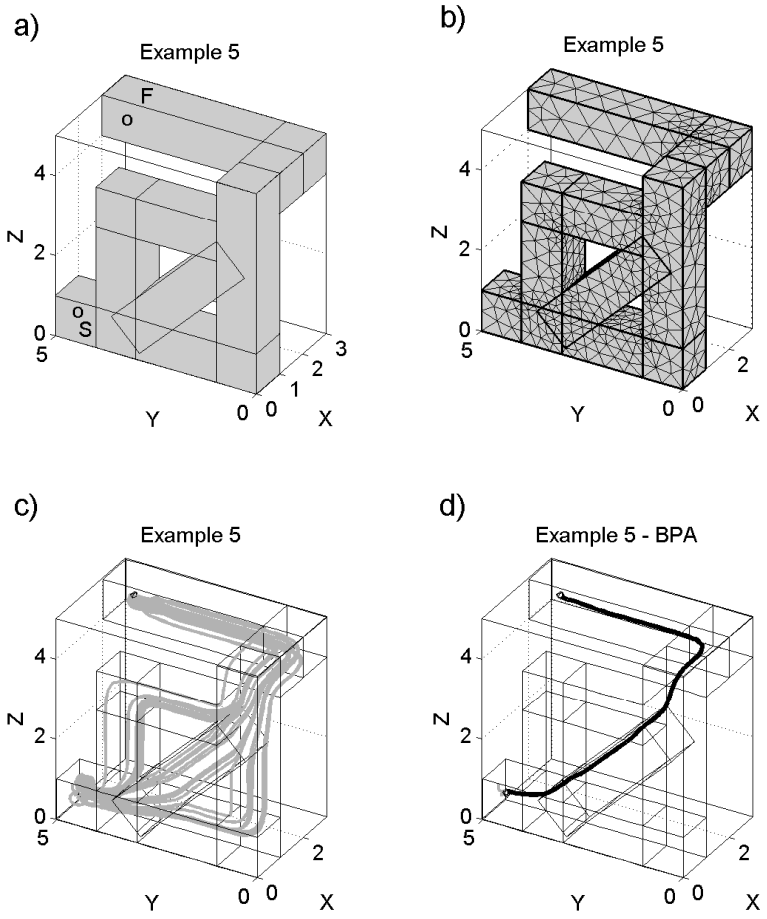

Figure 13: **Example 5**: **a)** Spatial domain Ω and position of $B_S(\epsilon)$ and $B_F(\epsilon)$. **b)** Mesh associated to Ω. **c)** Streamlines (gray line) returned by the **BPA**. **d)** Solution (black line) returned by the **BPA**.

to a 2D domain, but this process cannot be trivially automatized for any 3D domain and is time consuming. Thus, here, we work directly with the 3D domain Ω. In order to avoid paths crossing the vertical borders of $\partial\Omega$, we add to Ω internal walls to those borders. This modified spatial domain is denoted by Ω_m and is reported in Figure 13-**e**. Then, we run the **BPA** with the parameters introduced before and project the obtained streamlines on the ground of Ω_m (see Figure 13-**f**). The final solution is returned in around 17.2 (s) and is also shown in Figure 13-**f**. Thus, for 3D problems of the form **3DPR-2**, the proposed approach generates reasonable solutions.

*4.6. Sensitivity analysis of the **BPA***

As said in Section 2.1, the **BPA** requires the values of various parameters to compute the solution of the considered pathfinding problem.

Here, we are interested in studying the impact of changes in some of those parameters on the **BPA** outputs. In particular, we focus on the following parameters: the Reynolds number R_e (obtained by varying, for instance, u_0); the radius of the inlet ($\partial B_S(\epsilon)$) and outlet ($\partial B_F(\epsilon)$) boundaries ϵ; and the number of elements of the mesh N_{mesh}. Additionally, we analyze the effect of considering a Stokes flow in the model (i.e., we omit the term $\rho(\mathbf{u}\cdot\nabla)\mathbf{u}$ in System (1)) instead of the full Navier-Stokes equations.

To do so, we consider the following problems: **Example 1**, described in Section 4.1; **Example 2**, described in Section 4.2, with the whole domain information and denoted by **Example 2$_{\mathbf{wi}}$**; **Example 3**, described in Section 4.3, with the fixed domain Ω_0 and denoted by **Example 3**$_{\Omega_0}$; and **Example 3**, described in Section 4.3, with the fixed domain Ω_1 and denoted by **Example 3**$_{\Omega_1}$.

Each problem is solved with the **BPA** and considering the following sets of parameters:

- **Set**$_{\text{Init}}$: $\epsilon = 1$ (m), $R_e = 0.9$ (i.e., $u_0 = 8e{-}8$ (m.s^{-1})) and $N_{\text{mesh}} \approx 4500$ elements. System (1) is considered to model the fluid flow. Those values correspond to the ones used in Section 4.1.

- **Set**$_{\epsilon=a}$: $\epsilon = a$, with a=0.1 (m), 2 (m) and 10 (m) (this last value is not admissible when solving **Example 1**), and the other parameters values are taken from **Set**$_{\text{init}}$.

- **Set**$_{u_0=b}$: $u_0 = b$, with $b = 8e-9$ (m.s^{-1}), $b = 8e-7$ (m.s^{-1}) and $b = 8e-6$ (m.s^{-1}), corresponding to $R_e = 0.09, 9$ and 90, respectively. The values of the other parameters are set to the ones of **Set**$_{\text{init}}$.

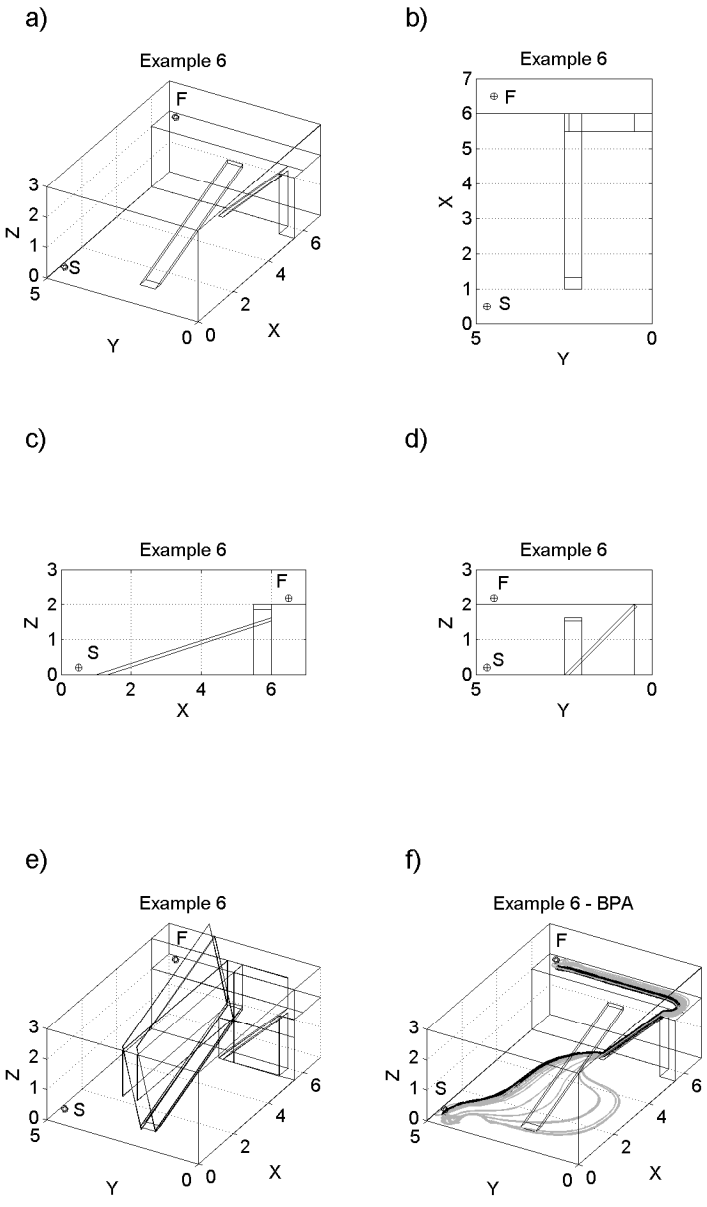

Figure 14: **Example 6**: **a)** Spatial domain Ω and position of $B_S(\epsilon)$ and $B_F(\epsilon)$. **b)** X-Y view of Ω. **c)** X-Z view of Ω. **d)** Y-Z view of Ω **e)** Spatial domain Ω_m. **f)** Streamlines (gray line) and solution (black line) returned by the **BPA** and projected on the ground of Ω_m.

- $\mathbf{Set}_{N_{\text{mesh}}=c}$: $N_{\text{mesh}} = c$, with c=2500 elements and 7000 elements, and the other parameters values from $\mathbf{Set}_{\text{init}}$.
- $\mathbf{Set}_{\text{Stokes}}$: We consider the parameters values from $\mathbf{Set}_{\text{init}}$ and a Stokes flow.

For each case, we compute the length in (m) of the returned path, denoted by **lrp**, and the computational time in (s) required to find this solution, denoted by **cti**. Furthermore, we determine the number of streamlines starting from the inlet boundary $\partial B_S(\epsilon)$ and reaching the outlet boundary $\partial B_F(\epsilon)$, denoted by **nsl**. Indeed, due to numerical errors, some streamlines approximated numerically do not reach the final destination.

The obtained results are reported on Table 1.

	Example 1			Example $\mathbf{2_{wi}}$		
Set of parameters	**lrp** (m)	**cti** (s)	**nsl**	**lrp** (m)	**cti** (s)	**nsl**
$\mathbf{Set}_{\text{Init}}$	285	3.9	43	522	3.6	47
$\mathbf{Set}_{\epsilon=0.1}$	286	4.0	45	523	3.8	47
$\mathbf{Set}_{\epsilon=2}$	284	3.9	16	520	3.7	48
$\mathbf{Set}_{\epsilon=10}$	-	-	-	510	3.7	47
$\mathbf{Set}_{u_0=8e-9}$	285	3.9	44	522	3.4	47
$\mathbf{Set}_{u_0=8e-7}$	285	3.9	40	522	3.9	47
$\mathbf{Set}_{u_0=8e-6}$	292	4	12	548	4.0	45
$\mathbf{Set}_{N_{\text{mesh}}=2500}$	284	3.7	15	521	3.2	45
$\mathbf{Set}_{N_{\text{mesh}}=7000}$	286	4	45	522	4.1	49
$\mathbf{Set}_{\text{Stokes}}$	285	3.7	44	522	3.5	48
	Example $\mathbf{3}_{\Omega_0}$			**Example** $\mathbf{3}_{\Omega_1}$		
Set of parameters	**lrp** (m)	**cti** (s)	**nsl**	**lrp** (m)	**cti** (s)	**nsl**
$\mathbf{Set}_{\text{Init}}$	329	3.5	49	498	3.8	47
$\mathbf{Set}_{\epsilon=0.1}$	330	4.0	48	499	3.9	47
$\mathbf{Set}_{\epsilon=2}$	328	3.5	48	497	3.9	48
$\mathbf{Set}_{\epsilon=10}$	360	3.6	28	504	3.9	38
$\mathbf{Set}_{u_0=8e-9}$	329	3.8	49	499	3.7	47
$\mathbf{Set}_{u_0=8e-7}$	332	3.9	47	499	4.0	47
$\mathbf{Set}_{u_0=8e-6}$	350	3.9	47	524	4.1	47
$\mathbf{Set}_{N_{\text{mesh}}=2500}$	328	3.6	45	497	3.3	45
$\mathbf{Set}_{N_{\text{mesh}}=7000}$	329	4.1	49	499	4.1	49
$\mathbf{Set}_{\text{Stokes}}$	329	3.3	49	499	3.7	47

Table 1: Results obtained by the **BPA** when considering different sets of parameters (**Set of parameters**) and the cases **Example 1**, **Example** $\mathbf{2_{wi}}$, **Example** $\mathbf{3}_{\Omega_0}$ and **Example** $\mathbf{3}_{\Omega_1}$: length of the returned path in (m) (**lrp**); computational time in (s) (**cti**); and number of streamlines starting from the inlet boundary and reaching the outlet boundary (**nsl**).

We observe on this table that for values of ϵ lower than 10 (m) (i.e., $\mathbf{Set}_{\text{Init}}$, $\mathbf{Set}_{\epsilon=0.1}$ and $\mathbf{Set}_{\epsilon=2}$), the outputs returned by the **BPA** are equivalent between each other (i.e., we obtain similar values of **lrp**, **cti** and **nsl**). However, when $\epsilon = 10$ (m) the inlet and outlet boundaries are situated at a close distance of $\partial\Omega$ making difficult to evaluate correctly some streamlines (e.g., the value of **nsl** decreases up to 28 and 38 for **Example** $\mathbf{3}_{\Omega_0}$ and

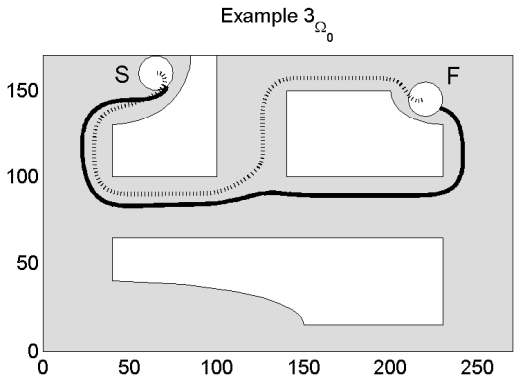

Figure 15: Solutions returned by the **BPA** when considering the case **Example 3**$_{\Omega_0}$ and the sets of parameters $\mathbf{Set}_{\text{Init}}$ (black dotted line) and $\mathbf{Set}_{\epsilon=10}$ (black continuous line). The position of $B_S(\epsilon)$ and $B_F(\epsilon)$ are also reported.

Example 3$_{\Omega_1}$, respectively). A similar phenomenon occurs for **Example 1** with $\mathbf{Set}_{\epsilon=2}$, as the width of the pipes composing the spatial domain is thinner than in the other examples. As a consequence, the solution of **Example 3**$_{\Omega_0}$ is significantly different from the one returned with $\mathbf{Set}_{\text{Init}}$ (both solutions are presented in Figure 15) and its length is 360 (m).

Focusing on the value of the Reynolds number R_e (or, equivalently, the value of u_0), if $R_e \leq 9$ (i.e., $u_0 \leq 8e-7$ (m.s^{-1})), no noticeable changes in the outputs are observed. However, when $R_e = 90$ (i.e., $u_0 = 8e-6$ (m.s^{-1})) the apparition of vorticity in the fluid flow produces an increase of the length of the returned paths between 2% and 10% with respect to the case $\mathbf{Set}_{\text{Init}}$. Those solutions are shown in Figure 16.

Regarding the value of N_{mesh}, if the mesh is coarse (i.e., $N_{\text{mesh}} = 2500$ elements), due to the lack of quality in the numerical approximation of System (1), the number of streamlines starting from the inlet and reaching the outlet, **nsl**, is drastically reduced. However, in the considered examples this does not impact the final solution. Building a finer mesh (i.e., $N_{\text{mesh}} = 7000$ elements) does not improve the results in a significant way.

The use of a Stokes flow produces similar results than considering the full Navier-Stokes equations. Furthermore, the computational time is reduced by around 5% when working with the Comsol implementation.

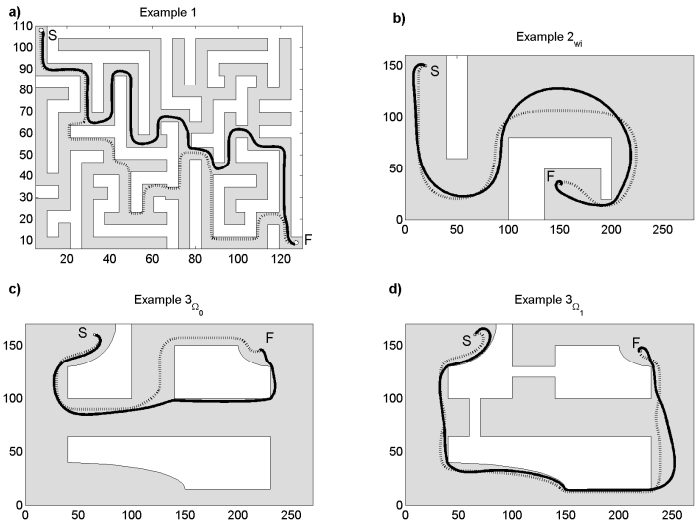

Figure 16: Solutions returned by the **BPA** when considering the sets of parameters $\mathbf{Set}_{\text{Init}}$ (black dotted line) and $\mathbf{Set}_{u_0=8e-6}$ (black continuous line) and the cases: **a) Example 1**, **b) Example 2$_{\mathbf{wi}}$**, **c) Example 3$_{\Omega_0}$** (**c**) and **d) Example 3$_{\Omega_1}$**. The position of $B_S(\epsilon)$ and $B_F(\epsilon)$ are also reported.

Those results seem to indicate that, in order to not deteriorate the efficiency of the **BPA**, the Reynolds number R_e (or equivalently the value of u_0) should be lower than 9 and the radius ϵ should be chosen to generate the inlet and outlet boundaries far enough from $\partial\Omega$. Additionally, the mesh should be fine enough (here, at least 4500 elements) in order to obtain a reasonable amount of streamlines for studying several alternative routes from the starting point to the final point. Moreover, the use of a Stokes flow seems to produce similar results than the ones obtained with a Navier-Stokes model, for a slightly reduced computational time.

Finally, we are also interested in studying the effect on the solutions of changes in the geometry of Ω. In particular, we focus on the impact of narrowing some parts of the spatial domain on the fluid flow. To this aim, we consider two experiments, denoted by **Example 7-1** and **Example 7-2**, whose spatial domains, initial points and final points are depicted in Figure 17. We observe that the only difference between both problems is that one pipe of the spatial domain (near the outlet) is narrowed. We solve those problems by considering the **BPA** and the set of parameters $\mathbf{Set}_{\epsilon=0.1}$. On the one hand, **Example 7-1** is solved in 3.4 (s) and the length of the returned path is 2.0 (m). On the other hand, **Example 7-2** is solved in 3.3 (s) and the solution measures is 2.2 (m). The returned paths and the computed streamlines are presented in Figure 17. We observe that those solutions present some differences. Furthermore, the configurations of streamlines have been modified in the area near the narrowed pipe. Those changes in the velocity field can be in part explained by the evolution of the pressure distribution in the narrowed zones (see, e.g., the Bernoulli's principle detailed in [4]). Those results show the importance of the spatial domain when using CFD techniques for determining paths. Indeed, in some cases, the spatial domain could be modified to avoid that the vehicle be close from areas of danger (e.g., near a cliff) and a particular attention should be paid when narrowing parts of this domain.

5. Conclusions

In this article, we have applied the idea of using models coming from the Computational Fluid Dynamics field to solve pathfinding problems with spatial constraints. Those problems may include: incomplete spatial information; dynamical evolution of the environment; One-way routes; and 3D environments for ground or nonholonomic flying vehicles.

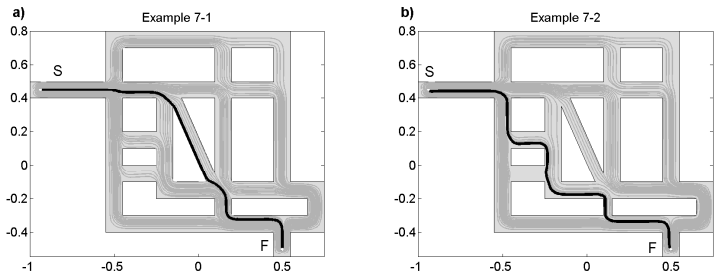

Figure 17: Solutions (black line) and streamlines (gray lines) returned by the **BPA** when considering the set of parameters $\mathbf{Set}_{\epsilon=0.1}$ and the cases: **a) Example 7-1** and **b) Example 7-2**. The position of $B_S(\epsilon)$ and $B_F(\epsilon)$ are also reported.

For solving those problems, we have detailed three specific algorithms: the **BPA**, the **DyGiPA** and the **DySCoPA**. Those algorithms allow obtaining paths between a starting and a final points.

Then, we have proposed a particular implementation of this approach by using Comsol multiphysics and Matlab software.

Next, we have studied the interest our method by considering several benchmark cases including all the difficulties detailed previously. We have observed that the paths returned by our algorithms are not the shortest ones. For instance, in the considered examples, the **BPA** has generated paths larger than the ones given by the **A***, in a range from 0.4% to 37% . However, those paths correspond to reasonable solutions of the considered problems. Furthermore, the computational times required by our algorithms for solving those test cases are similar to the ones needed by a simple **A*** algorithm, but are greater then the ones produced by efficient pathfinding algorithms found in the literature [7, 45, 33]. However, one of the objectives of this work is to propose easy-to-implement pathfinding algorithms in Comsol Multyphysics, a software known to be computationally expensive when solving CFD equations.

All those results tend to show that our methodology helps to quickly implement algorithms to solve complex pathfinding problems, for which the resolution computational time is not primordial.

Finally, we have performed a sensitivity analysis of the **BPA** regarding some of its key parameters. We have observed that, in order to avoid a deterioration of the **BPA** performances in the considered numerical examples,

attention should be paid when choosing the value of the Reynolds number, the radius of the inlet and outlet boundaries and the number of elements of the mesh and when narrowing some parts of the spatial domain. Additionally, the use of a Stokes flow seems to reduce the computational time and seems to produce similar results than the ones obtained with the Navier-Stokes equations.

In future works, we will focus on the application of incomplete models (see [26, 28]) or other type of PDEs to reduce the complexity of those algorithms.

A file containing the **BPA** code used during **Example 1** can be downloaded at

`http://www.mat.ucm.es/~ivorra/BPA.zip`

Acknowledgements

This work was carried out thanks to the financial support of the Spanish "Ministry of Economy and Competitiveness" under projects MTM2011-22658 and MTM2015-64865-P; the "Junta de Andalucía" and the European Regional Development Fund through the project P12-TIC301; and the research group MOMAT (Ref. 910480) supported by "Banco de Santander" and "Universidad Complutense de Madrid". The author would like to thank Angel M. Ramos del Olmo and Tatiana Diaz Jimenez for their valuable help during this work.

[1] Waves in isotropic totalistic cellular automata: Application to real-time robot navigation. Advances in Complex Systems **19**(4), 1650,012–18 (2016). DOI 10.1142/S0219525916500120

[2] Amutha, B., Ponnavaikko, M.: Location update accuracy in human tracking system using zigbee modules. International Journal of Computer Science and Information Security **6**(2), 322 – 331 (2009)

[3] Arvo, J., Kirk, D.: Fast ray tracing by ray classification. SIGGRAPH Comput. Graph. **21**(4), 55–64 (1987)

[4] Batchelor, G.: An Introduction to Fluid Dynamics. Cambridge University Press (2000). URL `http://dx.doi.org/10.1017/CBO9780511800955`. Cambridge Books Online

[5] Bathe, K.: Computational Fluid and Solid Mechanics. Elsevier Science (2001). URL https://books.google.es/books?id=Id06Z4YMJLMC

[6] Bretti, G., Natalini, R., Piccoli, B.: A fluid-dynamic traffic model on road networks. Archives of Computational Methods in Engineering **14**(2), 139–172 (2007). DOI 10.1007/s11831-007-9004-8. URL http://dx.doi.org/10.1007/s11831-007-9004-8

[7] Burns, E.A., Hatem, M., Leighton, M.J., Ruml, W.: Implementing fast heuristic search code. In: D. Borrajo, A. Felner, R.E. Korf, M. Likhachev, C.L. Lpez, W. Ruml, N.R. Sturtevant (eds.) SOCS. AAAI Press (2012)

[8] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Lydia, W., Kavraki, E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementation. Intelligent Robotics and Autonomous Agents series. MIT Press (2005)

[9] Chrpa, L., Novak, P.: Dynamic trajectory replanning for unmanned aircrafts supporting tactical missions in urban environments. Holonic and Multi-Agent Systems for Manufacturing. Springer (2011)

[10] Ciarlet, P., Lions, J.: Handbook of Numerical Analysis: Numerical methods for fluids (pt. 3). Handbook of Numerical Analysis. North-Holland (1990). URL https://books.google.es/books?id=S0Hqp3vOVxkC

[11] Connolly, C., Burns, J., Weiss, R.: Path planning using laplace's equation. In: Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on, vol. 3, pp. 2102–2106 (1990)

[12] Connor, D.: Integrating Planning and Control for Constrained Dynamical Systems. PhD., University of Pennsylvania (2007)

[13] Daniel, K., Nash, A., Koenig, S., A., F.: Theta*: Any-angle path planning on grids. Journal of Artificial Intelligence Research **39**, 533 – 579 (2010)

[14] Dean, W.: Lxxii. the stream-line motion of fluid in a curved pipe (second paper). The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **5**(30), 673–695 (1928). DOI 10.1080/14786440408564513

[15] Dickmann, D.: On the Near Field Mean Flow Structure of Transverse Jets Issuing Into a Supersonic Freestream. University of Texas at Arlington (2007). URL https://books.google.es/books?id=4ee-g96_F5gC

[16] Dijkstra, E.: A Short Introduction to the Art of Programming. Techn. Hogeschool (1971)

[17] Eberly, D.: 3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics. CRC Press (2006)

[18] Fay, J.: Introduction to Fluid Mechanics. MIT Press (1994). URL https://books.google.es/books?id=XGVpue4954wC

[19] Fuerstman, M., Deschatelets, P., Kane, R., Schwartz, A., Kenis, P., Deutch, J., , Whitesides, G.: Solving mazes using microfluidic networks. Langmuir **19**(11), 4714–4722 (2003). DOI 10.1021/la030054x

[20] Girod, B., Greiner, G., Niemann, H.: Principles of 3D Image Analysis and Synthesis. The Springer International Series in Engineering and Computer Science. Springer US (2013). URL https://books.google.es/books?id=jVHuBwAAQBAJ

[21] Glowinski, R., Neittaanmäki, P.: Partial Differential Equations: Modelling and Numerical Simulation. Computational Methods in Applied Sciences. Springer Netherlands (2008). URL https://books.google.es/books?id=xKhfyc0Nf54C

[22] Hertzog, D., Ivorra, B., Mohammadi, B., Bakajin, O., , Santiago, J.: Optimization of a microfluidic mixer for studying protein folding kinetics. Analytical Chemistry **78**(13), 4299–4306 (2006). DOI 10.1021/ac051903j

[23] Heywood, J.G., Rannacher, R., Turek, S.: Artificial boundaries and flux and pressure conditions for the incompressible navierstokes equations. International Journal for Numerical Methods in Fluids **22**(5), 325–352 (1996)

[24] Hunt, B., Lipsman, R., Rosenberg, J.: A Guide to MATLAB: For Beginners and Experienced Users. Cambridge University Press (2001). URL https://books.google.es/books?id=XhQBx9LJKIAC

[25] Hysing, J., Turek, S.: Evaluation of commercial and academic cfd codes for a two-phase flow benchmark test case. International Journal of Computational Science and Engineering **10**(4), 387–394 (2015)

[26] Infante, J.A., Ivorra, B., Ramos, A., Rey, J.: On the modelling and simulation of high pressure processes and inactivation of enzymes in food engineering. Mathematical Models and Methods in Applied Sciences **19**(12), 2203–2229 (2009). DOI 10.1142/S0218202509004091. URL http://www.worldscientific.com/doi/abs/10.1142/S0218202509004091

[27] Isebe, D., Azerad, P., Bouchette, F., Ivorra, B., Mohammadi, B.: Shape optimization of geotextile tubes for sandy beach protection. International Journal for Numerical Methods in Engineering **74**(8), 1262–1277 (2008). DOI 10.1002/nme.2209. URL http://dx.doi.org/10.1002/nme.2209

[28] Ivorra, B., Hertzog, D., Mohammadi, B., Santiago, J.: Semi-deterministic and genetic algorithms for global optimization of microfluidic protein-folding devices. International Journal for Numerical Methods in Engineering **66**(2), 319–333 (2006). DOI 10.1002/nme.1562. URL http://dx.doi.org/10.1002/nme.1562

[29] Ivorra, B., Redondo, J., Santiago, J., Ortigosa, P., Ramos, A.: Two- and three-dimensional modeling and optimization applied to the design of a fast hydrodynamic focusing microfluidic mixer for protein folding. Physics of Fluids **25**(3), 032001 (2013). DOI http://dx.doi.org/10.1063/1.4793612. URL http://scitation.aip.org/content/aip/journal/pof2/25/3/10.1063/1.4793612

[30] Johnson, R.: Handbook of Fluid Dynamics. Handbook Series for Mechanical Engineering. Taylor & Francis (1998)

[31] Katevas, N.: Mobile Robotics in Healthcare. Assistive technology research series. IOS Press (2001). URL https://books.google.es/books?id=jT__IKy9wTgC

[32] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research **5**(1), 90–98 (1986)

[33] Koenig, S., Likhachev, M.: D*lite. In: Eighteenth National Conference on Artificial Intelligence, pp. 476–483. American Association for Artificial Intelligence (2002)

[34] Kwon, H.J.: Use of comsol simulation for undergraduate fluid dynamics course. In: 2012 ASEE Annual Conference & Exposition, San Antonio, Texas. https://peer.asee.org/22167 (2012)

[35] Lee, V., Law, M., Wee, S.: Theory to practice on finite element method and computational fluid dynamics tools. Australasian Journal of Engineering Education **22**(2), 123–133 (2015)

[36] Lolla, S.: Path Planning in Time Dependent Flows using Level Set Methods. PhD., University of Massachusetts Institute Of Technology (2012)

[37] Louste, C., Liegeois, A.: Near optimal robust path planning for mobile robots: the viscous fluid method with friction. Journal of Intelligent and Robotic Systems **27**(1), 99–112 (2000)

[38] Nau, D., Kumar, V., Kanal, L.: General branch and bound, and its relation to A* and AO*. Artificial Intelligence **23**(1), 29 – 58 (1984)

[39] Pepper, D., Wang, X.: Benchmarking COMSOL Multiphysics 3.5a CFD problems. Proceeding of the Cosmol Conference 2009, Boston. Comsol Inc. (2009)

[40] Pimenta, L., Michael, N., Mesquita, R., Pereira, G., Kumar, V.: Control of swarms based on hydrodynamic models. In: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pp. 1948–1953 (2008)

[41] Premakumar, P.: A* (A star) search for path planning tutorial. Matlab Central. URL: `http://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/26248/versions/3/download/zip` (2010)

[42] Ramos Del Olmo, A.: Introducción al análisis matemático del método de elementos finitos. Editorial Complutense (2013)

[43] Rimon, E., Koditschek, D.: Exact robot navigation using artificial potential functions. Robotics and Automation, IEEE Transactions on **8**(5), 501–518 (1992)

[44] Roussos, G., Dimarogonas, D.V., Kyriakopoulos, K.J.: 3d navigation and collision avoidance for nonholonomic aircraft-like vehicles. International Journal of Adaptive Control and Signal Processing **24**(10), 900–920 (2010). DOI 10.1002/acs.1199. URL http://dx.doi.org/10.1002/acs.1199

[45] Sun, X., Yeoh, W., Uras, T., Koenig, S.: Incremental ara*: An incremental anytime search algorithm for moving-target search. In: International Conference on Automated Planning and Scheduling (2012)

[46] Suzuno, K., Ueyama, D., Branicki, M., Tth, R., Braun, A., Lagzi, I.: Maze solving using fatty acid chemistry. Langmuir **30**(31), 9251–9255 (2014). DOI 10.1021/la5018467

[47] Szab, C., Sobota, B.: Path-finding algorithm application for route-searching in different areas of computer graphics. In: New Frontiers in Graph Theor. InTech (2012)

[48] Tabatabaian, M.: Comsol 5 for Engineers. Multiphysics Modeling Series. Mercury Learning & Information (2015). URL https://books.google.es/books?id=twhSrgEACAAJ

[49] Twizell, E., Bright, N.: Numerical modelling of fan performance. Applied Mathematical Modelling **5**(4), 246 – 250 (1981). DOI http://dx.doi.org/10.1016/S0307-904X(81)80074-1. URL http://www.sciencedirect.com/science/article/pii/S0307904X81800741

[50] Villacorta-Atienza, J., Calvo, C., Makarov, V.: Prediction-for-compaction: navigation in social environments using generalized cognitive maps. Biological Cybernetics **109**(3), 307–320 (2015). DOI 10.1007/s00422-015-0644-8. URL http://dx.doi.org/10.1007/s00422-015-0644-8

[51] Wang, J., Deng, W.: Optimizing capacity of signalized road network with reversible lanes. Transport **First Online**, 1–11 (2015)

[52] Wu, X., Zhang, S.: The study and application of artificial intelligence pathfinding algorithm in game domain. In: Computer Science and Service System (CSSS), 2011 International Conference on, pp. 3772–3774. IEEE (2011). DOI 10.1109/CSSS.2011.5974547

[53] Zeng, W., Church, R.L.: Finding shortest paths on real road networks: The case for A*. Int. J. Geogr. Inf. Sci. **23**(4), 531–543 (2009). DOI 10.1080/13658810801949850. URL http://dx.doi.org/10.1080/13658810801949850