

# Homework 1

## Exercises around the k-coloring problem

Computational Complexity (CC-MIRI)

Duc DANG VU

Teacher: Albert ATSERIAS



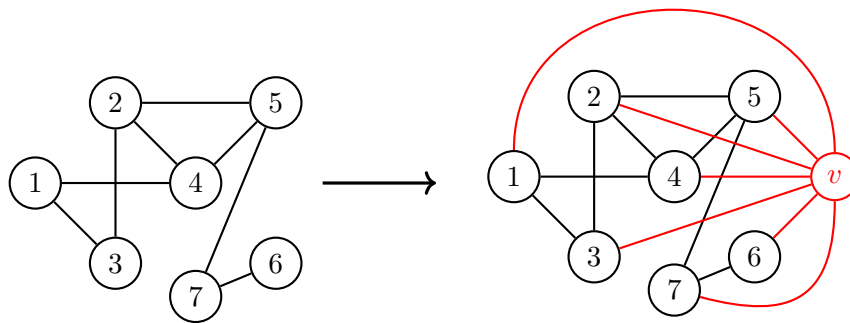
UPC-FIB 2023

## 1 $k$ -COL vs $(k+1)$ -COL

*Statement: For every integer  $k \geq 1$ , give a direct proof (one that does not use any other results like Cook-Levin theorem) that  $k$ -COL  $\leq^p (k+1)$ -COL*

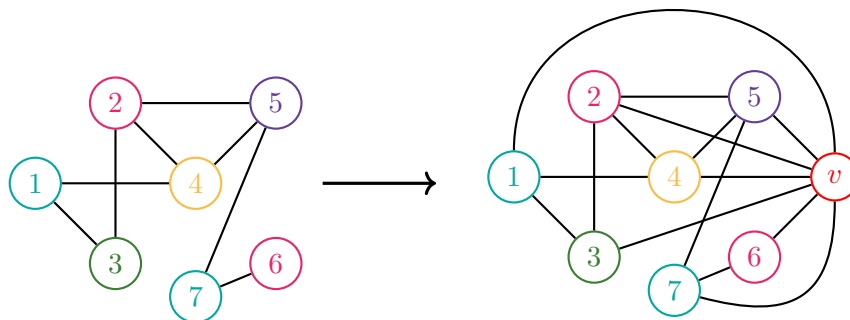
To show that  $k$ -COL is polynomial time reducible to the  $(k+1)$ -coloring problem, we need to find a polynomial time algorithm that takes an instance of  $k$ -COL as input and outputs an instance of  $(k+1)$ -COL. A valid  $k$ -coloring of one of these instances has to be valid if and only if the other instance is valid.

Let  $G$ , a graph with  $n$  vertices, be an instance of  $k$ -COL. We construct an instance of  $(k+1)$ -COL. To do so, we add a new vertex  $v$  to the graph and we connect it to all vertices of  $G$ . Let  $G'$  be the resulting graph, with  $n+1$  vertices.



The example above illustrates the construction of a graph by this algorithm. This construction takes  $O(n)$  time because we have to connect the new vertex to each vertex of  $G$ , thus we have to travel along the entire list of vertices of  $G$ , which takes a time  $n$ .

Let's now assume that the graph  $G$  has a valid  $k$ -coloring. We then color the vertex  $v$  with a new color which is not used in the  $k$ -coloring of  $G$ . We can claim that  $v$  has no neighbours with the same color since  $v$  is connected to every vertex of  $G$ , and has no color used in  $G$ . Knowing that  $G$  has a valid  $k$ -coloring, we can conclude that  $G'$  has a valid  $(k+1)$ -coloring.



In the example above,  $v$  has no neighbours of the same color, thus the second graph has a valid  $(k + 1)$ -coloring.

Let's now assume that  $G'$  has a valid  $(k + 1)$ -coloring. We can remove  $v$  and its edges from  $G'$ , and obtain a new graph  $G$  with a valid  $k$ -coloring. This follows because no two adjacent vertices in  $G'$  have the same color, and  $v$  has a unique color in  $G'$  ( $v$  is connected to every other vertex so it must have a unique color).

The construction of  $G'$  in polynomial time and the fact that we showed the equivalence between the  $k$ -coloring of  $G$  and the  $(k + 1)$ -coloring of  $G'$  prove that  $\underline{k\text{-COL} \leq^p (k + 1)\text{-COL}}$ .

## 2 3-COL

*Statement:* Look up a proof that 3-COL is NP-complete in the literature and reproduce it here in your own words. State your reference. For full credit, try to find the oldest reference for this.

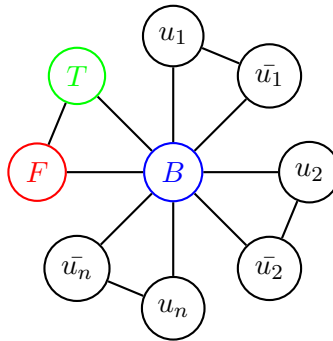
We will take as a reference the proof provided by Larry Stockmayer in March 1973. This paper, named *Planar 3-colorability is polynomial complete*, is based on Karp's work on graphs. Here is the URL of this paper: <https://dl.acm.org/doi/pdf/10.1145/1008293.1008294>.

To show that 3-COL is NP-complete, we first need to show that it is NP. This is the case, because in order to verify that a graph  $G = (V, E)$  with  $n$  vertices satisfies 3-COL, we have to go through the list of all vertices and check whether or not their neighbors are colored with a different color than the vertex studied. This check takes  $O(n^2)$  time, thus 3-COL is NP.

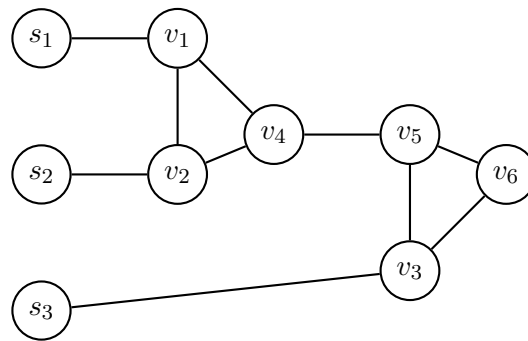
Now we need to show that 3-COL is NP-hard. To do so, we have to give a polynomial-time reduction from 3-SAT to 3-COL. In other words, given an instance  $\Phi$  of 3-SAT, we need to construct a graph  $G$  that satisfies 3-COL if and only if  $\Phi$  is satisfiable.

Let  $\Phi$  be a 3-SAT instance (a 3-CNF formula), with  $C_1, C_2, \dots, C_m$  the clauses of  $\Phi$ . Let  $x_1, x_2, \dots, x_n$  be the variables of  $\Phi$ .

The first step to build the graph  $G$  is to construct a triangle where we assign values  $T$ ,  $F$  and  $B$  to each vertex. These values correspond to "True", "False" and "Base". They correspond to the three colors of the graph. Then, for each variable  $x_i$ , we create a triangle  $(B, u_i, \bar{u}_i)$ . These triangles will represent each value of  $x_i$  and its negation  $\bar{x}_i$  (True for one and False for the other one). We see that if  $u_i$  is colored True, then  $\bar{u}_i$  has to be colored False and vice-versa, which is what we are looking for. This first step should look like this:

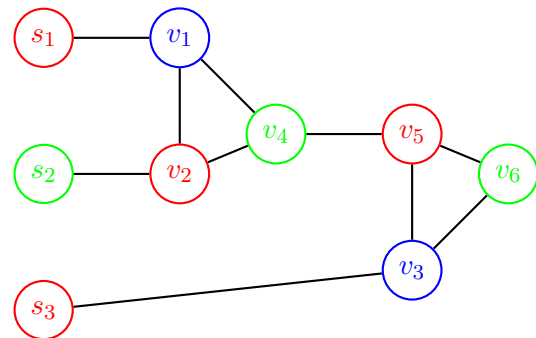
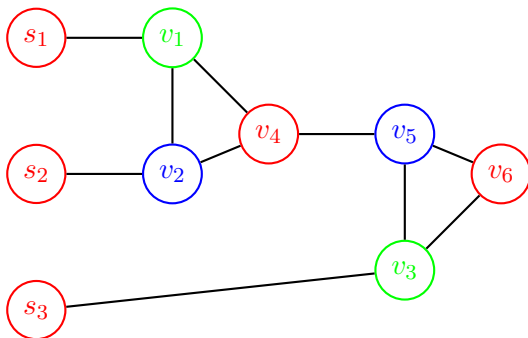


The second step consists in building a structure that will correspond to each clause of  $\Phi$ . In order to do so, Larry Stockmayer constructed the following structure, that we will call  $S$  for the rest of the proof:



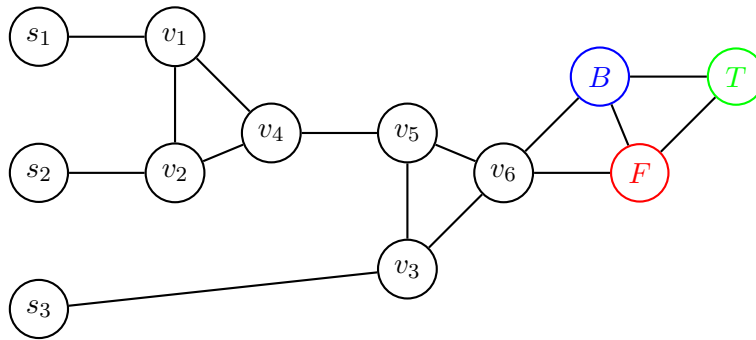
This structure  $S$  contains six vertices denoted by  $v_i$  and three entries  $s_i$ , which correspond to the three literals of a clause. In order to have a valid 3-coloring,  $S$  has to verify these 2 properties:

- If  $s_1$ ,  $s_2$  and  $s_3$  have the same color  $c$ , then  $v_6$  has to be colored with the color  $c$ . In other words, if the three clauses are  $F$  (respectively  $T$ ), then  $v_6$  has to be  $F$  (respectively  $T$ ). This is what we expect from a clause in a 3-CNF formula.
- If  $s_1$ ,  $s_2$  and  $s_3$  have colors where at least one of them is different of the two others, then there exist a valid 3-coloring of  $S$  where  $v_6$  is colored with  $T$ . In other words, if at least one literal is assigned with  $T$ , then we can assign  $v_6$  with  $T$ , which is what we expect from a clause in a 3-CNF formula.



These two colored structures illustrate the two properties: they have a valid 3-coloring and the output  $v_6$  correspond to the truth assignment of a clause with three literals.

The final step of the construction of  $G$  consists in linking all structures  $S$  (there are as many as clauses) to the vertices labelled  $B$  and  $F$  of the first triangle:



We will have  $m$  structures  $S$  linked to the base vertex. For each structure, we will link the vertices  $u_i$  (defined above) corresponding to each literals of one clause as entries of these structures.

Now that we have constructed the graph  $G$ , we now have to prove that the 3-SAT instance  $\Phi$  is satisfiable if and only if the graph  $G$  is 3-colorable.

Let's assume that  $\Phi$  is satisfiable. Let  $(y_1, y_2, y_3, \dots, y_n)$  be the satisfying assignment. If  $y_i$  is True, then we color the corresponding vertex  $u_i$  with  $T$  and the vertex  $\bar{u}_i$  with  $F$ . If  $y_i$  is False, then we will do the opposite. This coloring is valid knowing that  $u_i$  and  $\bar{u}_i$  are adjacent to the Base vertex. Moreover, all output vertices of each structures (the vertex  $v_6$ ) can be colored as  $T$ . Indeed,  $\Phi$  is satisfiable, that means that for all structures  $S$ , at least one entry vertex  $s_i$  is colored  $T$ , thus the output vertex  $v_6$  can be colored  $T$  by the second property. Knowing that each output vertex is connected to a  $B$  vertex and a  $F$  vertex, we can claim that  $G$  has a valid 3-coloring.

Let's now assume that  $G$  is 3-colorable. We construct the satisfying assignment of  $\Phi$  by setting each literal  $x_i$  to True if the corresponding vertex  $u_i$  is colored with  $T$ , and vice-versa. We suppose that  $\Phi$  is not satisfiable. That means that at least one clause has to be False, thus all literals of such a clause have to be False. Therefore, the structure  $S$  associated to such a clause has an output vertex  $v_6$  colored with  $F$ , by the first property and knowing that all three entries are colored with  $F$ . Thus, a vertex colored with  $F$  would be adjacent to another vertex colored with  $F$  by construction: this is impossible because  $G$  is 3-colorable. Therefore,  $\Phi$  is satisfiable.

The construction of the graph  $G$  takes a polynomial time, and we just gave a reduction from

3-SAT. We can now claim that 3-COL is NP-hard. Knowing that 3-COL is also NP, we can conclude that **3-COL is NP-complete.**

### 3 Self-reducibility

*Statement: Show that, for every integer  $k \geq 1$ , if the decision version  $k$ -COL of the  $k$ -coloring problem can be solved in polynomial time, then its search version SEARCH- $k$ -COL can also be solved in polynomial time, which means that there is a polynomial time algorithm that given a graph  $G$  outputs a valid  $k$ -coloring of it in the form of a list of pairs (**vertex**, **color**). Clearly state the main idea of your algorithm at the beginning of your answer.*

The idea to solve this problem is to explore each vertex  $v$  of a given graph  $G$  and to assign to each one a color  $c$  not used by one of its neighbours. Then, we check with the decision algorithm if  $G$  can have a valid  $k$ -coloring, knowing that the vertex  $v$  is colored with color  $c$ . If this is not the case, we color it with another color not used by one of its neighbours. By repeating this process with all vertices, we will find a valid  $k$ -coloring of the graph.

But how can we assign a color to each vertex so that the decision algorithm can understand which color we decide to assign at each step? The idea is to construct a new graph  $G'$  by adding to  $G$   $k$  vertices, which will correspond to the  $k$  colors of  $k$ -COL. We will denote this new graph  $C$  (as "colors") such as:  $C = (V_C, E_C) = (\{c_1, c_2, \dots, c_k\}, E_C)$ . We easily see that if we connect a vertex  $v$  to every  $c_i$  except one named  $c_j$ , then  $v$  has to be colored  $j$ . This trick will allow us to check with the decision algorithm if a given coloring of a vertex can lead to a valid coloring of the graph.

Note that in order for this idea to work, we have to make the subgraph  $C$  complete. In other words, we have to make every pair of vertices in  $V_C$  connected by an edge. This ensure that every  $c_i$  must have a different color. Otherwise, every  $c_i$  could have the same color, which is not what we want.

Now that we explained this idea, we now construct an algorithm to solve SEARCH- $k$ -COL using the decision algorithm  $k$ -COL:

Given a graph  $G$  and an integer  $k \geq 1$ :

1. Run  $k$ -COL on  $G$ . If it outputs NO, output "The graph cannot be  $k$ -colored"
2. If it outputs YES, add a complete graph  $C$  to  $G$  with  $k$  vertices, denoted  $c_1, c_2, \dots, c_k$ . The resulting graph is  $G' = (V_{G'}, E_{G'})$ .
3. Initialize a list **coloring** that will contain the valid  $k$ -coloring of  $G$  and a list **colored** that will contain the vertices of  $G$  already colored.

4. Pick a vertex  $v$  which is not in **colored**. For each neighbour  $u$  of  $v$ , if  $u$  is in **colored**, then there exists a color  $i$  such as  $(u, c_i) \notin E_{G'}$ . Add the edge  $(v, c_i)$  in  $E_{G'}$ . This means that  $v$  cannot be colored as one of its neighbours.
5. For each remaining colors  $j$ , add all the edges  $(v, c_l)$  (where  $l$  is not  $j$ ) in  $E_{G'}$  and run the  $k$ -COL algorithm on  $G'$ . If it outputs **NO**, remove the edges  $(u, c_l)$  and repeat this step for the next color  $j$ . This step tries every possible color for  $v$  and check if this color  $j$  can induce a valid  $k$ -coloring of  $G'$ . If it outputs **YES**, add the pair  $(v, j)$  to **coloring** and add  $v$  to **colored**. Repeat step 4. and 5.
6. When every vertex have been colored, output **coloring**.

This algorithm outputs a list of pairs that gives a valid  $k$ -coloring of  $G$ , because at each step we ensure that no vertex is colored as one of its neighbours, and the color assigned has to be valid thanks to the use of the decision algorithm  $k$ -COL.

We now determinate the complexity of this algorithm. The decision algorithm is solved in polynomial time, so let's assume that  $k$ -COL is solved in  $O(n^c)$  time, where  $c$  is a constant and  $n$  is the number of vertices of  $G$ . Steps 1, 3 and 6 are done in  $O(1)$  time. Step 2 creates a complete graph with  $k$  vertices, so it will be done in  $O(|V_C| + |E_C|) = O(k^2)$ . Step 4 is done in time  $O(k)$  because each vertex has at most  $k$  neighbours. In step 5, we call the decision algorithm at most  $k$  times, and the creation of the edges at each call is  $O(k)$ . That means that step 5 is done in  $O(k + kn^c) = O(kn^c)$ .

Finally, we can say that the algorithm we just built is done in  $O(kn^c)$  time, where  $k$  and  $c$  are constants. We can conclude that **SEARCH- $k$ -COL can be solved in polynomial time.**

# Compte-rendu TP Pollution

MIE : Approche variationnelle et Éléments finis

Cours : Guillaume Chiavassa

Auteur : Duc DANG VU



Centrale Marseille  
2022-2023



# Table des matières

Introduction . . . . .	2
Création du code . . . . .	2
Création et maillage du conduit . . . . .	2
Modélisation du liquide circulant dans le conduit . . . . .	2
Modélisation de la circulation de l'agent polluant . . . . .	3
Résultat final . . . . .	4
Images de la simulation numérique . . . . .	4
Observations et interprétations . . . . .	5
Conclusion : où doit se cacher le poisson ? . . . . .	5

## Introduction

L'objectif de ce TP est de modéliser la diffusion d'un agent polluant au sein d'un conduit, dans lequel s'écoule un liquide. Nous utiliserons le logiciel *FreeFem* ainsi que les éléments théoriques appris en cours de MIE. Ce document présentera les étapes de construction du code ainsi que les interprétations du résultat final.

## Création du code

### Création et maillage du conduit

La première étape est de modéliser le conduit en respectant la géométrie donnée par l'énoncé. En procédant avec des équations paramétriques, on y arrive sans grandes difficultés. Ces lignes de codes sont suivies par quelques lignes permettant d'effectuer le maillage du domaine. Voici le code ainsi que son résultat :

```
border a(t=-x2,-x1){x=t; y=y1; }
border b(t=y1,0){x=-x1; y=t; }
border c(t=-x1,x1){x=t; y=0; }
border d(t=0,y1){x=x1; y=t; }
border e(t=x1,x2){x=t; y=y1; }
border f(t=y1,y2){x=x2; y=t; }
border g(t=x2,-x2){x=t; y=y2; }
border h(t=y2,y1){x=-x2; y=t; }

int n=5; // unite du nombre de noeuds

// maillage du domaine avec n noeuds sur le bord a,...

mesh Th=buildmesh( a(3*n)+b(3*n)+c(8*n)+d(3*n)+e(3*n)+f(2*n)+g(15*n)+h(2*n) );
```

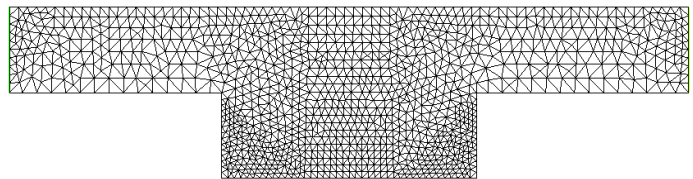


FIGURE 1 – Création et maillage du conduit

### Modélisation du liquide circulant dans le conduit

À l'intérieur de ce conduit circule un liquide. L'écoulement de ce liquide est supposé permanent, laminaire et est donc régi par les lois de Stokes. On impose un profil parabolique à l'entrée du conduit et on impose une vitesse nulle sur tout les autres bords sauf celui à la sortie. Les lignes de code correspondantes sont disponibles dans la littérature *FreeFem* (inutile de les présenter ici). Le champ de vitesse obtenu est le suivant :

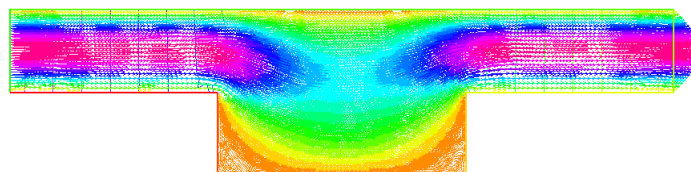


FIGURE 2 – Champ de vitesse régnant dans le conduit

## Modélisation de la circulation de l'agent polluant

Il faut à présent modéliser la circulation de l'agent polluant au sein du conduit. La concentration  $\Phi$  du polluant est régie par des équations d'advection-diffusion données dans l'énoncé. On effectue une discrétisation en temps par la méthode d'Euler implicite :  $\frac{\partial \Phi}{\partial t} = \frac{\Phi^{n+1} - \Phi^n}{dt}$  avec  $dt$  constant tel que  $t_{n+1} = t_n + dt$ . On obtient alors le problème suivant :

$$\Phi^{n+1} + dt \left( \vec{u} \cdot \vec{\nabla} \Phi^{n+1} - \nu \Delta \Phi^{n+1} \right) = \Phi^n$$

On traduit ce problème avec une approche variationnelle grâce à la méthode de Galerkin. On cherche alors une solution approchée  $\Phi_h$  appartenant à l'espace d'approximation  $V_h$  tel que pour tout  $v_h \in V_h$  :

$$\int_{\Omega} \Phi_h^{n+1} v_h + dt \left( \int_{\Omega} \vec{u} \cdot \vec{\nabla} \Phi_h^{n+1} v_h - \nu \int_{\Omega} \Delta \Phi_h^{n+1} v_h \right) = \int_{\Omega} \Phi_h^n v_h$$

En utilisant la formule de Green et sachant que sur le bord du domaine, on a  $\vec{\nabla} \Phi_h \cdot \vec{n} = 0$ , on obtient :

$$\int_{\Omega} \Phi_h^{n+1} v_h + dt \left( \int_{\Omega} \vec{u} \cdot \vec{\nabla} \Phi_h^{n+1} v_h + \nu \int_{\Omega} \vec{\nabla} \Phi_h^{n+1} \cdot \vec{\nabla} v_h \right) = \int_{\Omega} \Phi_h^n v_h$$

Ces étapes sont traduites dans le logiciel *FreeFem* grâce aux lignes de code suivantes :

```
// definition de l'espace d'approximation

espace Vh(Th,P2);          // P1 ou P2 FE space
Vh phih,vh,phin;          // inconnues et fct test.

//DEFINIR LE PROBLEME chaleur()

problem pollution(phih,vh) = // definition du probleme
  int2d(Th)(phih*vh)
  +int2d(Th)( dt*nu*( dx(phih)*dx(vh) + dy(phih)*dy(vh) ) ) // forme bilineaire
  +int2d(Th)( dt*u1*dx(phih)*vh)
  +int2d(Th)( dt*u2*dy(phih)*vh)
  + int2d(Th)( -phin*vh ) // forme lineaire
  +on(a,b,c,d,e,g,h,phih=0);
```

FIGURE 3 – Formulation variationnelle dans le logiciel *FreeFem*

On a imposé une concentration de polluant nulle sur tout les bords excepté ceux de droite et de gauche (on considère que le conduit est fermé).

Il ne reste plus qu'à introduire la concentration à  $t = 0$ , puis à écrire la boucle de temps :

```
// Boucle en temps

phin=exp(-10*((x-xs)*(x-xs)+(y-ys)*(y-ys)));

while (t<Tfinal)
{
  pollution; // resoudre le pb variationnel
  plot(phih,fill=1,value=true,nbiso=40);
  phin=phin; // reinitialiser Un
  t=t+dt;
}
plot(phih,wait=1,fill=1,value=true,nbiso=40);
```

FIGURE 4 – Boucle de temps

## Résultat final

### Images de la simulation numérique

On lance la modélisation avec  $dt = 0.05$  et  $v_{max} = 1$ . Voici quelques images de la progression de l'agent polluant, en fonction de différentes valeurs du coefficient de diffusion  $\nu$  :

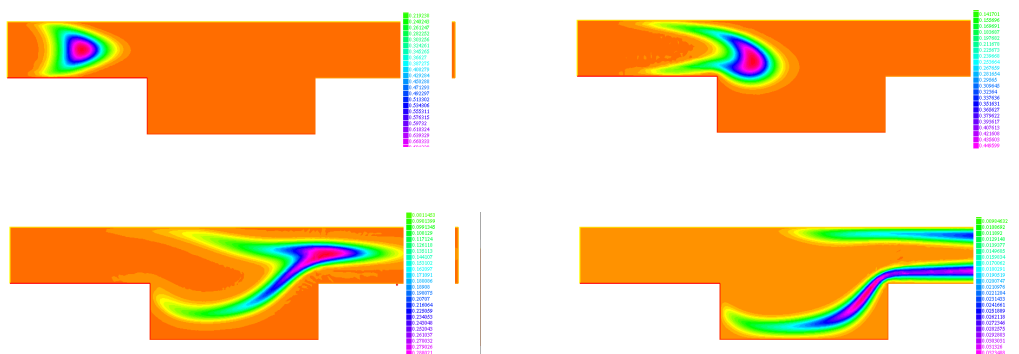


FIGURE 5 – Évolution de la concentration de l'agent polluant au sein du conduit ( $\nu = 0.001$ )

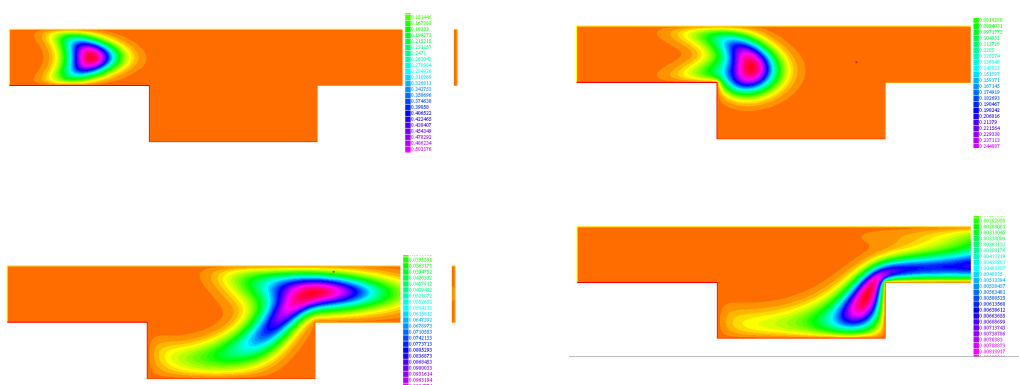


FIGURE 6 – Évolution de la concentration de l'agent polluant au sein du conduit ( $\nu = 0.01$ )

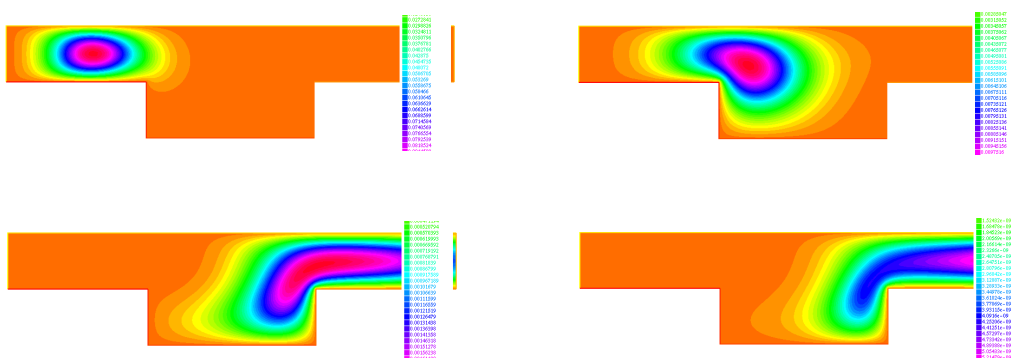


FIGURE 7 – Évolution de la concentration de l'agent polluant au sein du conduit ( $\nu = 0.1$ )

## Observations et interprétations

On peut faire plusieurs observations suite à ces modélisations :

- Plus l'on augmente la valeur de  $v_{max}$ , plus le polluant se déplace à une grande vitesse. Ceci est normal puisque cette valeur correspond à la valeur maximale de l'écoulement du fluide qui circule dans le conduit.
- Plus on augmente la valeur du coefficient de diffusion  $\nu$ , plus l'agent polluant se diffusera, et donc prendra de la place dans le conduit. Ceci confirme l'exactitude du terme de diffusion dans l'équation d'advection-diffusion : plus il est grand, plus la diffusion sera importante.
- Le polluant garde une trajectoire similaire quelle que soit la valeur du coefficient de diffusion  $\nu$ . Ceci s'explique par la nature de l'écoulement au sein du conduit : puisqu'il est laminaire et permanent, il ne varie pas au cours du temps, et donc le polluant garde sa trajectoire même s'il se diffuse de manière plus importante.
- On remarque que dans tout les cas, les coins inférieurs droit et gauche du conduit sont épargnés par l'agent polluant : ces zones ne sont pas atteintes par ce dernier. Ceci s'explique par la présence de tourbillons dans ces coins ; conséquences des équations de Stokes qui régissent l'écoulement, ainsi que les conditions de vitesse nulles sur les bords.

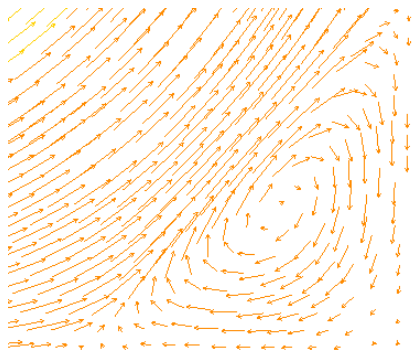


FIGURE 8 – Tourbillons présents dans le coin du conduit

## Conclusion : où doit se cacher le poisson ?

Ainsi, forts de ces observations, nous pouvons répondre à la question posée par l'énoncé :

**Le poisson doit se cacher dans les coins inférieurs droit et gauche du conduit pour avoir le plus de chances de survivre.**

En effet, c'est à ces endroits précis que l'agent polluant ne passe pas.