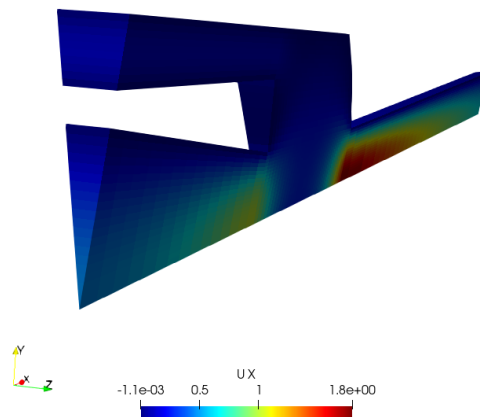




## Informe N°2: Estudio de caso axisimétrico



### Resumen:

Preparación de un caso con simetría alrededor de un eje a partir de la malla generada con *blockMesh* en el [informe anterior](#). Comparación de resultados entre ambos casos.

Autor: Guillermo Rolle  
Supervisor: Dr. Ing. César Pairetti

Febrero 2020

## Contents

<b>1</b>	<b>Introducción</b>	<b>2</b>
1.1	Información preliminar del problema . . . . .	2
<b>2</b>	<b>Preparación del caso</b>	<b>3</b>
2.1	Mallado Inicial . . . . .	4
2.2	Reparacion de la malla . . . . .	5
<b>3</b>	<b>Preparación de la simulación</b>	<b>6</b>
<b>4</b>	<b>Simulación: <i>simpleFoam</i></b>	<b>6</b>
<b>5</b>	<b>Preparación del caso RTD</b>	<b>9</b>
<b>6</b>	<b>Simulación: <i>scalarTransportFoam</i></b>	<b>10</b>
<b>7</b>	<b>Post proceso</b>	<b>11</b>

# 1 Introducción

En este trabajo se mostrará cómo modificar una malla creada con *blockMesh* para realizar un estudio de un problema axisimétrico.

Modificaremos la malla creada en el [Informe01](#) para que represente tubos de sección circular. No es necesario crear toda la geometría en 3D porque existen herramientas de OpenFOAM para aprovechar la simetría del problema. Solamente habrá 1 celda en la dirección del eje Z, pero esta vez la sección tendrá forma de "porción de pizza" para representar una sección cilíndrica.

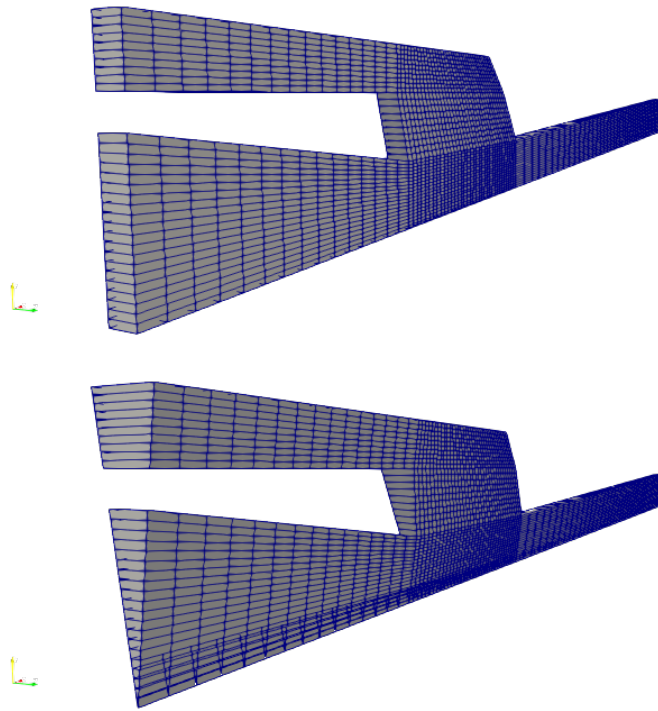


Figure 1: Comparación de la geometría. Original (arriba). Modificado (abajo)

## 1.1 Información preliminar del problema

Utilizaremos exactamente los mismos parámetros que el caso anterior para independizar mejor los cambios debido a la simetría de la malla. Las presiones a las entradas y salida son las mismas y también lo son las tolerancias de los métodos numéricos. El caso final preparado correspondiente a este informe se encuentra en el [repositorio](#).

## 2 Preparación del caso

Comenzaremos generando una copia del [caso 01](#) y eliminamos el directorio con toda la información correspondiente a la malla en `./constant/polyMesh`

```
$ cd casos_cfd
$ mkdir 02
$ cp -r ./01 ./02
$ cd 02
$ rm -r constant/polyMesh
```

Para realizar la simetría, es necesario que las caras "frontAndBack" definidas en el diccionario [blockMeshDict](#) sean 2 diferentes.

Es decir, modificar esto:

```
empty frontAndBack
(
  (0 1 12 13)
  (1 2 5 12)
  (2 3 4 5)
  (12 5 6 11)
  (11 6 7 8)
  (10 11 8 9)

  (14 27 26 15)
  (15 26 19 16)
  (16 19 18 17)
  (26 25 20 19)
  (25 22 21 20)
  (24 23 22 25)
)
```

Por esto

```
empty front
(
  (0 1 12 13)
  (1 2 5 12)
  (2 3 4 5)
  (12 5 6 11)
  (11 6 7 8)
  (10 11 8 9)
)
empty back
(
  (14 27 26 15)
  (15 26 19 16)
  (16 19 18 17)
  (26 25 20 19)
  (25 22 21 20)
  (24 23 22 25)
)
```

### OpenFOAM

#### tips:

Recordar que debe estar cargado el entorno OpenFOAM en la terminal

## 2.1 Mallado Inicial

Crearemos la malla con *blockMesh*. El resultado será la misma malla que el caso anterior pero si lo observamos en *ParaView*, las caras "front" y "back" son ahora 2 componentes distintas.

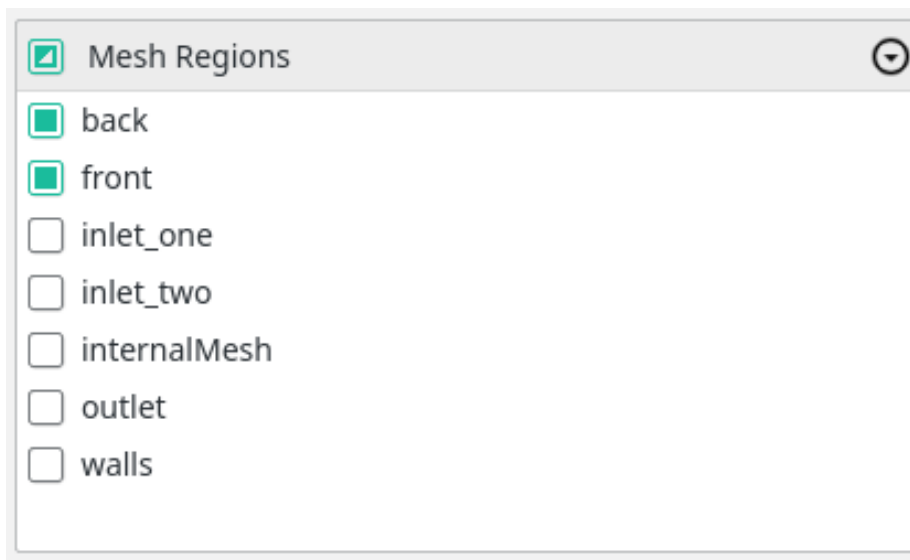


Figure 2: Regiones de malla en ParaView

Para crear la simetría utilizaremos la función [extrudeMesh](#). Necesitamos crear un diccionario en la carpeta *system* llamado *extrudeMeshDict*. Podemos copiar uno de los casos tutorial de *OpenFOAM* o mejor podemos descargar uno directamente del repositorio en cuestión: [extrudeMeshDict](#).

En este diccionario se define las caras de "cuña" (wedge), el eje de simetría y el ángulo de la extrusión.

```
constructFrom patch;
sourceCase "$FOAM_CASE";
sourcePatches (front);

// If construct from patch: patch to use for back (can be same as
    sourcePatch)
exposedPatchName back;
....
//- Linear extrusion in point-normal direction
extrudeModel      wedge;

point (0 0 0);
// punto de paso del eje
axis (1 0 0);
// direccion del eje de simetria
```

```
angle 15;
// angulo de la seccion circular
....
```

Ahora podemos ejecutar la utilidad *extrudeMesh*

```
$ extrudeMesh
```

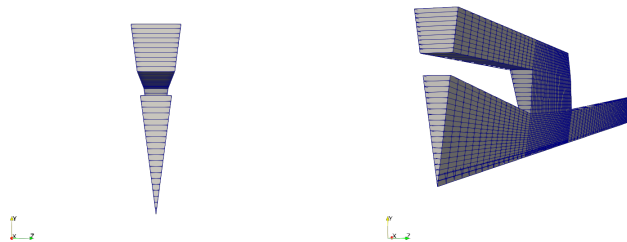


Figure 3: Sección con forma de 'porción de pizza' en ParaView

Sería interesante observar el resultado de *checkMesh* para encontrar errores en la malla. Podemos ver que hay errores debido a que, por ejemplo, hay ciertas caras que resultaron ser "aplastadas" (las del eje central) en la extrusión en cuña. Estas caras tienen área nula, lo que es problemático a la hora de resolver las simulaciones.

## 2.2 Reparacion de la malla

Podemos utilizar la funcionalidad *collapseEdges* para reparar la malla. Esta utilidad colapsa aristas pequeñas y combina aristas que estén en línea.

```
$ collapseEdges -overwrite
```

Una vez más, comprobamos con *checkMesh* para verificar que no haya inconvenientes con la malla:

```
$ checkMesh
...
Checking geometry...
Overall domain bounding box (0 0 -0.00104421) (0.1 0.00793156 0.00104421)
Mesh has 2 geometric (non-empty/wedge) directions (1 1 0)
Mesh has 3 solution (non-empty) directions (1 1 1)
Wedge front with angle 7.50001 degrees
Wedge back with angle 7.50001 degrees
All edges aligned with or perpendicular to non-empty directions.
Boundary openness (-5.20681e-21 -2.03166e-15 2.13322e-15) OK.
Max cell openness = 2.62994e-16 OK.
Max aspect ratio = 22.9592 OK.
```

### Linux tips:

Es posible escribir un script de comandos de Linux y ejecutarlos con `./script.sh`. Ver *comandos.sh*

```

Minimum face area = 2.91172e-09. Maximum face area = 2.2737e-06. Face
    area magnitudes OK.
Min volume = 1.24767e-12. Max volume = 5.46506e-10. Total volume =
    3.41533e-07. Cell volumes OK.
Mesh non-orthogonality Max: 51.1811 average: 16.3359
Non-orthogonality check OK.
Face pyramids OK.
Max skewness = 0.930108 OK.
Coupled point location match (average 0) OK.

Mesh OK.

End

```

### 3 Preparación de la simulación

Vamos a utilizar las mismas condiciones iniciales y mismos diccionarios que en el otro caso. Sin embargo, debemos actualizar los archivos `0/U` y `0/p` con los parches *front* y *back* como las definimos anteriormente.

También, debemos modificar el tipo de los parches de *empty* a *wedge*. [Ver más información sobre boundaries](#)

Las modificaciones en los archivos son (en ambos):

```

front
{
type            wedge;
}
back
{
type            wedge;
}
}

```

Ver los archivos completos: `0/U`; `0/p`

Los otros diccionarios en *system*, como *fvSolution* y *fvSchemes*, los dejamos entonces sin modificar como en la simulación anterior.

### 4 Simulación: *simpleFoam*

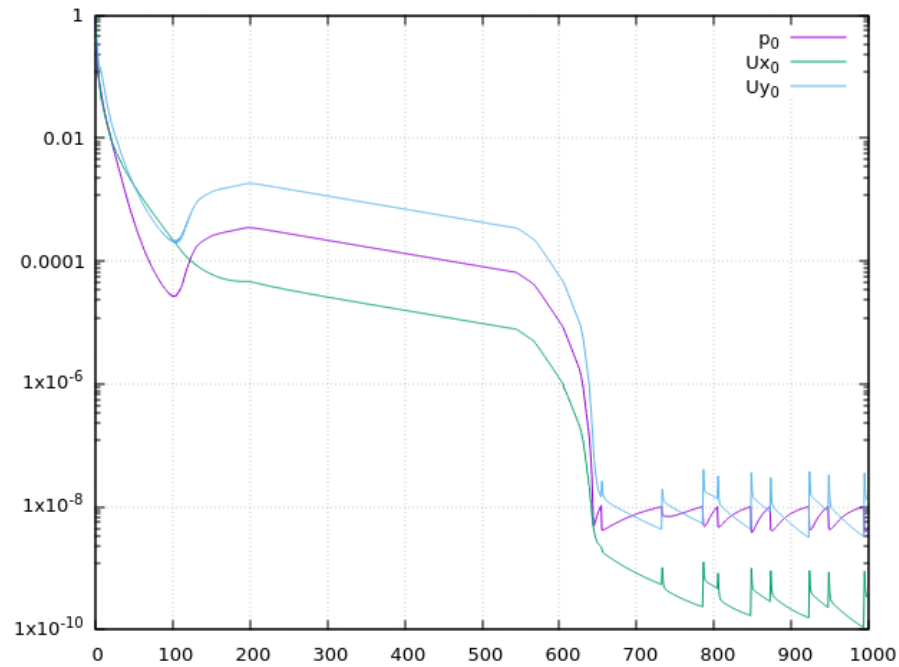
En este caso, la simulación no converge. Podemos ver que calcula hasta la última iteración definida en el *controlDict*. Volvemos a ejecutar la simulación y procesamos los archivos log para verificar los residuales. Luego los graficamos entrando a la carpeta `./logs` con *gnuplot*.

```

$ simpleFoam | tee log.simpleFoam
$ foamLog log.simpleFoam
$ gnuplot

```

```
> load 'graficar_res.gp'
```



**gnuplot tips:**  
Es posible  
escribir un  
script de  
comandos de  
gnuplot y  
ejecutarlos con  
gnuplot  
<script.gp> o  
desde gnuplot  
con 'load  
<script.gp>'

Figure 4: Residuales no convergentes

Esto es así porque *simpleFoam* está intentando resolver para  $U_z$  también por la simetría y ese valor no converge. Para solucionarlo vamos a evitar controlar por  $U_z$  por fines prácticos. Para eso, debemos modificar en el archivo `system/fvSolution`:

Esta línea:  $U$   $1e-5$ ;  
Por esta:  $(U_x|U_y)$   $1e-5$ ;

Ver archivo completo: [system/fvSolution](#)



Podemos ejecutar la simulación nuevamente con los mismos comandos anteriores y vemos los residuales nuevamente. Esta vez, con  $U_z$  también:

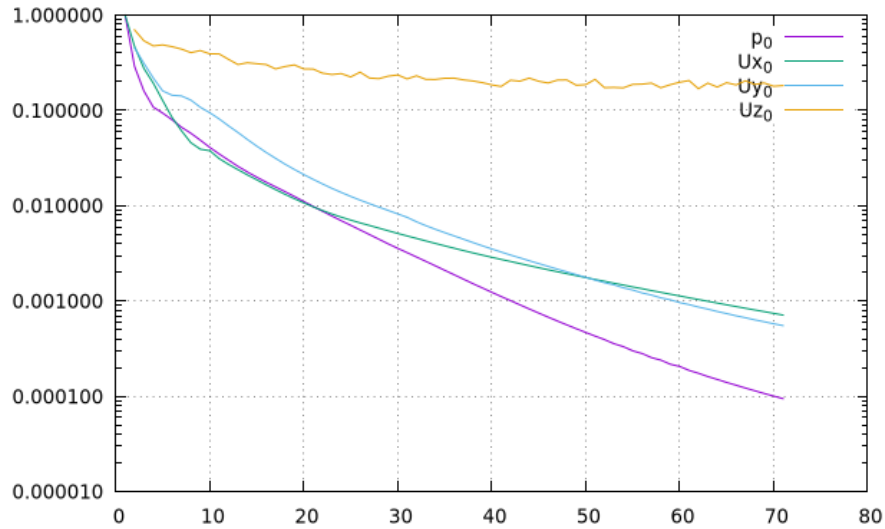


Figure 5: Residuales convergentes

Nótese que no se controla por  $U_z$ .

En la figura 6 se compara el perfil de velocidad (componente X) obtenido en el caso axisimétrico y en el caso previo. Se observa que el valor máximo es mayor en el caso axisimétrico y además se encuentra sobre el eje de simetría.

Otra observación interesante que podemos realizar en la misma figura, es que en este nuevo caso hay una mayor cantidad de líneas de traza provenientes de la entrada de agroquímico. Esto nos permite sospechar un aumento de la concentración final en la salida, una vez hayamos terminado la simulación RTD.

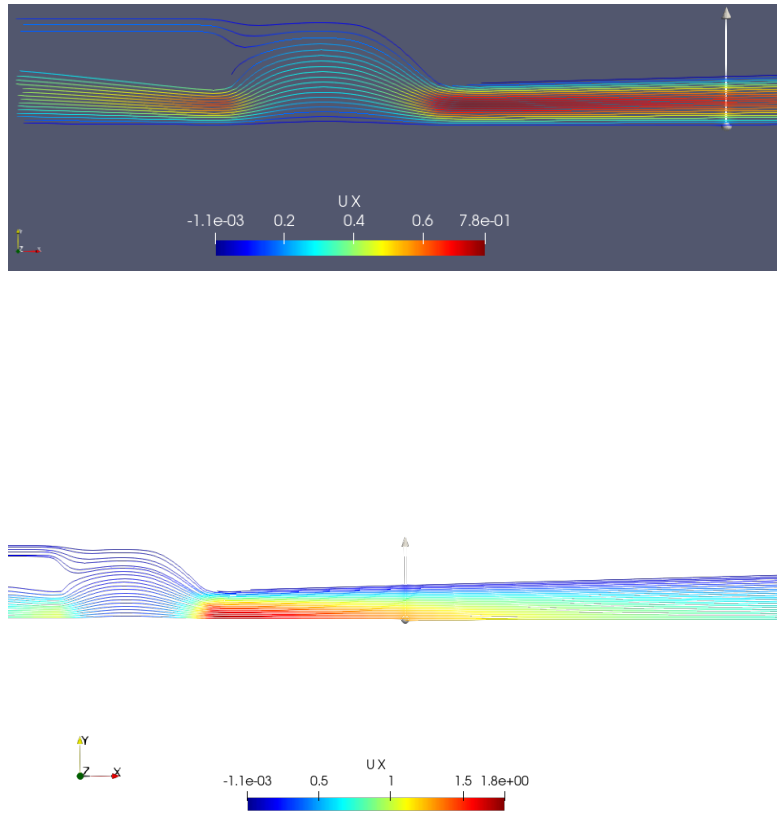


Figure 6: Líneas de traza. Escala de velocidad  $U_x$ . Informe01 (arriba), Informe02 (abajo)

## 5 Preparación del caso RTD

Una vez obtenido el campo de velocidad con *simpleFoam*, preparamos el caso RTD. Dentro del directorio del caso principal, creamos una carpeta llamada "RTD" y usaremos de plantilla el caso RTD del informe anterior. Reemplazamos las condiciones iniciales de velocidad por el resultado final de la última simulación (en este caso, iteración número 71):

```
$ mkdir RTD
$ cd RTD/
$ cp ../71/U ./0
```

Debemos modificar las condiciones de "T" de manera que coincidan con los patches definidos en la malla nueva. [Ver 0/T](#).

Copiamos la malla generada antes al nuevo caso:

```
$ cp -r ../constant/polyMesh ./constant
```

Los otros archivos en `constant` y `system` los dejamos como en el caso anterior. En el siguiente link se encuentra el directorio principal del caso RTD correspondiente a este informe: [RTD - 02](#)

**Linux tips:**  
El comando `cp -r` indica una copia recursiva. Es decir, permite copiar directorios completos en lugar de archivos puntuales. Ver `man cp`

## 6 Simulación: *scalarTransportFoam*

Ya tenemos el caso RTD preparado y podemos ejecutar la simulación con el solver *scalarTransportFoam*. Cabe destacar que este es un solver transitorio a diferencia de *simpleFoam* que es estacionario.

```
$ scalarTransportFoam | tee log.scalarTransportFoam
$ foamLog log.scalarTransportFoam
```

Podemos cargar el archivo de estado de ParaView *RTD.pvsm* para ver las gráficas y filtros correspondientes.

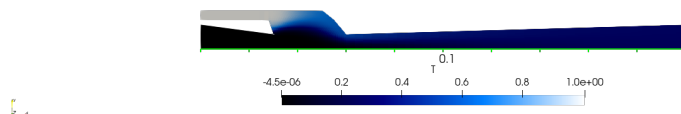


Figure 7: Concentración de agroquímico ( $t = 2s$ )

En la figura 8 se muestra la concentración de agroquímicos de los dos casos realizados. Se aprecia claramente un aumento en la concentración final.

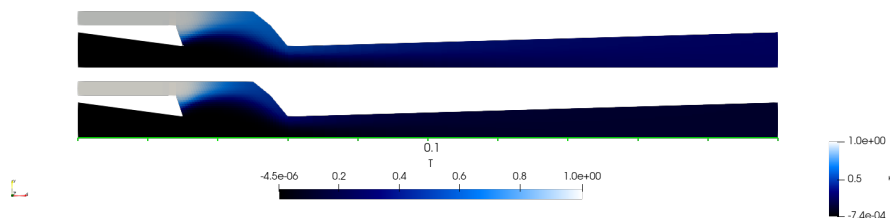


Figure 8: Concentración - Informe01 (abajo), Informe02 (arriba)

## 7 Post proceso

Para obtener el valor promedio de concentración a la salida, podemos utilizar la función *patchAverage* definida en el caso anterior.

```
$ postProcess -func 'patchAverage'
```

Esto nos crea una carpeta `postProcess` con los resultados. Graficamos ahora en *gnuplot*:

```
$ gnuplot
> cd 'postProcessing/patchAverage/0'
> set term wxt 0
> plot 'surfaceFieldValue.dat' w l title "outlet"
> set xlabel "Tiempo [s]"
> set ylabel "Concentracion []"
> set grid
> replot
```

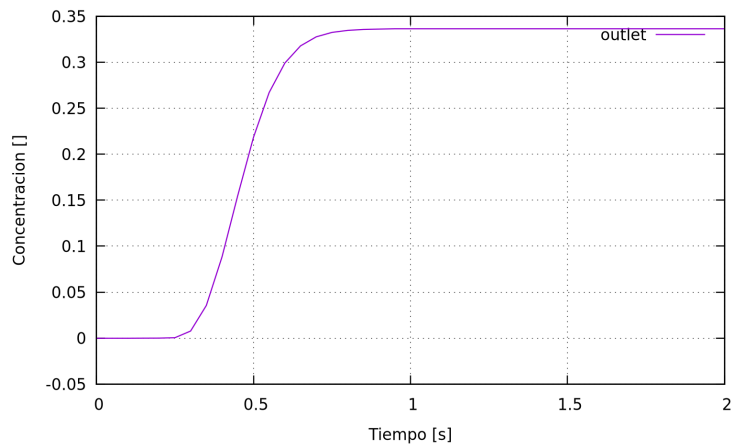


Figure 9: Concentración en la salida en función del tiempo

Por último, verificar el perfil de concentración y comparar el perfil de velocidad a la salida en ambos casos con ParaView.

En la figura 10 se observa que la concentración en la salida no es completamente uniforme. Ésta oscila en un rango del 1.5%

Es interesante comparar el perfil de velocidad en los dos casos. En la figura 11 se observa que en el caso axisimétrico (Informe02) la velocidad del fluido alcanza un máximo bastante mayor al del primer caso, y, como era de esperar, ese máximo se encuentra sobre el eje central de la malla.

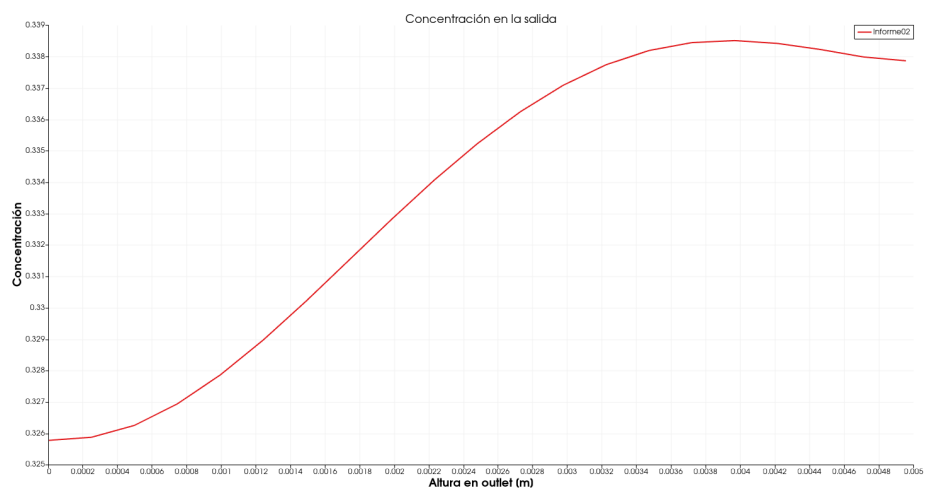


Figure 10: Perfil de concentración a la salida

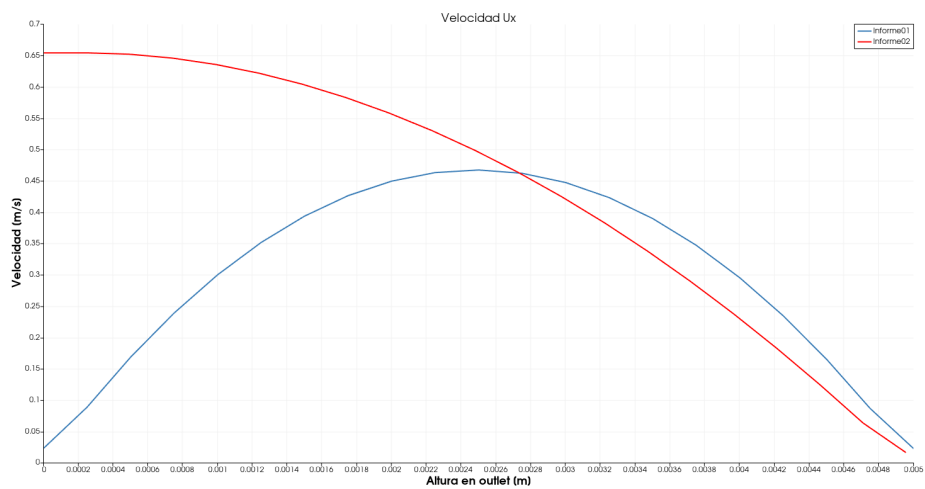


Figure 11: Perfil de velocidad  $U_x$  en la salida