



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA CIVIL – HIDRÁULICA

**MODELACIÓN NUMÉRICA CFD DE PROCESOS DE BIODEGRADACIÓN
AEROBIA**

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
BENIGNO DURÁN AGUILAR

TUTOR PRINCIPAL
DR. VÍCTOR MANUEL ARROYO CORREA
INSTITUTO MEXICANO DE TECNOLOGÍA DEL AGUA

JIUTEPEC, MORELOS, OCTUBRE 2019



Universidad Nacional
Autónoma de México

Dirección General de Bibliotecas de la UNAM

Biblioteca Central



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

JURADO ASIGNADO:

Presidente: Dr. Aguilar Chávez Ariosto
Secretario: M.I. González Verdugo José A.
Vocal: Dr. Arroyo Correa Víctor Manuel
1 er. Suplente: M.I. Escalante Estrada Mauricio J.
2 d o. Suplente: Dr. Ballinas González Héctor Alonso

Lugar donde se realizó la tesis: Instituto Mexicano de Tecnología del Agua (IMTA), Jiutepec, Morelos.

TUTOR DE TESIS:

Dr. Víctor Manuel Arroyo Correa

FIRMA

Dedicatoria

Con respeto y agradecimiento a mis padres Benigno Durán Méndez y María Trinidad Aguilar Santillán†.

A mis hermanos Rogelio, Santiago Raúl y Rubén por su ejemplo y motivación.

Agradecimientos

Al Dr. Víctor Manuel Arroyo Correa, director de tesis.

A mis sinodales: Dr. Ariosto Aguilar Chávez, M. I. José Alfredo González Verdugo, M. I. Mauricio de Jesús Escalante Estrada, y Dr. Héctor Alonso Ballinas González, por su revisión y comentarios a este trabajo.

A la Universidad Nacional Autónoma de México (UNAM), al Instituto Mexicano de Tecnología del Agua (IMTA) y al Consejo Nacional de Ciencia y Tecnología (CONACyT).

Resumen

La investigación sobre procesos de biodegradación aerobia resueltos con modelación numérica CFD (*Computational Fluid Dynamics*, en inglés) ha tomado relevancia en fechas recientes. Los modelos de biodegradación consideran en forma simultánea, la difusión, advección y reacción, de los componentes físico, químico y biológico.

El objetivo de este trabajo de investigación es mostrar una metodología para la modelación numérica con procesos de biodegradación aerobia, teniendo como base de solución hidrodinámica las bibliotecas de OpenFOAM, 2018 (*Open Field Operation and Manipulation*) y el lenguaje de programación Python para la solución de las ecuaciones de difusión, advección y reacción.

Para el pre – procesamiento de la simulación numérica se empleó el software SALOME y para la visualización de los resultados se utilizó el software ParaView (2018) que es un software gratuito de visualización científica interactiva.

Para construir los escenarios de biodegradación aerobia se utilizó como referencia lo presentado en el documento *Contaminant Transport and Biodegradation A Numerical Model for Reactive Transport in Porous Media*, de Celia M. A., J. S. Kindred e I. Herrera, (1989).

Además, se presentan diferentes ejemplos para mostrar la utilidad y aplicación de la metodología que puede ser implementada para una, dos o tres dimensiones.

Palabras clave

Modelación numérica, CFD, OpenFOAM, Python, SALOME, ParaView, advección, difusión, reacción y biodegradación aerobia.

Abstract

The research on aerobic biodegradation processes resolved with numerical modeling CFD (Computational Fluid Dynamics) has taken relevance recently. Biodegradation models simultaneously consider the diffusion, advection and reaction of the physical, chemical and biological components.

The objective of this research work is to show a methodology for numerical modeling with aerobic biodegradation processes, having as base of the hydrodynamic solution the OpenFOAM libraries, 2018 (Open field operation and manipulation) and the Python programming language for the solution of the diffusion, advection and reaction equations.

For the pre - processing of the numerical simulation the SALOME software is used and for the visualization of the results the software ParaView (2018) is used, which is a free interactive scientific software.

To construct the aerobic biodegradation scenarios, the information presented in the document Contaminant Transport and Biodegradation A Numerical Model for Reactive Transport in Porous Media, by Celia M. A., J. S. Kindred and I. Herrera, (1989) was used as a reference.

In addition, different examples are presented to show the utility and the application that is implemented for one, two or three dimensions.

Keywords

Numerical modeling, CFD, OpenFOAM, Python, SALOME, ParaView, advection, diffusion, reaction and aerobic biodegradation.

Contenido

Figuras	9
Tablas	14
1 Introducción.....	15
1.1 Justificación	16
1.2 Objetivos	16
2 Antecedentes generales.....	17
2.1 Dinámica de fluidos computacional y sus aplicaciones en la Hidráulica y modelos Hidrobiológicos.....	17
2.2 Fundamentos principales para el estudio de la ecuación de difusión – advección – reacción	25
2.2.1 Difusión	25
2.2.2 Advección.....	27
2.2.3 Término reactivo en la ecuación de difusión – advección	28
2.3 Biodegradación aerobia	29
3 Estado del arte	31
3.1 Ecuación de difusión – advección – reacción	31
3.2 Análisis diferencial de la ecuación Hidrodinámica	33
3.3 OpenFOAM.....	36
3.4 SALOME	38
3.5 Python.....	39
3.6 ParaView.....	39
4 Construcción del modelo numérico 3D con términos de biodegradación aerobia	
41	
4.1 Verificación del modelo numérico con términos de biodegradación aerobia.	
47	

4.2	Ejemplo de aplicación 2D, transporte de un contaminante conservativo	50
4.3	Ejemplo de aplicación 2D, transporte de dos especies.....	56
5	Aplicación del modelo numérico a un sistema de lagunas de estabilización aerobia en 3D.....	68
5.1	Ventajas y desventajas del sistema de lagunas de estabilización	69
5.2	Tipos de lagunas de estabilización aerobia	70
5.2.1	De baja tasa	70
5.2.2	De alta tasa	71
5.3	Descripción general del problema de estudio	72
5.4	Determinación de datos hidrodinámicos	76
5.4.1	Pre - procesamiento de la simulación numérica.....	76
5.4.2	Solución hidrodinámica con OpenFOAM.....	78
5.5	Simulación del proceso de biodegradación aerobia 3D	92
5.5.1	Resultados	97
6	Conclusiones.....	109
7	Referencias.....	111
8	Anexos.....	114
8.1	Anexo 1. Modelo numérico para la solución unidimensional (1D) con términos de biodegradación aerobia.....	114
8.2	Anexo 2. Modelo numérico para la solución bidimensional (2D) con términos de biodegradación aerobia.....	117
8.3	Anexo 3. Directorio de archivos OpenFOAM para la simulación hidrodinámica de las lagunas de estabilización	121
8.4	Anexo 4. Código de programación en Python para la solución de las ecuaciones de advección, difusión y reacción	148

Figuras

Figura 2-1 Comparación del modelo computacional por etapas (Covarrubias, Velázquez, Bustamante, Delgado, & Escalante, 2015)	19
Figura 2-2 Predicción de regímenes de reacción y el gas disuelto total aguas abajo del vertedor (Wang, Marcela, & Larry, 2018).....	19
Figura 2-3. Comparación de la distribución de la velocidad a lo largo de la línea de simetría en la dirección del flujo principal a partir de simulaciones en ANSYS FLUENT, OpenFOAM y experimentos (Franziska, Christoph, Thomas, Friederike, & Rudiger, 2016).....	20
Figura 2-4 Comparación cuantitativa entre valores de velocidad medidos y simulados (Aguilar Chavez & Laurel Castillo, 2011).....	21
Figura 2-5 Comparación cualitativa entre distribución de concentración de aire (Aguilar Chavez & Laurel Castillo, 2011).....	21
Figura 2-6 Evolución del reactivo c_2 en el lago cuando se inyectaron escalares alternativamente (Millán Barrera, Arroyo Correa, & Laurel Castillo, 2013).....	22
Figura 2-6 Aplicación de BIOMOC (I. Essaid & A. Bekins, 1997).....	23
Figura 2-7 Tubo de difusión de gases de Graham (Cussler, 2007).....	25
Figura 2-8 Transporte de una sustancia si se produjera solo advección (Román, 2017)	27
Figura 2-9 Transporte de una sustancia si se produjera solo difusión y advección (Román, 2017)	27
Figura 3-1 Comparación de solución numérica OTF y solución analítica (Celia, Kindred, & Ismael, June 1989)	31
Figura 3-2 OpenFOAM, software CFD de código abierto (OpenFOAM, 2018)	37
Figura 3-3 Estructura general de OpenFOAM (OpenFOAM, 2018)	37
Figura 3-4 Visualización del software SALOME (SALOME, 2018).....	38
Figura 3-5 Simulación de flujo utilizando OpenFOAM y ParaView para visualización (ParaView, 2018).....	40
Figura 4-1. Solución numérica de las ecuaciones (3.1) y (3.2) utilizando Python 2019 mediante un esquema explícito para un tiempo de simulación de 68 días.	49

Figura 4-2. Solución numérica de las ecuaciones (3.1) y (3.2) utilizando una función de prueba óptima (OTF siglas en inglés) para un tiempo de simulación de 68 días, Celia M. A., J. S. Kindred e I. Herrera, (1989).....	49
Figura 4-3 Solución analítica vista en planta.....	52
Figura 4-4 Solución del modelo numérico vista en planta.....	52
Figura 4-5 Solución ELLAM, vista en planta en el $t = 510\pi$, (Hong, Sharpley, & Shushuang, 1996).....	53
Figura 4-6 Solución en elemento finito para un $t = 510\pi$ tipo GAL, (Hong, Sharpley, & Shushuang, 1996).....	53
Figura 4-7 Solución en elemento finito para un $t = 510\pi$ tipo QPG, (Hong, Sharpley, & Shushuang, 1996).....	54
Figura 4-8 Comparación de solución analítica y modelo numérico en el tiempo $t = (110)\pi$	54
Figura 4-9 Comparación de solución analítica y modelo numérico en el tiempo $t = (210)\pi$	55
Figura 4-10 Comparación de solución analítica y modelo numérico en el tiempo $t = (310)\pi$	55
Figura 4-11 Comparación de solución analítica y modelo numérico en el tiempo $t = (410)\pi$	55
Figura 4-12 Comparación de solución analítica y modelo numérico en el tiempo $t = (510)\pi$	56
Figura 4-13. Vista 2D de c_1 para un $t = 0$ días.....	59
Figura 4-14. Vista 2D de c_2 para un $t = 0$ días.....	59
Figura 4-15. Corte transversal de c_1 y c_2 para un $t = 0$ días con $c_2 \text{ máx} = 10\text{mg/l}$	60
Figura 4-16. Vista 2D de c_1 para un $t = 20$ días.....	60
Figura 4-17. Vista 2D de c_2 para un $t = 20$ días.....	61
Figura 4-18. Corte transversal de c_1 y c_2 para un $t = 20$ días con $c_2 \text{ máx} = 5.93\text{mg/l}$	61
Figura 4-19. Vista 2D de c_1 para un $t = 50$ días.....	62
Figura 4-20. Vista 2D de c_2 para un $t = 50$ días.....	62

Figura 4-21. Corte transversal de c_1 y c_2 para un $t = 50$ días con $c_2 \text{ máx} = 3.74 \text{ mg/l}$	63
Figura 4-22. Vista 2D de c_1 para un $t = 100$ días.....	63
Figura 4-23. Vista 2D de c_2 para un $t = 100$ días.....	64
Figura 4-24. Corte transversal de c_1 y c_2 para un $t = 100$ días con $c_2 \text{ máx} = 2.02 \text{ mg/l}$	64
Figura 4-25. Vista 2D de c_1 para un $t = 150$ días.....	65
Figura 4-26. Vista 2D de c_2 para un $t = 150$ días.....	65
Figura 4-27. Corte transversal de c_1 y c_2 para un $t = 150$ días con $c_2 \text{ máx} = 1.20 \text{ mg/l}$	66
Figura 5-1. Vista aérea del sistema de lagunas aireadas del Distrito Sanitario de Veazie, Maine, USA (DEP, 2003).....	71
Figura 5-2. Vista satelital de la zona de estudio	73
Figura 5-3 Diagrama de flujo del proceso de tratamiento general (DEP, 2003)	74
Figura 5-4 Dimensiones de la laguna 1 de estabilización aerobia.....	75
Figura 5-5 Dimensiones de la laguna 2 y 3 de estabilización aerobia.....	75
Figura 5-6 Geometría de la laguna 1 de estabilización aerobia	76
Figura 5-7 Geometría de la laguna 2 y 3 de estabilización aerobia.....	76
Figura 5-8 Malla de la laguna 1 de estabilización aerobia.....	77
Figura 5-9 Malla de la laguna 2 y 3 de estabilización aerobia	77
Figura 5-10. Conversión de mallas de SALOME a OpenFOAM para la laguna 1 .	78
Figura 5-11. Conversión de mallas de SALOME aOpenFOAM para la laguna 2 y 3	78
Figura 5-12. Comando checkMesh para verificar la validez de la malla.....	78
Figura 5-13. Condiciones de frontera de la laguna 1 para OpenFOAM.	85
Figura 5-14. Condiciones de frontera de la laguna 2 y 3 para OpenFOAM.....	85
Figura 5-15. Vista en planta del campo de velocidades en la laguna 1, con $z = 2.9$ m y $t = 7,200$ s.....	86
Figura 5-16. Vista ampliada en planta del campo de velocidades en la entrada de la laguna 1, con $z = 2.9$ m y $t = 7,200$ s.	86
Figura 5-17. Vista en la sección de entrada del flujo en la laguna 1, $t = 7,200$ s...	87

Figura 5-18. Vista en la sección de salida del flujo en la laguna 1, $t = 7,200$ s.	87
Figura 5-19. Vista en planta del campo de velocidades en la laguna 2 y 3, con $z = 3.0$ m y $t = 7,200$ s.....	88
Figura 5-20. Vista ampliada en planta del campo de velocidades en la laguna 2 y 3, con $z = 3.0$ m y $t = 7,200$ s.	88
Figura 5-21. Vista en la sección de entrada del flujo en la laguna 2 y 3, $t = 7,200$ s.	89
Figura 5-22. Vista en la sección de salida del flujo en la laguna 2 y 3, $t = 7,200$ s.	89
Figura 5-23. Perfil de velocidad en el centro de la Laguna 1, $t = 7,200$ s.....	90
Figura 5-24. Perfil de velocidad en el centro de la Laguna 2 y 3, $t = 7,200$ s.	90
Figura 5-25. Exportación de datos U_x , U_y y U_z de ParaView a archivo <i>txt</i>	91
Figura 5-26. Procesamiento de los datos de velocidad para su utilización en Python en archivo <i>txt</i>	91
Figura 5-27. Condición inicial y condiciones de frontera para la laguna 1 del proceso de biodegradación aerobia.	93
Figura 5-28. Condición inicial y condiciones de frontera para la laguna 2 y 3 del proceso de biodegradación aerobia.	94
Figura 5-29 Oxígeno disuelto (c_1) para la laguna 1 en todos los tiempos de simulación.	97
Figura 5-30 Oxígeno disuelto (c_1) para la laguna 2 en todos los tiempos de simulación.	98
Figura 5-31 Oxígeno disuelto (c_1) para la laguna 3 en todos los tiempos de simulación.	98
Figura 5-32 Concentración de c_2 en la laguna 1, elevación = 0.9 m y $t = 10$ días.	99
Figura 5-33 Concentración de c_2 en la laguna 2, elevación = 0.9 m y $t = 10$ días.	99
Figura 5-34 Concentración de c_2 en la laguna 3, elevación = 0.9 m y $t = 10$ días.	99
Figura 5-35 Concentración de c_2 en la laguna 1, elevación = 1.9 m y $t = 10$ días.	100
Figura 5-36 Concentración de c_2 en la laguna 2, elevación = 1.9 m y $t = 10$ días.	100

Figura 5-37 Concentración de c_2 en la laguna 3, elevación = 1.9 m y $t = 10$ días.	100
Figura 5-38 Concentración de c_2 en la laguna 1, elevación = 2.9 m y $t = 10$ días	101
Figura 5-39 Concentración de c_2 en la laguna 2, elevación = 2.9 m y $t = 10$ días	101
Figura 5-40 Concentración de c_2 en la laguna 3, elevación = 2.9 m y $t = 10$ días.	101
Figura 5-41 Concentración de c_2 en la laguna 1, elevación = 0.9 m y $t = 30$ días.	102
Figura 5-42 Concentración de c_2 en la laguna 2, elevación = 0.9 m y $t = 30$ días.	102
Figura 5-43 Concentración de c_2 en la laguna 3, elevación = 0.9 m y $t = 30$ días.	102
Figura 5-44 Concentración de c_2 en la laguna 1, elevación = 1.9 m y $t = 30$ días.	103
Figura 5-45 Concentración de c_2 en la laguna 2, elevación = 1.9 m y $t = 30$ días.	103
Figura 5-46 Concentración de c_2 en la laguna 3, elevación = 1.9 m y $t = 30$ días.	103
Figura 5-47 Concentración de c_2 en la laguna 1, elevación = 2.9 m y $t = 30$ días.	104
Figura 5-48 Concentración de c_2 en la laguna 2, elevación = 2.9 m y $t = 30$ días.	104
Figura 5-49 Concentración de c_2 en la laguna 3, elevación = 2.9 m y $t = 30$ días	104
Figura 5-50 Concentración de c_2 en la laguna 1, elevación = 0.9 m y $t = 59$ días.	105
Figura 5-51 Concentración de c_2 en la laguna 2, elevación = 0.9 m y $t = 59$ días.	105

Figura 5-52 Concentración de c_2 en la laguna 3, elevación = 0.9 m y t = 59 días.	105
Figura 5-53 Concentración de c_2 en la laguna 1, elevación = 1.9 m y t = 59 días	106
Figura 5-54 Concentración de c_2 en la laguna 2, elevación = 1.9 m y t = 59 días	106
Figura 5-55 Concentración de c_2 en la laguna 3, elevación = 1.9 m y t = 59 días.	106
Figura 5-56 Concentración de c_2 en la laguna 1, elevación = 2.9 m y t = 59 días	107
Figura 5-57 Concentración de c_2 en la laguna 2, elevación = 2.9 m y t = 59 días	107
Figura 5-58 Concentración de c_2 en la laguna 3, elevación = 2.9 m y t = 59 días	107

Tablas

Tabla 4-1. Valores máximo y mínimo después de una vuelta $t = 510\pi$, obtenida con diferentes métodos	51
Tabla 4-2. Valores mínimos y máximos de c_1 con diferentes tiempos de análisis	67
Tabla 4-3. Valores mínimos y máximos de c_2 con diferentes tiempos de análisis	67
Tabla 5-1 Especificaciones generales del sistema de lagunas de estabilización..	74
Tabla 5-2 Estructura de OpenFOAM para la simulación con el solver pisoFOAM	79
Tabla 5-3 Condiciones de frontera para la Laguna 1.	82
Tabla 5-4 Condiciones de frontera para la Laguna 2 y 3.....	83
Tabla 5-5 Descripción de los tipos de frontera utilizados en OpenFOAM.	84

1 Introducción

Los modelos numéricos de Dinámica de Fluidos Computacionales (CFD, *Computational Fluid Dynamics*, en inglés) se utilizan para resolver las ecuaciones Navier – Stokes, energía y conservación de masa, y tienen una buena versatilidad para analizar problemas con geometrías no regulares, todo esto por medio de computadoras digitales. Adicional a las geometrías no regulares, también existen propuestas de solución de dinámica de fluidos en más de una fase y su interacción entre ellas.

En la actualidad, los modelos numéricos han sido de gran ayuda para la simulación de diferentes procesos físicos, existe una tendencia creciente en el uso de modelos numéricos predictivos de diferentes procesos, entre los cuales se encuentran los procesos hidrodinámicos e hidrobiológicos que sirven para la toma de decisiones en la ingeniería.

El objetivo principal de este trabajo fue desarrollar un modelo numérico que solucione las ecuaciones que gobiernan los procesos de biodegradación aerobia, teniendo como base la plataforma de solución hidrodinámica del modelo CFD OpenFOAM y para la solución del conjunto de ecuaciones de difusión – advección – reacción se utilizó el lenguaje Python. El sistema de CFD OpenFOAM (2018) se utilizó por ser un código abierto, con ayudas de aplicación didácticas y con referencias de uso reportados en la literatura, especialmente para el tema que se desarrolló este trabajo.

Una vez desarrollado el solucionador se verificarán los resultados obtenidos por Celia M. A., J. S. Kindred e I. Herrera, (1989). Cabe destacar que esta referencia es la principal debido a la utilización del modelo de ecuaciones de transporte de contaminantes y biodegradación y que es tomada como marco teórico en muchas otras referencias.

Posteriormente se realizarán ejemplos de aplicación en dos y tres dimensiones, además se tiene también la finalidad de demostrar la importancia y utilidad de softwares libres de código abierto que sirven para la solución de problemas específicos que se presentan en Mecánica de Fluidos.

1.1 Justificación

Por lo general las plataformas de código abierto y de licencia dejan muy limitadas las aplicaciones para procesos de biodegradación aerobia. Por lo tanto, es necesario la incorporación de estos procesos que tengan una aplicación práctica.

Con el trabajo de investigación se tendrá un modelo numérico que permitirá solucionar y visualizar los resultados para problemas específicos en el proceso de transporte de contaminantes con términos de biodegradación aerobia, asimismo servirá de base para incorporar soluciones con un sentido más práctico y que involucren reacciones más complejas.

1.2 Objetivos

- Solución del modelo numérico CFD de procesos de biodegradación aerobia.
- Caracterizar los diferentes componentes que intervienen en el flujo con el proceso de biodegradación aerobia.
- Obtener y analizar simulaciones realizadas en 1D, 2D y 3D.
- Comparar los resultados obtenidos mediante el modelo numérico con respecto a otros estudios.

2 Antecedentes generales

2.1 Dinámica de fluidos computacional y sus aplicaciones en la Hidráulica y modelos Hidrobiológicos

La historia de la dinámica de fluidos computacional (CFD) comenzó a principios de la década de 1970, alrededor de ese tiempo, se convirtió en un acrónimo de una combinación de física, matemáticas numéricas y, en cierta medida, ciencias de la computación empleadas para simular flujos de fluidos. En la década de 1980, se comenzó con la solución de las primeras ecuaciones de Euler bidimensionales (2-D) y también tridimensionales (3-D), más tarde a mediados de la década de 1980, el enfoque comenzó a cambiar a la simulación de flujos viscosos significativamente más exigente gobernada por las ecuaciones de Navier-Stokes (Blazek, 2001).

El modelado tradicional en ingeniería se basa en gran medida en modelos empíricos o semi empíricos. Estos modelos a menudo funcionan muy bien para condiciones sencillas bien conocidas, pero no son confiables para problemas complejos. Una nueva tendencia es que los ingenieros utilizan cada vez más la dinámica de fluidos computacional (CFD) para analizar el flujo y el rendimiento en el diseño de nuevos equipos y procesos. CFD permite un análisis detallado del flujo combinado con la transferencia de masa y calor. Las herramientas modernas de CFD también pueden simular el transporte de especies químicas, reacciones químicas, combustión, evaporación, condensación y cristalización (Andersson, y otros, 2012).

La dinámica de fluidos computacional (CFD) se ha convertido en una herramienta indispensable para los ingenieros. El tiempo de solución para un análisis de CFD se está reduciendo continuamente ya que las computadoras son cada vez más rápidas y el software utiliza algoritmos cada vez más eficientes. El bajo costo, la precisión satisfactoria y el tiempo de simulación cada vez menor permiten que CFD sea un complemento mejor con el empleo de modelos físicos. Hay muchos programas comerciales disponibles, que se han convertido en fáciles de usar, y con muchas configuraciones predeterminadas, de modo que incluso un usuario sin experiencia puede obtener resultados confiables para problemas simples. Sin embargo, la

mayoría de las aplicaciones requieren una comprensión más profunda de la dinámica de fluidos y modelación numérica. Como los modelos no son completos, los ingenieros de CFD tienen que determinar qué modelos son los más apropiados para cada caso particular. Esto es importante ya que los resultados del análisis a menudo se usan para tomar decisiones sobre qué prototipos y procesos se deben construir (Andersson, y otros, 2012).

Para el área de la Hidráulica y modelos Hidrobiológicos se encuentran diferentes investigaciones que utilizan CFD para diversos estudios, por mencionar algunos se citan los siguientes artículos:

“Comparación de resultados experimentales de un Venturi con simulación de Dinámica de Fluidos Computacional (CFD)” (Covarrubias, Velázquez, Bustamante, Delgado, & Escalante, 2015) presenta un análisis comparativo de la información experimental obtenida en laboratorio y reportada por Sotelo Ávila (1979) y la utilización de CFD, cabe destacar que los autores obtuvieron buena congruencia entre los valores del flujo, gasto y caída de presión en el manómetro diferencial.

Los autores destacan que la modelación con la Dinámica de Fluidos Computacional y la experimentación en laboratorio son métodos complementarios de análisis de fenómenos hidráulicos, y su combinación es la mejor estrategia para el encuentro de una solución técnica, práctica, económica y confiable.

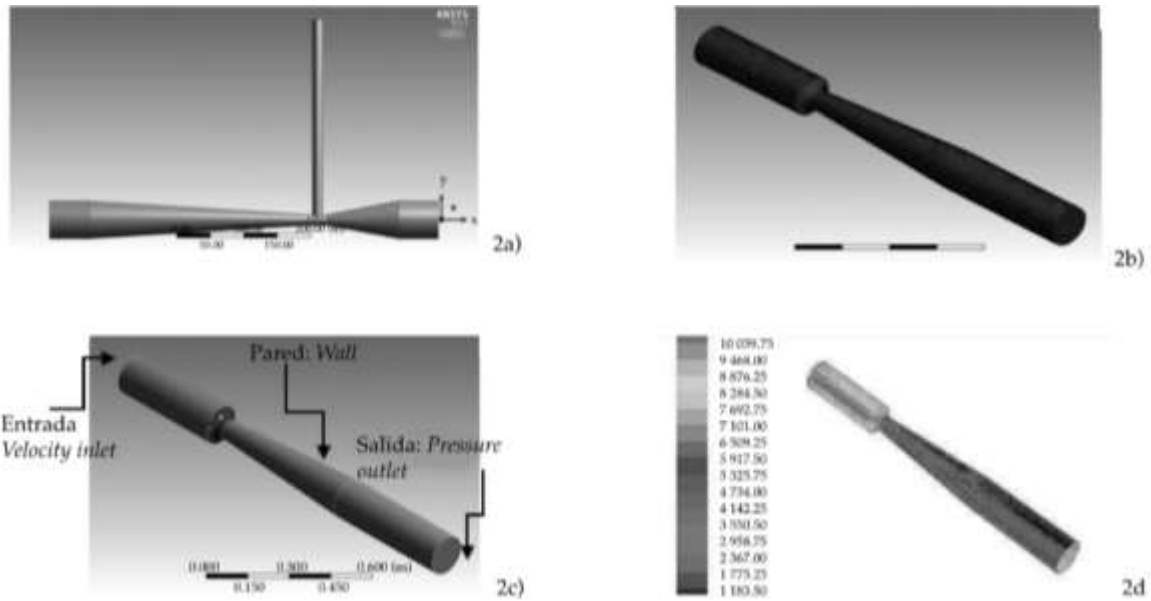


Figura 2-1 Comparación del modelo computacional por etapas (Covarrubias, Velázquez, Bustamante, Delgado, & Escalante, 2015)

“Spillway jet regime and total dissolved gas prediction with a multiphase flow model” (Wang, Marcela, & Larry, 2018) muestra un modelo numérico basado en el código abierto OpenFOAM con el fin de desarrollar regímenes de reacción y el gas disuelto total aguas abajo de los vertederos. Este modelo utiliza el método del volumen de fluido (VOF) para rastrear la interfaz entre el aire y el agua.

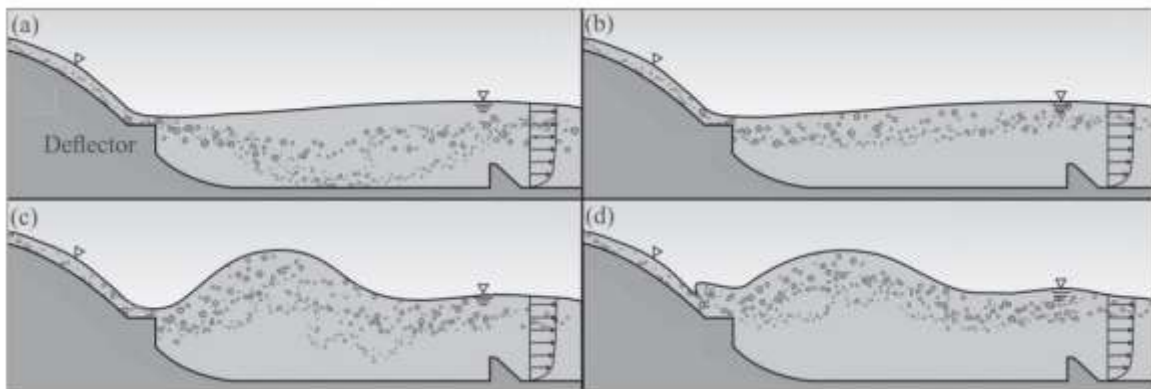


Figura 2-2 Predicción de regímenes de reacción y el gas disuelto total aguas abajo del vertedor (Wang, Marcela, & Larry, 2018)

“Assessment of particle – tracking models for dispersed particle – laden flows implemented in OpenFOAM and ANSYS FLUENT”, (Franziska, Christoph, Thomas, Friederike, & Rudiger, 2016) presenta la comparación de dos problemas de referencia para el flujo turbulento de partículas dispersas en donde se prueban y comparan estos flujos con los software CFD OpenFOAM y ANSYS FLUENT (Figura 5), además los autores comentan que los resultados de la simulación y los experimentos revelan buena concordancia.

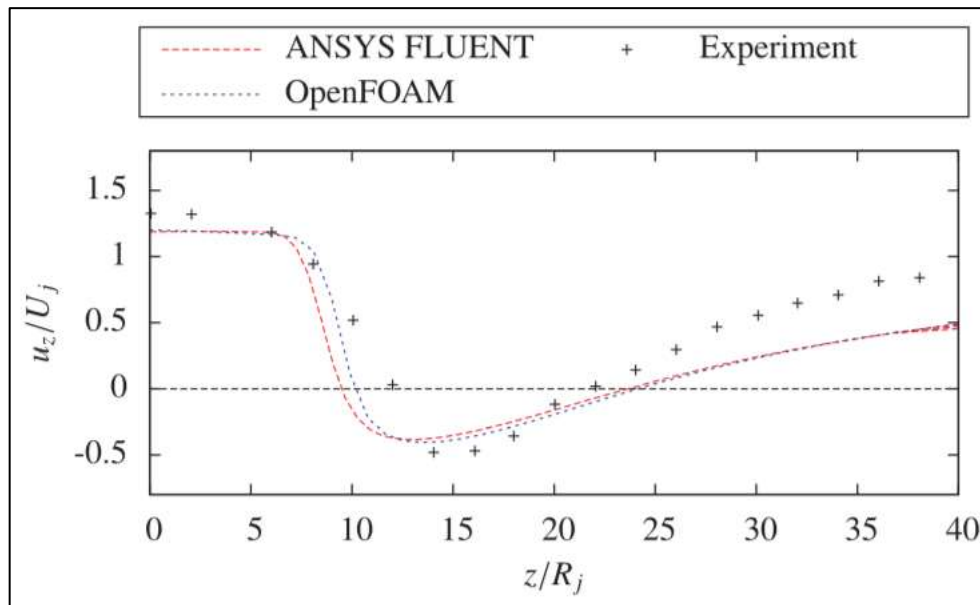


Figura 2-3. Comparación de la distribución de la velocidad a lo largo de la línea de simetría en la dirección del flujo principal a partir de simulaciones en ANSYS FLUENT, OpenFOAM y experimentos (Franziska, Christoph, Thomas, Friederike, & Rudiger, 2016).

“Simulación física y matemática del flujo en vertedores escalonados” (Aguilar Chavez & Laurel Castillo, 2011), en este trabajo de investigación se presentan modelaciones numéricas de diferentes vertedores escalonados para poder conocer la capacidad del software CFD Flow3D y poder compararlos con datos obtenidos en modelos físicos de laboratorio y datos experimentales de literatura, cabe destacar que se concluye que el software de Dinámica de Fluidos Computacional es una herramienta robusta para la revisión del funcionamiento hidráulico de grandes obras

hidráulicas, ya que los resultados muestran gran semejanza con los valores obtenidos del modelo físico de estudio.

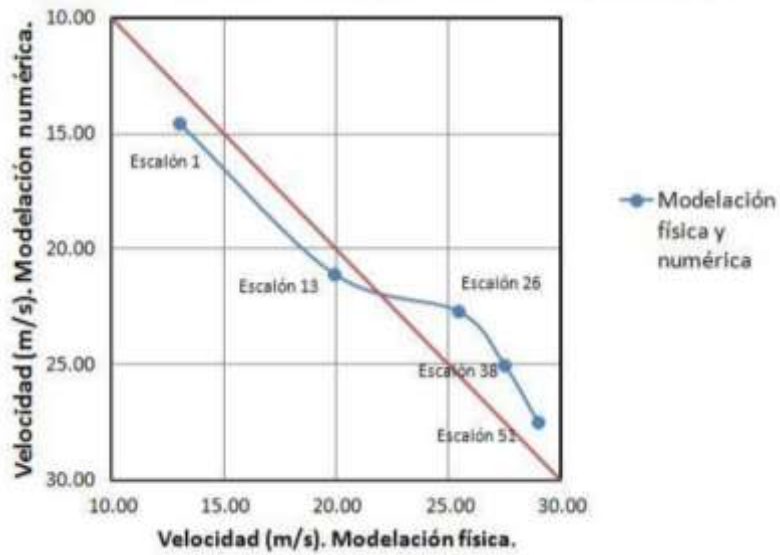


Figura 2-4 Comparación cuantitativa entre valores de velocidad medidos y simulados (Aguilar Chavez & Laurel Castillo, 2011)

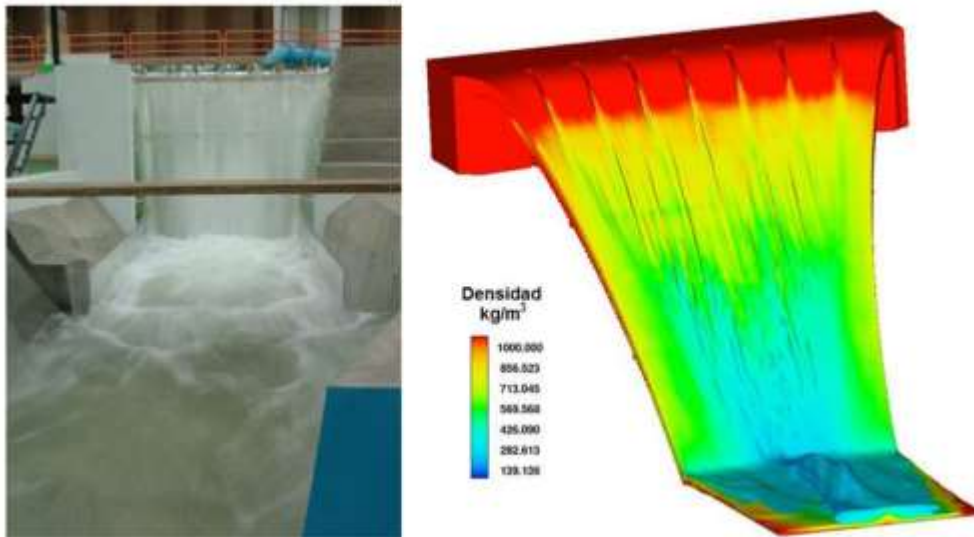


Figura 2-5 Comparación cualitativa entre distribución de concentración de aire (Aguilar Chavez & Laurel Castillo, 2011)

“Modeling contaminant transport with aerobic biodegradation in a shallow water” (Millán Barrera, Arroyo Correa, & Laurel Castillo, 2013) presentan una aplicación que se implementó en el software CFD Flow-3D, introduciéndolo en la subrutina correspondiente del código fuente y compilándolo para luego aplicarlo en el modelado de transporte de especies en un cuerpo de aguas poco profundas en dos dimensiones, cabe destacar que sus resultados obtenidos son consistentes y satisfactorios, además concluyen que la utilización de software CFD es una herramienta confiable para reproducir soluciones analíticas y también esquemas más estables.

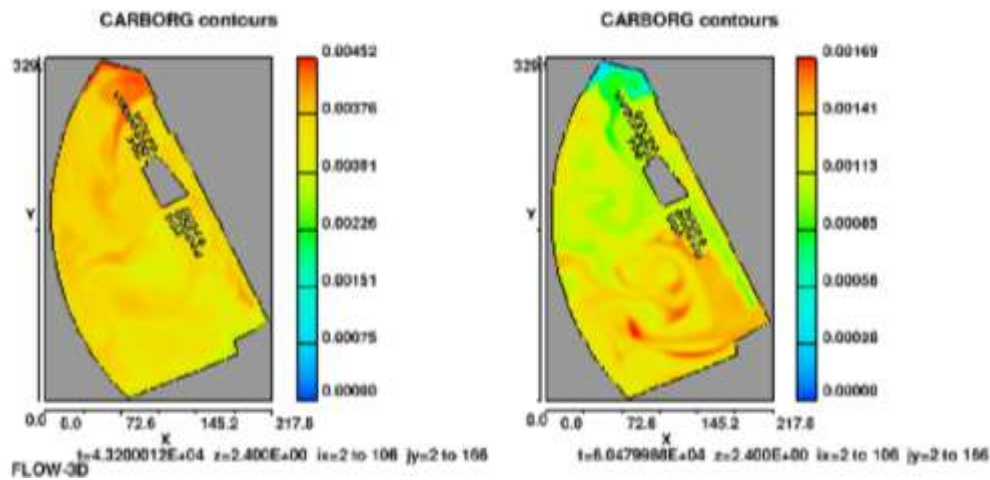
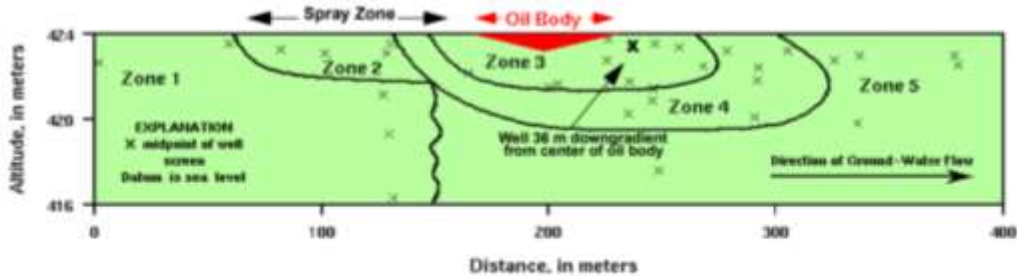


Figura 2-6 Evolución del reactivo c_2 en el lago cuando se inyectaron escalares alternativamente (Millán Barrera, Arroyo Correa, & Laurel Castillo, 2013).

“Modeling Solute – Transport and Biodegradation with BIOMOC” (I. Essaid & A. Bekins, 1997). Es un modelo numérico de transporte de solutos bidimensional que simula el transporte y la biotransformación de solutos de reacción múltiple. BIOMOC simula:

- Advección, dispersión hidrodinámica, fuentes de fluidos, retardo, descomposición y biodegradación.
- Especies de solutos múltiples y concentraciones de fase inmóvil.

- Problemas de superficie o de sección transversal unidimensionales y bidimensionales.
- Múltiples procesos de degradación y poblaciones microbianas, incluida la biodegradación secuencial.
- Tasas de transformación biológica que incluyen cinética Monod única, múltiple y mínima; e inhibición competitiva, no competitiva y de Haldane.
- Aproximaciones de orden cero o de primer orden de las tasas de biodegradación.
- El crecimiento y la descomposición microbiana por la disponibilidad de nutrientes.



Zone 1 - oxygenated uncontaminated native ground water.
 Zone 2 - reduced oxygen concentrations with refractory high molecular-weight hydrocarbons.
 Zone 3 - anoxic and with high concentrations of hydrocarbons, dissolved manganese, iron, and methane.
 Zone 4 - transitional from anoxic conditions to fully oxygenated conditions.
 Zone 5 - oxygenated water with slightly elevated dissolved inorganic and organic constituents.

Figura 2-7 Aplicación de BIOMOC (I. Essaid & A. Bekins, 1997)

“DYRESM – CAEDYM (Dynamic Reservoir Simulation Model – Computational Aquatic Ecosystem Dynamics Model)” (Hamilton, 2019). Es un modelo hidrodinámico – ecológico unidimensional que puede usarse para investigar las interacciones entre procesos físicos, químicos y biológicos que ocurren en lagos y depósitos a lo largo de escalas de tiempo que van desde días a estaciones interanuales. DYRESM utiliza un esquema de capas lagrangianas en el que el lago se representa como capas que tienen el mismo grosor y las capas se expanden o contraen teniendo en cuenta los cambios de densidad provocados por el calentamiento, la mezcla y los flujos de la superficie. El modelo DYRESM-CAEDYM

acoplado permite modelar Nitrógeno, Fósforo, Oxígeno Disuelto, Microbios, Procesos de Sedimentos y Dinámica de Fitoplancton en resolución vertical.

“CWR-ELCOM (Centre for Water Research Estuary and Lake Computer Model)” (Hodges, 2006). Es un modelo tridimensional de hidrodinámica, termodinámica y transporte de estuario y lago en 3D. El modelo resuelve las ecuaciones inestables de Navier Stokes (RANS) de Reynolds usando un método semi-implícito con la discretización cuadrática de Euvé Lagrange de advección de momento y un enfoque conservador ULTIMATE QUICKEST para el transporte escalar. Un modelo unidimensional de capas mixtas se extiende a 3D para el cierre de turbulencia de los términos de tensión de Reynolds verticales.

De acuerdo al análisis anterior se logra destacar la gran capacidad e importancia del uso de softwares CFD para la solución de una gran y diversa cantidad de problemas con relación a la Hidráulica, modelos Hidrobiológicos y en otras áreas de interés.

2.2 Fundamentos principales para el estudio de la ecuación de difusión – advección – reacción

2.2.1 Difusión

La difusión involucra un modelo matemático basado en una hipótesis fundamental o “ley”. Existen dos opciones comunes para dicha ley, lo más fundamental, la ley de difusión de Fick, utiliza un coeficiente de difusión. El segundo, que no tiene nombre formal, implica un coeficiente de transferencia de masa, un tipo constante de velocidad reversible (Cussler, 2007).

Las ideas actuales sobre la difusión se deben en gran parte a dos personas, Thomas Graham y Adolf Fick. Graham realizó importantes experimentos sobre difusión de líquidos, trabajó con soluciones diluidas. En una serie de experimentos, conectó dos botellas que contenían soluciones a diferentes concentraciones; Esperó varios días y luego separó las botellas y analizó su contenido. En otra serie de experimentos, colocó una pequeña botella que contenía una solución de concentración conocida en un frasco más grande que solo contenía agua. Después de esperar varios días, sacó la botella y analizó su contenido. Los resultados de Graham fueron simples y definitivos. Demostró que la difusión en líquidos era al menos varios miles de veces más lenta que la difusión en gases. Reconoció que el proceso de difusión se comportaba más lento a medida que avanzaba el experimento, lo cual concluyó que "la difusión necesariamente debe seguir una progresión decreciente " (Cussler, 2007).

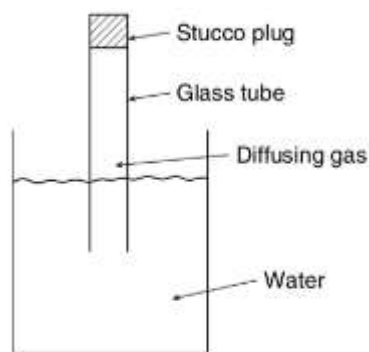


Figura 2-8 Tubo de difusión de gases de Graham (Cussler, 2007)

En 1855 en su primer artículo de difusión, Fick codificó los experimentos de Graham a través de una impresionante combinación de teorías cualitativas, analogías casuales y experimentos cuantitativos. La idea básica de Fick es la siguiente: "La difusión del material disuelto se deja completamente a la influencia de las fuerzas moleculares básicas de la misma ley para la difusión de calor en un conductor y que ya se ha aplicado con tanto éxito en la distribución de electricidad ". En otras palabras, la difusión se puede describir sobre la misma base matemática que la ley de Fourier para la conducción de calor o la ley de Ohm para la conducción eléctrica (Cussler, 2007).

Usando esta hipótesis básica, Fick desarrolló las leyes de la difusión mediante analogías con el trabajo de Fourier. Definió un flujo unidimensional total J_1 como:

$$J_1 = Aj_1 = -AD \frac{\partial c_1}{\partial z} \quad (2.1)$$

Donde A es el área a través de la cual se produce la difusión, j_1 es el flujo por unidad de área, c_1 es la concentración, z es la distancia, D es el coeficiente de difusión. Esta es la primera ley de Fick.

La segunda ley de Fick predice la forma en que la difusión causa que la concentración cambie con el tiempo, se trata de una ecuación diferencial parcial:

$$\frac{\partial c_1}{\partial t} = D \frac{\partial^2 c_1}{\partial z^2} \quad (2.2)$$

2.2.2 Advección

Los dos principales modos de transporte en la mecánica de fluidos es la advección (transporte asociado con el flujo de un fluido) y difusión (Socolofsky & Jirka, 2002).

La advección es el arrastre de la sustancia contaminante por el agua. Si solo existiera este proceso, el contaminante viajaría a la misma velocidad que el agua y la extensión ocupada por el contaminante sería constante. La advección simplemente transporta las sustancias contaminantes y se representa con la siguiente ecuación diferencial parcial:

$$\frac{\partial c_1}{\partial t} + \nabla \cdot (\mathbf{u}c_1) = D\nabla^2 c_1$$

(2.3)

En esta ecuación se asume que D es constante.

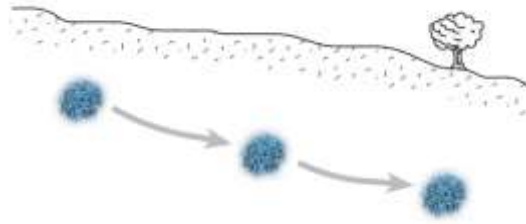


Figura 2-9 Transporte de una sustancia si se produjera solo advección (Román, 2017)

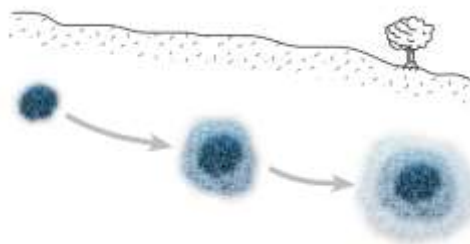


Figura 2-10 Transporte de una sustancia si se produjera solo difusión y advección (Román, 2017)

2.2.3 Término reactivo en la ecuación de difusión – advección

La transformación se define como la producción o pérdida de una determinada especie de interés a través de procesos físicos, químicos o biológicos.

Las reacciones de transformación se clasifican ampliamente como homogéneas o heterogéneas. Las reacciones homogéneas ocurren en todas partes dentro del fluido de interés. Esto significa que se distribuyen por todo el volumen de control; Por lo tanto, se representan como un término fuente o sumidero en la ecuación diferencial gobernante. Por el contrario, heterogéneo se tiene que las reacciones ocurren solo en los límites del fluido y no se distribuyen en todo el volumen de control (Socolofsky & Jirka, 2002).

La representación matemática es la siguiente:

$$\frac{\partial c_i}{\partial t} = S_i \tag{2.4}$$

Donde c_i es la masa total de la especie i y S_i es un término fuente o sumidero.

El término de reacción es simplemente la ley de velocidad cinética integrada sobre el volumen, entonces se tiene:

$$S_i = \pm R_i \tag{2.5}$$

Donde R_i es el término reactivo, el cual es una función que describe la ley de velocidad para la especie i .

Ahora se combinan estos términos para obtener la ecuación de difusión – advección – reacción.

$$\frac{\partial c_1}{\partial t} + \nabla \cdot (\mathbf{u}c_1) = D\nabla^2 c_1 \pm R$$

(2.6)

2.3 Biodegradación aerobia

Un sentido muy amplio, en la naturaleza, no hay desperdicio porque casi todo se recicla. En el sentido microbiológico, "biodegradación" significa la descomposición de todos los materiales orgánicos que se lleva a cabo mediante formas de vida que comprenden principalmente bacterias, hongos, protozoos y otros organismos. A través de este proceso biológicamente natural, los contaminantes tóxicos peligrosos se transforman en sustancias menos tóxicas o no tóxicas. Al aprovechar las comunidades microbianas, se pueden lograr las "fuerzas" naturales de la biodegradación, la reducción de desechos y la limpieza de la mayoría de los contaminantes ambientales. La parte de la actividad metabólica que produce energía consume oxígeno, lo que resulta en la formación inmediata de dióxido de carbono, agua y sales minerales en un proceso conocido como "mineralización" (Eskander & Saleh, 2017).

Biodegradación aerobia: "Es la descomposición de contaminantes orgánicos por microorganismos cuando el oxígeno está presente". La biodegradación aerobia también es conocido como respiración aeróbica (Eskander & Saleh, 2017).

Los sistemas de biodegradación consisten principalmente en el uso de microorganismos (levaduras, hongos o bacterias) existentes en el medio ambiente para descomponer o degradar sustancias peligrosas en sustancias de carácter menos tóxico o bien inocuas para el medio ambiente y la salud humana, dentro de las ventajas y desventajas de la biodegradación aerobia en el tratamiento de agua se muestran a continuación (Blanco, 2010).

Ventajas:

- Favorecen al medio ambiente.
- El proceso es más económico que los tratamientos químicos.
- Generan menos subproductos.
- Es posible llegar a la mineralización (proceso biológico que ocurre mediante la conversión de la materia orgánica a un estado inorgánico a través de la acción de microorganismos) total de los contaminantes y con esto permite que los nutrientes puedan volver al suelo en forma asimilable para las plantas.

Desventajas

- Se presenta inhibición por altas concentraciones y/o acumulación de metabolitos (productos que quedan después de la descomposición o metabolismo) tóxicos para los microorganismos.
- Se requieren tiempos considerablemente altos (de días a meses) a los requeridos por otro tipo de tratamientos, como los químicos.
- Se generan lodos derivados del crecimiento microbiano.

Para incluir estos procesos de biodegradación aerobia en modelos numéricos, el primer paso es expresarlos en modelos matemáticos para representar la relación entre distintas variables, parámetros y restricciones. Lo siguiente es la solución y validación del modelo para entender el fenómeno físico.

3 Estado del arte

3.1 Ecuación de difusión – advección – reacción

En 1989 Celia M. A., J. S. Kindred e I. Herrera presentan un procedimiento de solución numérica para la ecuación de advección – difusión – reacción en medios porosos que consiste en un método de función de prueba óptima (OTF, *optimal test function*, en inglés), además muestran un ejemplo que involucra el transporte de dos especies a través de los términos de reacción, este sistema se interpreta como el transporte de dos sustratos en presencia de una población biológica estacionaria (biomasa), para el ejemplo se expresa que el sustrato es un contaminante orgánico y el oxígeno disuelto como compuesto limitante, lo cual lleva a un problema de transporte con biodegradación aerobia.

En la Figura 3-1 se muestra la comparación de solución numérica y solución analítica. Las curvas solidas indican soluciones numéricas de OTF y los símbolos indican valores de la solución analítica.

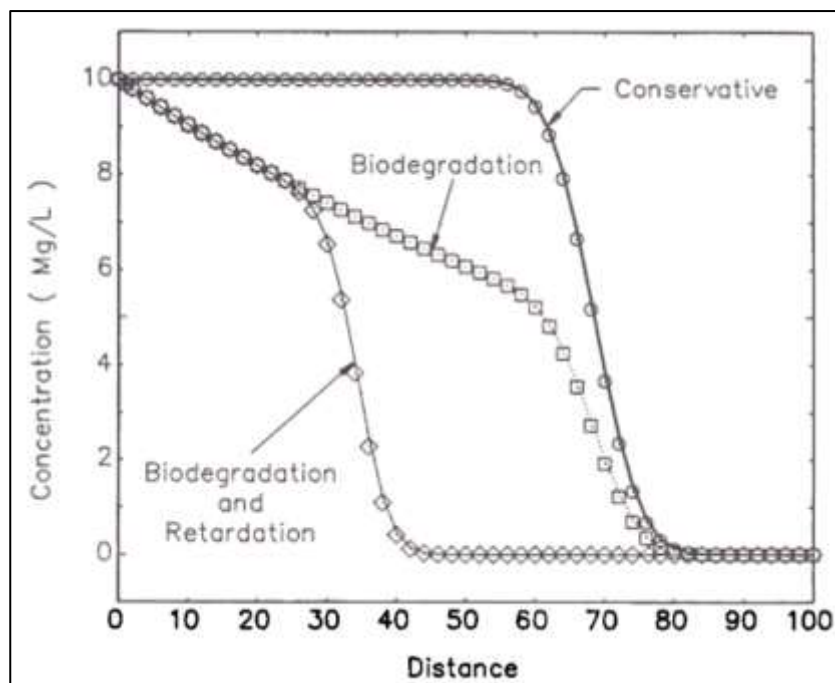


Figura 3-1 Comparación de solución numérica OTF y solución analítica (Celia, Kindred, & Ismael, June 1989)

La metodología que presenta Celia M. A., J. S. Kindred e I. Herrera, (1989) se ha utilizado para desarrollar diferentes investigaciones por parte del Servicio Geológico de los Estados Unidos (USGS), dentro de los trabajos de investigación se encuentra el denominado “*BIOMOC, A Multispecies Solute – Transport Model with Biodegradation*”, 1997. También han sido resueltos mediante técnicas relativamente nuevas como por ejemplo el método Euleriano Lagrangiano Localizado Adjunto (ELLAM) (Arroyo Correa, 2005).

Celia M. A., J. S. Kindred e I. Herrera, (1989) presentan un ejemplo que involucra el transporte de dos especies, acoplado a través de términos de reacción, en presencia de una población biológica estacionaria. Los sustratos disueltos pueden ser un contaminante orgánico y oxígeno disuelto, lo que lleva a un problema de transporte de biodegradación aerobia.

Las ecuaciones que gobiernan para este sistema son:

$$\frac{\partial c_1}{\partial t} + \mathbf{u} \frac{\partial c_1}{\partial x} - D \frac{\partial^2 c_1}{\partial x^2} + K_1(c_1, X_1)c_1 = f_1(c_2, X_1) \quad (3.1)$$

$$\frac{\partial c_2}{\partial t} + \mathbf{u} \frac{\partial c_2}{\partial x} - D \frac{\partial^2 c_2}{\partial x^2} + K_2(c_2, X_1)c_2 = f_2(c_1, X_1) \quad (3.2)$$

Las formas funcionales basadas en la analogía con la biodegradación son:

$$K_1(c_1, X_1) = \left(\frac{V_m^1 X_1}{K_h^1 + c_1} \right) \delta_1 \quad (3.3)$$

$$K_2(c_2, X_1) = \left(\frac{V_m^2 X_1}{K_h^2 + c_2} \right) \delta_2 \quad (3.4)$$

$$f_1(c_2, X_1) = -k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + c_2} \right) \delta_2 c_2 = -k_{12} K_2 c_2 \quad (3.5)$$

$$f_2(c_1, X_1) = -k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + c_1} \right) \delta_1 c_1 = -k_{21} K_1 c_1 \quad (3.6)$$

Donde:

V_m^i : es la tasa máxima de absorción para las especies i .

K_h^i : es la constante media de saturación para especies i .

k_{ij} : es el coeficiente de relación de rendimiento para la especie i cuando la especie j es limitante.

δ_i : es igual a 1 si el sustrato i es limitante y 0 de otra manera

Las ecuaciones (3.1) y (3.2) permiten que cualquiera de los sustratos sea limitante.

3.2 Análisis diferencial de la ecuación Hidrodinámica

El análisis diferencial proporciona una descripción muy detallada del flujo, ya que busca resolver el movimiento de las partículas de fluido que están ocupando en un instante el volumen de control, para lo cual obtiene como solución de las ecuaciones que gobiernan el comportamiento. Su aplicación se ve dificultada al tratar con un modelo matemático en ecuaciones en derivadas parciales no lineales y con el fenómeno de la turbulencia. Son muy escasos los flujos, todos ellos en régimen laminar (sin turbulencia) y en geometrías sencillas, en los que las ecuaciones se pueden simplificar hasta reducirlas a unas que puedan resolverse de manera analítica. El análisis diferencial de aquellos flujos que posean geometrías complejas y/o estén en régimen turbulento, requerirá de la utilización de técnicas numéricas avanzadas de resolución por computadora de las ecuaciones diferenciales (Dinámica de Fluidos Computacional) (Rivas, 2007).

Para las investigaciones sobre Dinámica de Fluidos Computacionales se han realizado una gran cantidad de investigaciones, teniendo como punto principal resolver numéricamente las ecuaciones de Navier Stokes que incluyen el transporte de masa, momentum y energía en los fluidos.

Las ecuaciones que gobiernan el control de un fluido newtoniano compresible son las siguientes (Versteeg & W, 2007):

Continuidad

$$\frac{\partial \rho}{\partial t} + \text{div}(\rho \mathbf{u}) = 0 \quad (3.7)$$

x-momentum

$$\frac{\partial(\rho u)}{\partial t} + \text{div}(\rho u \mathbf{u}) = -\frac{\partial p}{\partial x} + \text{div}(\mu \text{ grad } u) + S_{Mx} \quad (3.8)$$

y-momentum

$$\frac{\partial(\rho v)}{\partial t} + \text{div}(\rho v \mathbf{u}) = -\frac{\partial p}{\partial y} + \text{div}(\mu \text{ grad } v) + S_{My} \quad (3.9)$$

z-momentum

$$\frac{\partial(\rho w)}{\partial t} + \text{div}(\rho w \mathbf{u}) = -\frac{\partial p}{\partial z} + \text{div}(\mu \text{ grad } w) + S_{Mz} \quad (3.10)$$

Energía

$$\frac{\partial(\rho i)}{\partial t} + \text{div}(\rho i \mathbf{u}) = -p \text{div } \mathbf{u} + \text{div}(k \text{ grad } T) + \Phi + S_i \quad (3.11)$$

Donde S_M es la fuente de impulso y Φ es la función de disipación.

Ecuaciones de estado

$$p = p(\rho, T) \quad e \quad i = i(\rho, T) \quad (3.12)$$

Para un fluido incompresible e isotérmico la densidad ρ y la temperatura T son constantes y por lo tanto las ecuaciones (3.7), (3.8), (3.9) y (3.10) se convierten en:

Continuidad

$$\text{div } \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \quad (3.13)$$

x-momentum

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right) = -\frac{\partial p}{\partial x} + \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) + S_{Mx} \quad (3.14)$$

y-momentum

$$\rho \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} \right) = -\frac{\partial p}{\partial y} + \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right) + S_{My} \quad (3.15)$$

z-momentum

$$\rho \left(\frac{\partial w}{\partial t} + u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} \right) = -\frac{\partial p}{\partial z} + \mu \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) + S_{Mz} \quad (3.16)$$

Los efectos de las tensiones superficiales se explican explícitamente, los términos fuente S_{Mx} , S_{My} y S_{Mz} se incluyen en las contribuciones a las fuerzas del cuerpo. Por ejemplo, la fuerza del cuerpo debido únicamente a la gravedad se modelaría de la siguiente manera: $S_{Mx} = 0$, $S_{My} = 0$ y $S_{Mz} = -\rho g$.

En el flujo de fluidos compresibles, las ecuaciones de estado proporcionan el vínculo entre la ecuación de energía por un lado y las ecuaciones de conservación de masa y de impulso por el otro. Este enlace surge a través de la posibilidad de variaciones de densidad como resultado de variaciones de presión y temperatura en el campo de flujo (Versteeg & W, 2007).

Los líquidos y gases que fluyen a bajas velocidades se comportan como fluidos incompresibles. Sin variaciones de densidad, no existe un vínculo entre la ecuación de energía y las ecuaciones de conservación de masa y de momento. El campo de flujo a menudo se puede resolver considerando solo la conservación de masa y las ecuaciones de momento. La ecuación de energía solo debe resolverse junto con las demás si el problema implica la transferencia de calor (Versteeg & W, 2007).

Las ecuaciones que gobiernan el flujo de fluidos se convierten en ecuaciones diferenciales parciales (EDP) cuando se aplican para controlar volúmenes con un tamaño infinitesimal en un flujo de fluido. Sin embargo, es casi imposible aplicar soluciones analíticas a estas ecuaciones debido a que las ecuaciones no son lineales, se encuentran en tres dimensiones (3D) y presentan la existencia de dominios de soluciones complejas (Date, 2005).

Es por esta razón que los métodos numéricos son necesarios en este punto y el método de discretización se utiliza para aproximar las EDP mediante el uso de ecuaciones algebraicas. En el método de discretización, las ecuaciones se resuelven en dominios de extensión limitada para obtener resultados en ubicaciones discretas en el espacio y el tiempo.

3.3 OpenFOAM

OpenFOAM (*Open Field Operation and Manipulation*) es un software CFD de código abierto y gratuito que permite a los usuarios, como ingenieros, científicos, académicos y organizaciones comerciales, resolver una amplia gama de problemas, como Flujos de fluidos complejos, transferencia de calor, electromagnetismo, entre otros. Incluye bibliotecas y utilidades de pre - procesamiento, procesamiento y post - procesamiento.

OpenFOAM

The Open Source CFD Toolbox

Figura 3-2 OpenFOAM, software CFD de código abierto (*OpenFOAM, 2018*)

La estructura de OpenFOAM describe la amplia funcionalidad disponible, la ejecución de aplicaciones y procesamiento de los resultados. La interfaz para el pre - procesamiento y post - procesamiento son, en sí mismas utilidades de OpenFOAM, lo que garantiza un manejo de datos consistente en todos los entornos.

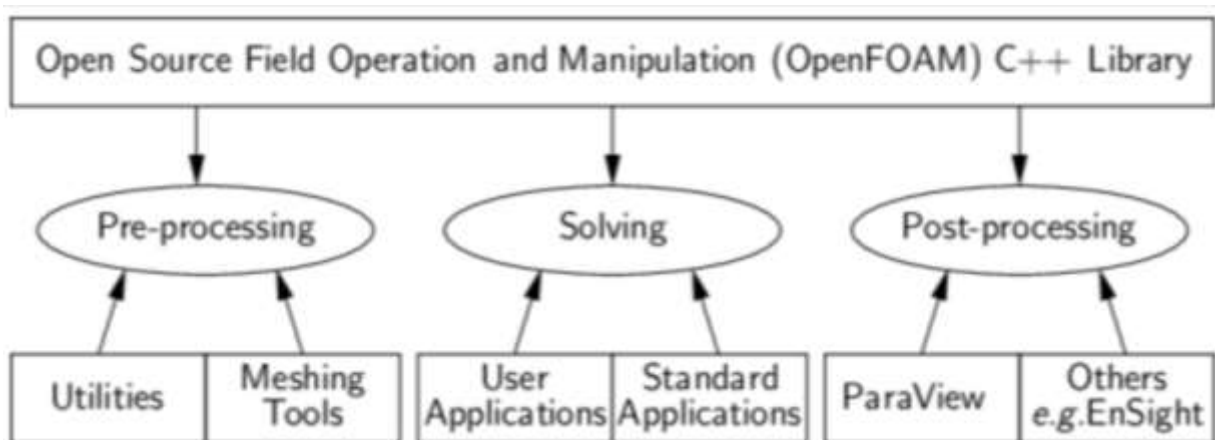


Figura 3-3 Estructura general de OpenFOAM (*OpenFOAM, 2018*)

Para el pre - procesamiento en la generación de mallas puede ser compatible con otras plataformas como SALOME, GMSH, ANSYS Fluent, entre otros. El usuario puede generar mallas utilizando otros paquetes y convertirlos al formato que utiliza OpenFOAM.

El código abierto permite que los usuarios modifiquen los archivos existentes de OpenFOAM de la biblioteca de C ++ para que los usuarios consigan compilar sus propios solucionadores, cabe destacar que tiene su propia sintaxis.

Debido a la amplia gama de aplicaciones de OpenFOAM, se debe elegir el solucionador disponible adecuado para una aplicación en particular. Los solucionadores estándar se pueden encontrar en la Guía del usuario de OpenFOAM y se dividen en varias categorías, por ejemplo. Flujo incompresible, flujo multifase, combustión y flujos de seguimiento de partículas. Los solucionadores en OpenFOAM utilizan esquemas numéricos. Para poder usar estos esquemas numéricos, cada dominio computacional debe dividirse en celdas de cuadrícula y cada centro de la celda de cuadrícula se usa para cálculos.

3.4 SALOME

SALOME es un software de código abierto que proporciona una plataforma genérica de pre - procesamiento y post - procesamiento para la simulación numérica. Se basa en una arquitectura abierta y flexible hecha de componentes reutilizables. Su objetivo es ser un generador de red rápido y fácil de usar para proporcionar capacidades avanzadas de visualización. También es compatible con OpenFOAM y las mallas que se crean se pueden convertir fácilmente al lenguaje de OpenFOAM.



Figura 3-4 Visualización del software SALOME (SALOME, 2018)

3.5 Python

Python es un lenguaje de programación que permite trabajar rápidamente e integrar sistemas de manera efectiva (Python, 2018), algunas características del lenguaje Python son las siguientes:

- Se trata de un lenguaje interpretado, de modo que puede usarse de modo interactivo o bien a través de la construcción de un *script*.
- Cualquier *script* puede ser utilizado como un módulo, es decir, una función u objeto que ya se construyó en algún archivo y puede ser importada desde cualquier otro.
- El lenguaje está orientado a objetos, los cuales se implementan de manera sencilla.
- Cuenta con librerías o módulos para cálculo numéricos (*Numeric, numpy*) y aplicaciones científicas (*scipy*) así como para hacer aplicaciones interactivas y gráficas cuya sintaxis es simple.
- Es un software libre y existe para todos los sistemas operativos.
- Entre otros.

3.6 ParaView

ParaView es una aplicación de visualización y análisis de datos multiplataforma de código abierto. Los usuarios de ParaView pueden crear rápidamente visualizaciones para analizar sus datos utilizando técnicas cualitativas y cuantitativas. La exploración de datos se puede realizar interactivamente en 3D o mediante programación utilizando las capacidades de procesamiento por lotes de ParaView (ParaView, 2018).

ParaView fue desarrollado para analizar conjuntos de datos extremadamente grandes utilizando recursos de computación de memoria distribuida. Puede ejecutarse en supercomputadoras para analizar conjuntos de datos de tamaño de gran escala, así como en computadoras portátiles para datos más pequeños, se ha convertido en una herramienta integral en muchos laboratorios nacionales,

universidades e industrias, y ha ganado varios premios relacionados con computación de alto rendimiento (ParaView, 2018).

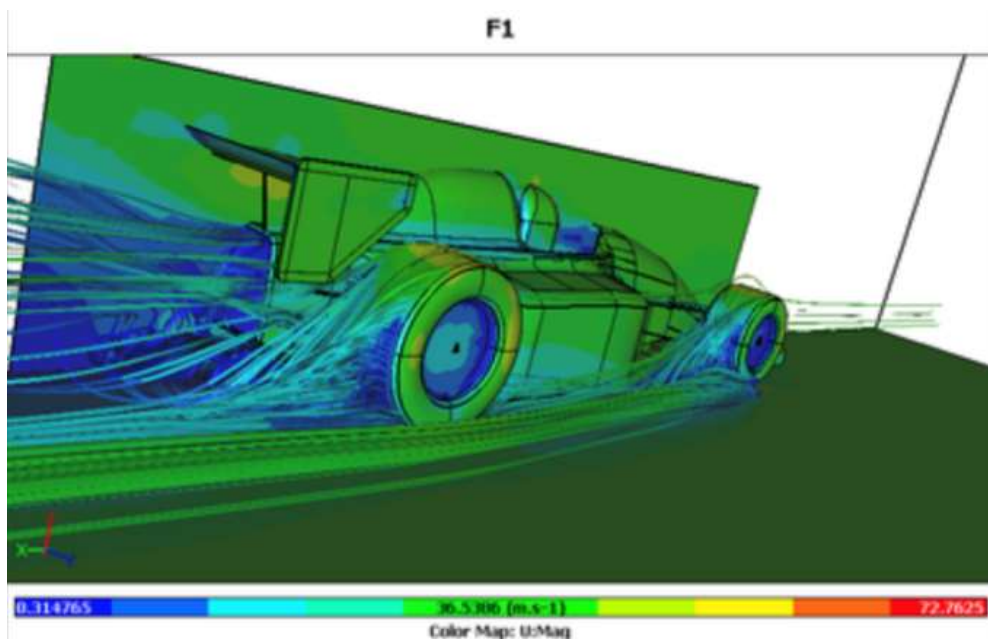


Figura 3-5 Simulación de flujo utilizando OpenFOAM y ParaView para visualización (ParaView, 2018).

4 Construcción del modelo numérico 3D con términos de biodegradación aerobia

El modelo numérico para solucionar problemas de transporte de dos especies con términos de biodegradación se mostrará a continuación con una aplicación para solucionar las ecuaciones. Las ecuaciones que gobiernan el sistema son:

$$\frac{\partial c_1}{\partial t} + \nabla \cdot (\mathbf{u}c_1) - D\nabla^2 c_1 + K_1(c_1, X_1)c_1 = f_1(c_2, X_1) \quad (4.1)$$

$$\frac{\partial c_2}{\partial t} + \nabla \cdot (\mathbf{u}c_2) - D\nabla^2 c_2 + K_2(c_2, X_1)c_2 = f_2(c_1, X_1) \quad (4.2)$$

Paso 1. Discretización de los términos de la ecuación (4.1), donde se tiene un parámetro i para los cambios en \mathbf{x} , j para los cambios en \mathbf{y} y k para los cambios en \mathbf{z} .

- Diferenciación temporal hacia adelante

$$\frac{\partial c_1}{\partial t} \cong \frac{C_{1,i,j,k}^{n+1} - C_{1,i,j,k}^n}{\Delta t}$$

- Diferenciación espacial central

$$\frac{\partial c_1}{\partial x} \cong \frac{C_{1,i+1,j,k}^n - C_{1,i-1,j,k}^n}{2\Delta x}$$

$$\frac{\partial c_1}{\partial y} \cong \frac{C_{1,i,j+1,k}^n - C_{1,i,j-1,k}^n}{2\Delta y}$$

$$\frac{\partial c_1}{\partial z} \cong \frac{C_{1,i,j,k+1}^n - C_{1,i,j,k-1}^n}{2\Delta z}$$

- Discretización de la segunda derivada espacial

$$\frac{\partial^2 c_1}{\partial x^2} \cong \frac{C_{1i+1,j,k}^n - 2C_{1i,j,k}^n + C_{1i-1,j,k}^n}{(\Delta x)^2}$$

$$\frac{\partial^2 c_1}{\partial y^2} \cong \frac{C_{1i,j+1,k}^n - 2C_{1i,j,k}^n + C_{1i,j-1,k}^n}{(\Delta y)^2}$$

$$\frac{\partial^2 c_1}{\partial z^2} \cong \frac{C_{1i,j,k+1}^n - 2C_{1i,j,k}^n + C_{1i,j,k-1}^n}{(\Delta z)^2}$$

Paso 2. Discretización de los términos de la ecuación (4.2)

- Diferenciación temporal hacia adelante

$$\frac{\partial c_2}{\partial t} \cong \frac{C_{2i,j}^{n+1} - C_{2i,j}^n}{\Delta t}$$

- Diferenciación espacial central

$$\frac{\partial c_2}{\partial x} \cong \frac{C_{2i+1,j,k}^n - C_{2i-1,j,k}^n}{2\Delta x}$$

$$\frac{\partial c_2}{\partial y} \cong \frac{C_{2i,j+1,k}^n - C_{2i,j-1,k}^n}{2\Delta y}$$

$$\frac{\partial c_2}{\partial z} \cong \frac{C_{2i,j,k+1}^n - C_{2i,j,k-1}^n}{2\Delta z}$$

- Discretización de la segunda derivada espacial

$$\frac{\partial^2 c_2}{\partial x^2} \cong \frac{C_{2i+1,j,k}^n - 2C_{2i,j,k}^n + C_{2i-1,j,k}^n}{(\Delta x)^2}$$

$$\frac{\partial^2 c_2}{\partial y^2} \cong \frac{C_{2i,j+1,k}^n - 2C_{2i,j,k}^n + C_{2i,j-1,k}^n}{(\Delta y)^2}$$

$$\frac{\partial^2 c_2}{\partial z^2} \cong \frac{C_{2i,j,k+1}^n - 2C_{2i,j,k}^n + C_{2i,j,k-1}^n}{(\Delta z)^2}$$

Paso 3. Sustitución de las ecuaciones del Paso 1 y Paso 2 en las ecuaciones (4.1) y (4.2).

La sustitución queda de la siguiente manera:

Para la ecuación (4.1) se tiene:

$$\begin{aligned}
& \frac{C_{1,i,j,k}^{n+1} - C_{1,i,j,k}^n}{\Delta t} + Ux \frac{C_{1,i+1,j,k}^n - C_{1,i-1,j,k}^n}{2\Delta x} + Uy \frac{C_{1,i,j+1,k}^n - C_{1,i,j-1,k}^n}{2\Delta y} \\
& + Uz \frac{C_{1,i,j,k+1}^n - C_{1,i,j,k-1}^n}{2\Delta z} - D \frac{C_{1,i+1,j,k}^n - 2C_{1,i,j,k}^n + C_{1,i-1,j,k}^n}{(\Delta x)^2} \\
& - D \frac{C_{1,i,j+1,k}^n - 2C_{1,i,j,k}^n + C_{1,i,j-1,k}^n}{(\Delta y)^2} \\
& - D \frac{C_{1,i,j,k+1}^n - 2C_{1,i,j,k}^n + C_{1,i,j,k-1}^n}{(\Delta z)^2} + \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j,k}^n} \right) \delta_1 C_{1,i,j,k}^n \\
& = -k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j,k}^n} \right) \delta_2 C_{2,i,j,k}^n
\end{aligned}$$

Para la ecuación (4.2) se tiene:

$$\begin{aligned}
& \frac{C_{2,i,j,k}^{n+1} - C_{2,i,j,k}^n}{\Delta t} + Ux \frac{C_{2,i+1,j,k}^n - C_{2,i-1,j,k}^n}{2\Delta x} + Uy \frac{C_{2,i,j+1,k}^n - C_{2,i,j-1,k}^n}{2\Delta y} \\
& + Uz \frac{C_{2,i,j,k+1}^n - C_{2,i,j,k-1}^n}{2\Delta z} - D \frac{C_{2,i+1,j,k}^n - 2C_{2,i,j,k}^n + C_{2,i-1,j,k}^n}{(\Delta x)^2} \\
& - D \frac{C_{2,i,j+1,k}^n - 2C_{2,i,j,k}^n + C_{2,i,j-1,k}^n}{(\Delta y)^2} \\
& - D \frac{C_{2,i,j,k+1}^n - 2C_{2,i,j,k}^n + C_{2,i,j,k-1}^n}{(\Delta z)^2} + \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j,k}^n} \right) \delta_2 C_{2,i,j,k}^n \\
& = -k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j,k}^n} \right) \delta_1 C_{1,i,j,k}^n
\end{aligned}$$

Paso 4. Presentación del modelo discreto para la solución de las ecuaciones (4.1) y (4.2)

Para la ecuación (4.1) se despeja $C_{1,i,j,k}^{n+1}$:

$$\begin{aligned}
& \frac{C_{1,i,j,k}^{n+1} - C_{1,i,j,k}^n}{\Delta t} + Ux \frac{C_{1,i+1,j,k}^n - C_{1,i-1,j,k}^n}{2\Delta x} + Uy \frac{C_{1,i,j+1,k}^n - C_{1,i,j-1,k}^n}{2\Delta y} \\
& + Uz \frac{C_{1,i,j,k+1}^n - C_{1,i,j,k-1}^n}{2\Delta z} - D \frac{C_{1,i+1,j,k}^n - 2C_{1,i,j,k}^n + C_{1,i-1,j,k}^n}{(\Delta x)^2} \\
& - D \frac{C_{1,i,j+1,k}^n - 2C_{1,i,j,k}^n + C_{1,i,j-1,k}^n}{(\Delta y)^2} \\
& - D \frac{C_{1,i,j,k+1}^n - 2C_{1,i,j,k}^n + C_{1,i,j,k-1}^n}{(\Delta z)^2} + \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j,k}^n} \right) \delta_1 C_{1,i,j,k}^n \\
& = -k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j,k}^n} \right) \delta_2 C_{2,i,j,k}^n
\end{aligned}$$

$$\begin{aligned}
& \frac{C_{1,i,j,k}^{n+1} - C_{1,i,j,k}^n}{\Delta t} \\
& = -Ux \frac{C_{1,i+1,j,k}^n - C_{1,i-1,j,k}^n}{2\Delta x} - Uy \frac{C_{1,i,j+1,k}^n - C_{1,i,j-1,k}^n}{2\Delta y} \\
& - Uz \frac{C_{1,i,j,k+1}^n - C_{1,i,j,k-1}^n}{2\Delta z} + D \frac{C_{1,i+1,j,k}^n - 2C_{1,i,j,k}^n + C_{1,i-1,j,k}^n}{(\Delta x)^2} \\
& + D \frac{C_{1,i,j+1,k}^n - 2C_{1,i,j,k}^n + C_{1,i,j-1,k}^n}{(\Delta y)^2} \\
& + D \frac{C_{1,i,j,k+1}^n - 2C_{1,i,j,k}^n + C_{1,i,j,k-1}^n}{(\Delta z)^2} - \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j,k}^n} \right) \delta_1 C_{1,i,j,k}^n \\
& - k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j,k}^n} \right) \delta_2 C_{2,i,j,k}^n
\end{aligned}$$

$$\begin{aligned}
C_{1,i,j,k}^{n+1} = & \left[-Ux \frac{C_{1,i+1,j,k}^n - C_{1,i-1,j,k}^n}{2\Delta x} - Uy \frac{C_{1,i,j+1,k}^n - C_{1,i,j-1,k}^n}{2\Delta y} \right. \\
& - Uz \frac{C_{1,i,j,k+1}^n - C_{1,i,j,k-1}^n}{2\Delta z} + D \frac{C_{1,i+1,j,k}^n - 2C_{1,i,j,k}^n + C_{1,i-1,j,k}^n}{(\Delta x)^2} \\
& + D \frac{C_{1,i,j+1,k}^n - 2C_{1,i,j,k}^n + C_{1,i,j-1,k}^n}{(\Delta y)^2} \\
& + D \frac{C_{1,i,j,k+1}^n - 2C_{1,i,j,k}^n + C_{1,i,j,k-1}^n}{(\Delta z)^2} - \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j,k}^n} \right) \delta_1 C_{1,i,j,k}^n \\
& \left. - k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j,k}^n} \right) \delta_2 C_{2,i,j,k}^n \right] \Delta t + C_{1,i,j,k}^n
\end{aligned} \tag{4.3}$$

Para la ecuación (4.2) se despeja $C_{2,i,j,k}^{n+1}$:

$$\begin{aligned}
& \frac{C_{2,i,j,k}^{n+1} - C_{2,i,j,k}^n}{\Delta t} + Ux \frac{C_{2,i+1,j,k}^n - C_{2,i-1,j,k}^n}{2\Delta x} + Uy \frac{C_{2,i,j+1,k}^n - C_{2,i,j-1,k}^n}{2\Delta y} \\
& + Uz \frac{C_{2,i,j,k+1}^n - C_{2,i,j,k-1}^n}{2\Delta z} - D \frac{C_{2,i+1,j,k}^n - 2C_{2,i,j,k}^n + C_{2,i-1,j,k}^n}{(\Delta x)^2} \\
& - D \frac{C_{2,i,j+1,k}^n - 2C_{2,i,j,k}^n + C_{2,i,j-1,k}^n}{(\Delta y)^2} \\
& - D \frac{C_{2,i,j,k+1}^n - 2C_{2,i,j,k}^n + C_{2,i,j,k-1}^n}{(\Delta z)^2} + \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j,k}^n} \right) \delta_2 C_{2,i,j,k}^n \\
& = -k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j,k}^n} \right) \delta_1 C_{1,i,j,k}^n
\end{aligned}$$

$$\begin{aligned}
& \frac{C_{2i,j,k}^{n+1} - C_{2i,j,k}^n}{\Delta t} \\
&= -Ux \frac{C_{2i+1,j,k}^n - C_{2i-1,j,k}^n}{2\Delta x} - Uy \frac{C_{2i,j+1,k}^n - C_{2i,j-1,k}^n}{2\Delta y} \\
&- Uz \frac{C_{2i,j,k+1}^n - C_{2i,j,k-1}^n}{2\Delta z} + D \frac{C_{2i+1,j,k}^n - 2C_{2i,j,k}^n + C_{2i-1,j,k}^n}{(\Delta x)^2} \\
&+ D \frac{C_{2i,j+1,k}^n - 2C_{2i,j,k}^n + C_{2i,j-1,k}^n}{(\Delta y)^2} \\
&+ D \frac{C_{2i,j,k+1}^n - 2C_{2i,j,k}^n + C_{2i,j,k-1}^n}{(\Delta z)^2} - \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i,j,k}^n} \right) \delta_2 C_{2i,j,k}^n \\
&- k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i,j,k}^n} \right) \delta_1 C_{1i,j,k}^n \\
C_{2i,j,k}^{n+1} &= \left[-Ux \frac{C_{2i+1,j,k}^n - C_{2i-1,j,k}^n}{2\Delta x} - Uy \frac{C_{2i,j+1,k}^n - C_{2i,j-1,k}^n}{2\Delta y} \right. \\
&- Uz \frac{C_{2i,j,k+1}^n - C_{2i,j,k-1}^n}{2\Delta z} + D \frac{C_{2i+1,j,k}^n - 2C_{2i,j,k}^n + C_{2i-1,j,k}^n}{(\Delta x)^2} \\
&+ D \frac{C_{2i,j+1,k}^n - 2C_{2i,j,k}^n + C_{2i,j-1,k}^n}{(\Delta y)^2} \\
&+ D \frac{C_{2i,j,k+1}^n - 2C_{2i,j,k}^n + C_{2i,j,k-1}^n}{(\Delta z)^2} - \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i,j,k}^n} \right) \delta_2 C_{2i,j,k}^n \\
&\left. - k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i,j,k}^n} \right) \delta_1 C_{1i,j,k}^n \right] \Delta t + C_{2i,j,k}^n
\end{aligned} \tag{4.4}$$

Una vez definido el modelo de solución se elaboró el código de programación en Python (2019). Para el modelo de solución de las ecuaciones (4.3) y (4.4) se establece que si $C_{2i,j,k}^n > C_{1i,j,k}^n$ se tendrá $\delta_1 = 1$ y $\delta_2 = 0$ mientras que cuando $C_{1i,j,k}^n > C_{2i,j,k}^n$ se tendrá $\delta_1 = 0$ y $\delta_2 = 1$, con este análisis se observan si las reacciones son limitantes o no limitantes en el comportamiento de las ecuaciones que gobiernan el proceso de transporte de dos especies con términos de biodegradación. Cabe mencionar que la construcción del modelo 3D servirá también para el desarrollo de los modelos en 1D y 2D.

4.1 Verificación del modelo numérico con términos de biodegradación aerobia.

De acuerdo a diferentes investigaciones sobre el proceso de transporte de contaminantes con términos de biodegradación aerobia surge la necesidad de seguir complementando y aportando modelos de simulación para fines de estudio y aplicación. Una aportación significativa sería la utilización de softwares libres y gratuitos para la solución de este tipo de problemas con diferentes geometrías y mallas.

Para verificar si la programación de las ecuaciones es adecuada en el software Python, se desarrolló la comparación de los resultados del modelo unidimensional de dos especies, en este caso: oxígeno disuelto y un contaminante orgánico informados por Celia M. A., J. S. Kindred e I. Herrera, (1989).

A continuación, se presenta la verificación con los siguientes datos:

$$V_m^i = 1.0 \text{ días}^{-1}$$

$$i = 1,2$$

$$K_h^i = 0.1 \text{ mg/l}$$

$$i = 1,2$$

$$k_{12} = 2.0$$

$$k_{21} = 0.5$$

c_1 : Oxígeno disuelto

c_2 : Contaminante orgánico

Las condiciones iniciales se asignan como:

$$c_1(x, 0) = 3.0 \text{ mg/l}$$

$$c_2(x, 0) = 0.0 \text{ mg/l}$$

Las condiciones de frontera son:

$$c_1(0, t) = 3.0 \text{ mg/l}$$

$$c_2(0, t) = 10.0 \text{ mg/l}$$

$$\frac{\partial c_1}{\partial x}(L, t) = \frac{\partial c_2}{\partial x}(L, t) = 0$$

El dominio de longitud es $L = 100 \text{ m}$

La especie estacionaria se fija en:

$$X_1 = 0.2 \text{ mg/l}$$

Los parámetros de flujo son:

$$U = 1.0 \text{ m/día}$$

$$D = 0.2 \text{ m}^2/\text{día}$$

Tiempo de simulación:

$$t = 68 \text{ días}$$

Se utilizó una malla de:

$$\Delta x = 1 \text{ m}$$

Con un diferencial de tiempo:

$$\Delta t = 0.1 \text{ días}$$

Una vez definidos los parámetros y datos del problema se determinó la solución del ejemplo de Celia M. A., J. S. Kindred e I. Herrera, (1989) y también la determinación del número de Courant (Cr) y número Peclet (Pe) en las dos soluciones numéricas a continuación:

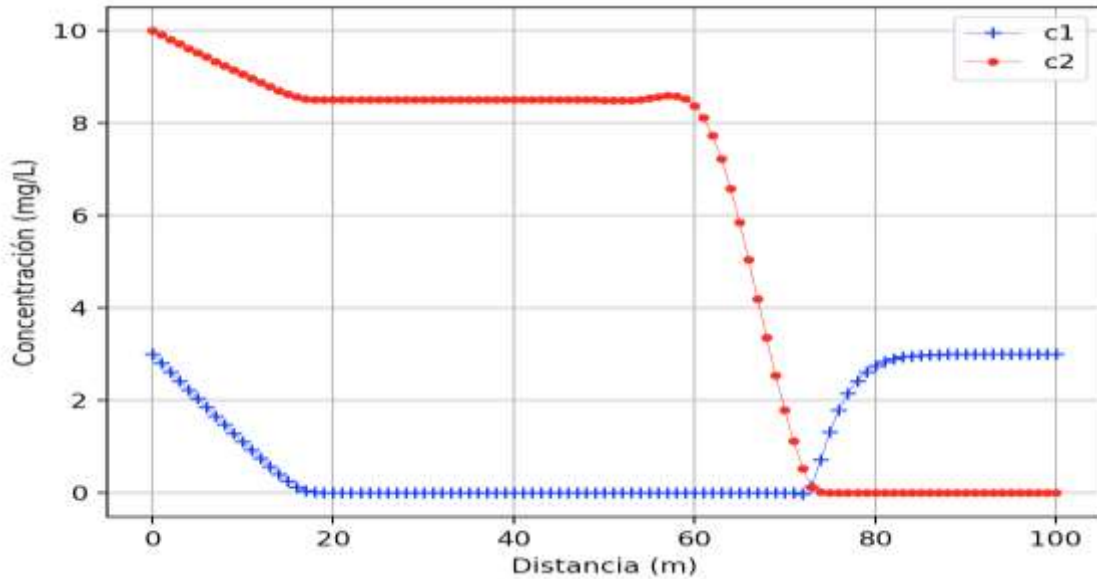


Figura 4-1. Solución numérica de las ecuaciones (3.1) y (3.2) utilizando Python 2019 mediante un esquema explícito para un tiempo de simulación de 68 días.

$$Cr = 0.1 \text{ y } Pe = 5$$

Mientras que los resultados presentados por Celia M. A., J. S. Kindred e I. Herrera, (1989) son los siguientes:

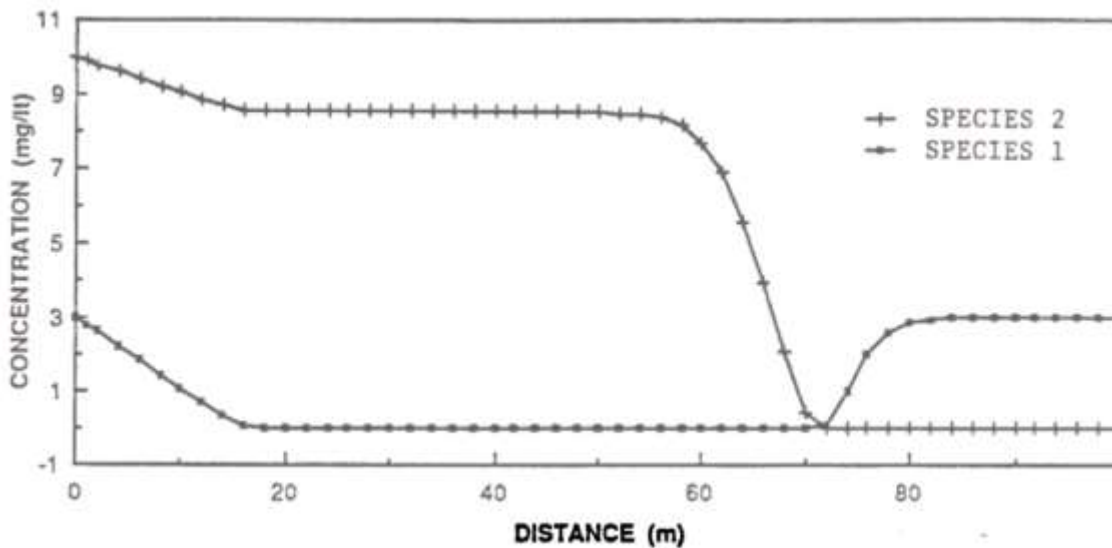


Figura 4-2. Solución numérica de las ecuaciones (3.1) y (3.2) utilizando una función de prueba óptima (OTF siglas en inglés) para un tiempo de simulación de 68 días, Celia M. A., J. S. Kindred e I. Herrera, (1989).

$$Cr = 0.1 \text{ y } Pe = 10$$

4.2 Ejemplo de aplicación 2D, transporte de un contaminante conservativo

Considérese el caso de transporte de un contaminante conservativo en el que la condición inicial está dada por una ley Gaussiana. El campo de velocidades del flujo es tipo rotacional y está descrito por $u_x(x, y) = -4y$, así como $u_y(x, y) = 4x$. El dominio espacial (x, y) es $\Omega = (-0.5, -0.5) \times (0.5, 0.5)$. El intervalo de tiempo es $[0, T] = [0, \pi/2]$, lo cual es el necesario para dar una rotación completa. La condición inicial está dada por:

$$c_0 = e^{\left[-\frac{(x-x_c)^2 + (y-y_c)^2}{2\sigma^2} \right]} \quad (4.5)$$

Las condiciones de frontera son:

$$\begin{aligned} \frac{\partial c}{\partial x}(x = -0.5, y, t) &= \frac{\partial c}{\partial x}(x = 0.5, y, t) = 0 \\ \frac{\partial c}{\partial y}(x, y = -0.5, t) &= \frac{\partial c}{\partial y}(x, y = 0.5, t) = 0 \\ t &\geq 0 \end{aligned}$$

Donde x_c , y_c y σ son los centros de la pluma de contaminante y la desviación estándar, respectivamente. La solución analítica para el ejemplo es el siguiente (Hong, Sharpley, & Shushuang, 1996):

$$c(x, y, t) = \frac{2\sigma^2}{2\sigma^2 + 4Dt} e^{\left[-\frac{(\bar{x}-x_c)^2 + (\bar{y}-y_c)^2}{2\sigma^2 + 4Dt} \right]} \quad (4.6)$$

Donde:

$$\begin{aligned} \bar{x} &= x \cos(4t) + y \sin(4t) \\ \bar{y} &= -x \sin(4t) + y \cos(4t) \end{aligned}$$

El modelo numérico tiene los siguientes datos:

$$\begin{aligned} D &= 0.0001 \\ x_c &= -0.25 \\ y_c &= 0 \\ \sigma &= 0.047 \end{aligned}$$

$$\Delta x = 0.01$$

$$\Delta y = 0.01$$

$$\Delta t = \frac{\pi}{1000}$$

Para conocer la estabilidad del modelo se utilizará el criterio del número de Courant y mediante la siguiente expresión: $Cr = \frac{[u]\Delta t}{\Delta x} \leq 1$. También se determinará el número de Peclet con la siguiente expresión: $Pe = \frac{\Delta x[u]}{D}$.

La condición inicial se centra en $(x, y) = (-0.25, 0)$, con un valor mínimo de 0 y un valor máximo de 1. En las figuras se muestran los contornos de la distribución de la pluma de contaminante para los tiempos indicados, a través de una vuelta completa. La Figura 4-1 corresponde a la solución analítica después de una vuelta completa, tiene un valor máximo de 0.8755 y un valor mínimo de 0. De igual forma, en la Figura 4-2 se muestran los resultados del modelo numérico donde se obtiene un valor máximo de 0.8263 y un valor mínimo de 0. Cabe destacar que los resultados reportados por (Hong, Sharpley, & Shushuang, 1996) con la aplicación del Método Euleriano – Lagrangiano Localizado Adjunto (ELLAM) tienen un valor máximo de 0.8599 y un valor mínimo de 0. Las figuras 4-4 y 4-5 corresponden a una solución con elemento finito tipo Galerkin (GAL) y Petrov – Galerkin (QPG), reportadas por (Hong, Sharpley, & Shushuang, 1996).

Tabla 4-1. Valores máximo y mínimo después de una vuelta $t = \left(\frac{5}{10}\right)\pi$, obtenida con diferentes métodos

Método	Valor máximo	Valor mínimo
Analítico	0.8755	0
Modelo numérico	0.8263	0
ELLAM	0.8599	0
GAL	0.7861	-0.1564
QPG	0.6197	-0.0978

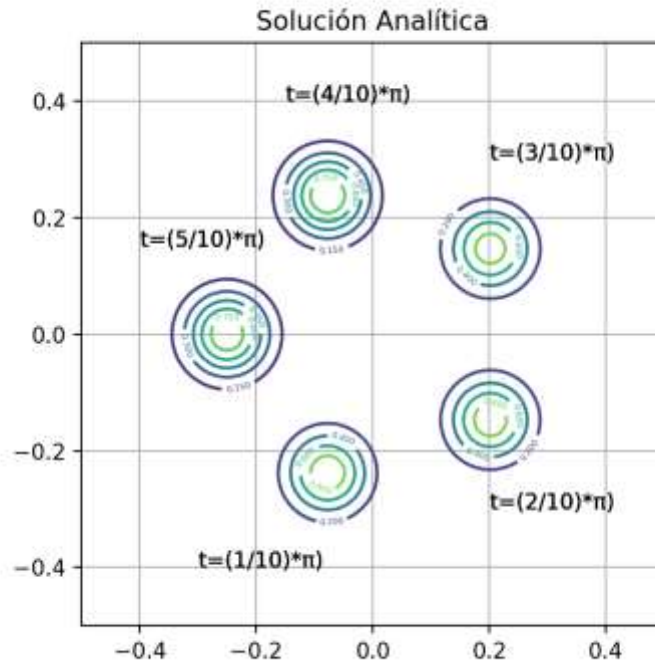


Figura 4-3 Solución analítica vista en planta.

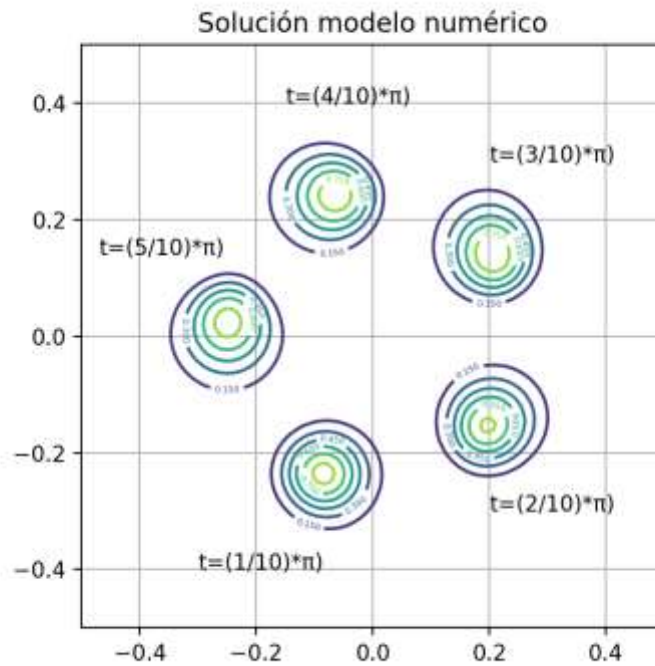


Figura 4-4 Solución del modelo numérico vista en planta.

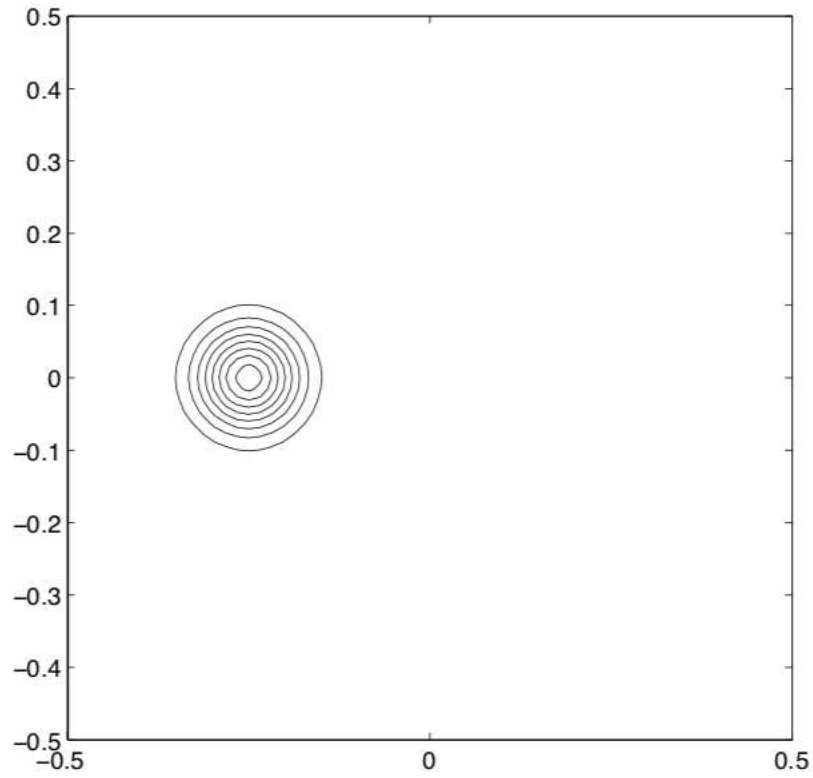


Figura 4-5 Solución ELLAM, vista en planta en el $t = \left(\frac{5}{10}\right)\pi$, (Hong, Sharpley, & Shushuang, 1996)

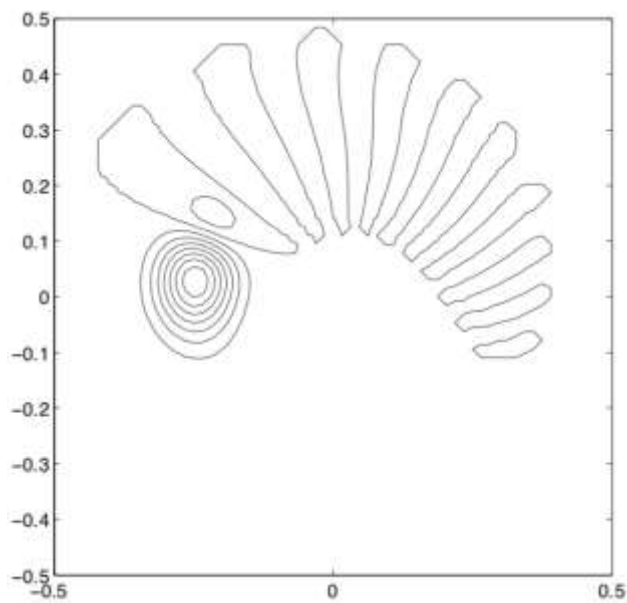


Figura 4-6 Solución en elemento finito para un $t = \left(\frac{5}{10}\right)\pi$ tipo GAL, (Hong, Sharpley, & Shushuang, 1996)

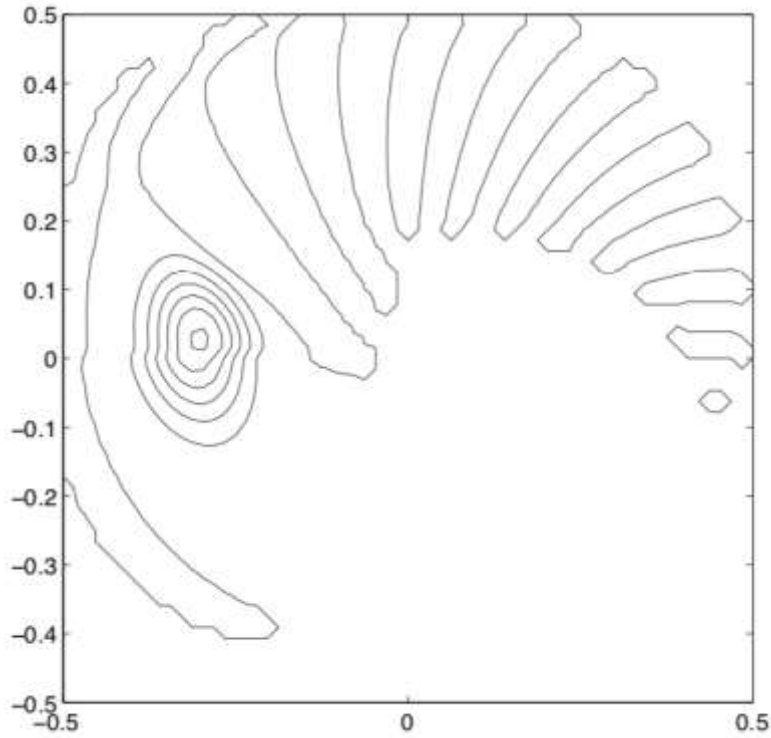


Figura 4-7 Solución en elemento finito para un $t = \left(\frac{5}{10}\right)\pi$ tipo QPG, (Hong, Sharpley, & Shushuang, 1996)

A continuación, se muestran figuras que comparan la solución analítica y la solución del modelo numérico, los cortes pasan por el valor máximo en cada tiempo de análisis.

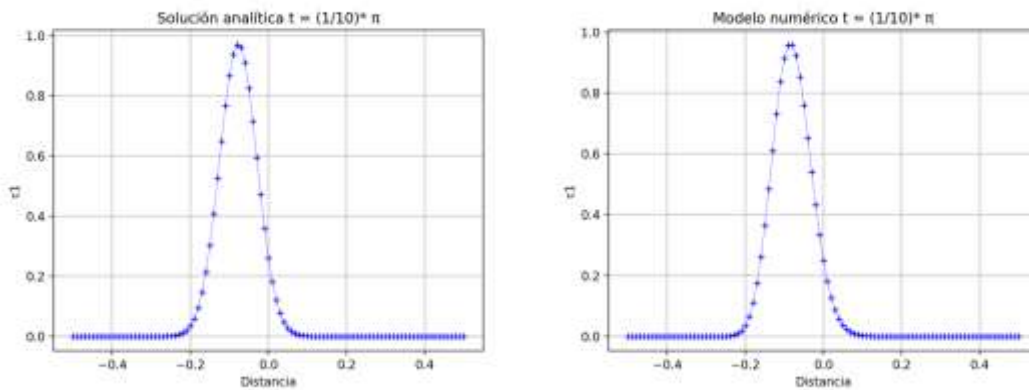


Figura 4-8 Comparación de solución analítica y modelo numérico en el tiempo $t = \left(\frac{1}{10}\right)\pi$

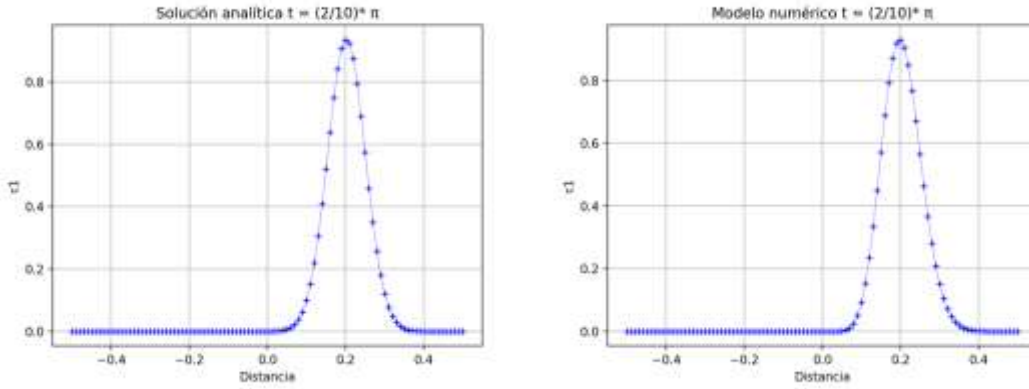


Figura 4-9 Comparación de solución analítica y modelo numérico en el tiempo $t = \left(\frac{2}{10}\right)\pi$

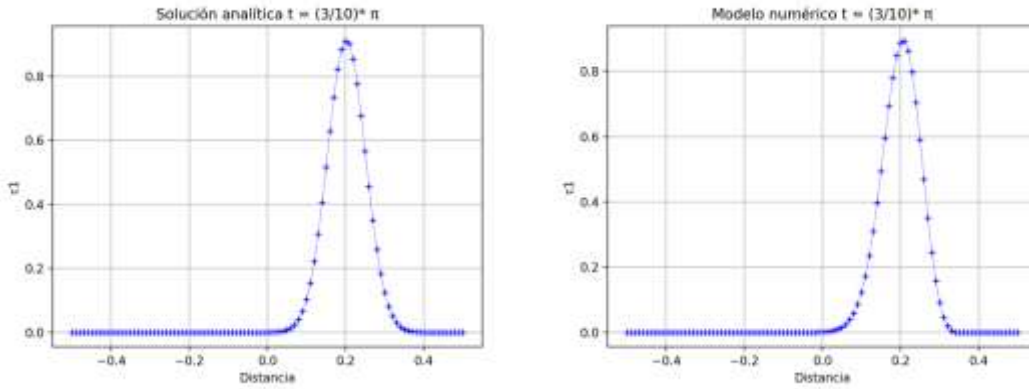


Figura 4-10 Comparación de solución analítica y modelo numérico en el tiempo $t = \left(\frac{3}{10}\right)\pi$

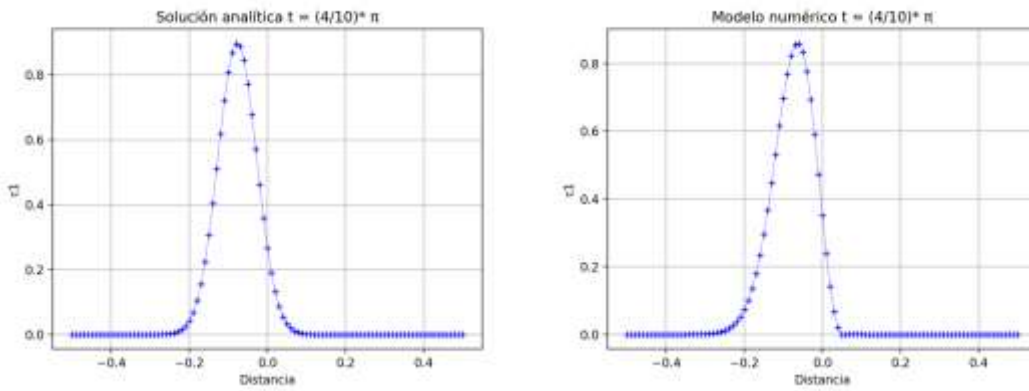


Figura 4-11 Comparación de solución analítica y modelo numérico en el tiempo $t = \left(\frac{4}{10}\right)\pi$

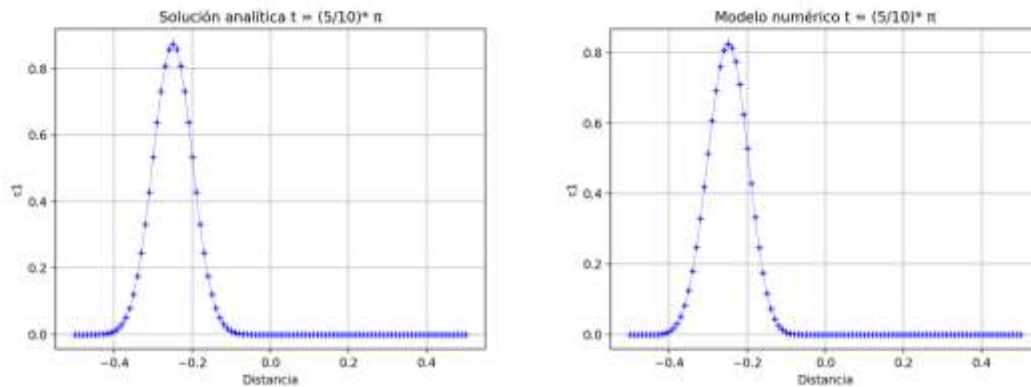


Figura 4-12 Comparación de solución analítica y modelo numérico en el tiempo
 $t = \left(\frac{5}{10}\right)\pi$

El valor máximo del número de Courant del modelo numérico es el siguiente:

$$Cr = \frac{(2)\left(\frac{\pi}{1000}\right)}{0.01} = \frac{1}{5}\pi = 0.6283$$

$$0.6283 < 1$$

Por lo tanto, la estabilidad del modelo numérico es aceptable.

Y el número de Peclet tiene valores entre 0 a 200.

La comparación que se logra observar en los cortes de la solución analítica y el modelo numérico se comporta de manera satisfactoria en todos los tiempos de análisis, por lo tanto, se tiene un buen comportamiento en el modelo numérico para la solución de este tipo de problemas.

4.3 Ejemplo de aplicación 2D, transporte de dos especies

El siguiente ejemplo implica el transporte de dos especies, acoplado a través de los términos de reacción. Las especies disueltas pueden ser un contaminante orgánico y oxígeno disuelto, lo que lleva a un problema de transporte de biodegradación

aerobia. Una analogía física para este sistema es el transporte de dos sustratos en presencia de una población biológica estacionaria.

Se considera el caso de transporte de un contaminante orgánico en el que la condición inicial está dada por una ley Gaussiana. El campo de velocidades del flujo está en m/día y es de tipo rotacional, está descrito por $u_x(x, y) = -0.04y + 2$, así como $u_y(x, y) = 0.04x - 2$. El dominio espacial (x, y) es $\Omega = (0, 100) \times (0, 100)$. El intervalo de tiempo es $[0, T] = [0, 150]$, lo cual es el necesario para dar una rotación completa.

Los datos para la simulación son los siguientes:

$$V_m^i = 1.0 \text{ días}^{-1}$$

$$i = 1, 2$$

$$K_h^i = 0.1 \text{ mg/l}$$

$$i = 1, 2$$

$$k_{12} = 2.0$$

$$k_{21} = 0.5$$

c_1 : Oxígeno disuelto

c_2 : Contaminante orgánico

Las condiciones iniciales se asignan como:

$$c_1(x, y, 0) = 3.0 \text{ mg/l}$$

$$c_2(x, y, 0) = 10e^{\left[-\frac{(x-x_c)^2 + (y-y_c)^2}{2\sigma^2}\right]} \text{ mg/l}$$

Donde x_c , y_c y σ son los centros de la pluma de contaminante y la desviación estándar, respectivamente, con los siguientes valores:

$$x_c = 25$$

$$y_c = 50$$

$$\sigma = 4.7$$

Las condiciones de frontera son:

$$\begin{aligned}\frac{\partial c}{\partial x}(x = 0, y, t) &= \frac{\partial c}{\partial x}(x = 100, y, t) = 0 \\ \frac{\partial c}{\partial y}(x, y = 0, t) &= \frac{\partial c}{\partial y}(x, y = 100, t) = 0 \\ t &\geq 0\end{aligned}$$

La especie estacionaria se fija en:

$$X_1 = 0.2 \text{ mg/l}$$

El coeficiente de difusión es:

$$D = 0.2 \text{ m}^2/\text{día}$$

Se utilizó una malla de:

$$\Delta x = \Delta y = 1 \text{ m}$$

Con un diferencial de tiempo:

$$\Delta t = 0.1 \text{ días}$$

Para conocer la estabilidad del modelo se utilizará el criterio del número de Courant mediante la siguiente expresión.

$$Cr = \frac{[u]\Delta t}{\Delta x} \leq 1$$

A continuación, se muestran las figuras con los resultados del modelo numérico, los cortes pasan por el valor máximo en cada tiempo de análisis. La condición inicial de c_2 se centra en $(x, y) = (25, 50)$, con un valor mínimo de 0 mg/l y un valor máximo de 10 mg/l.

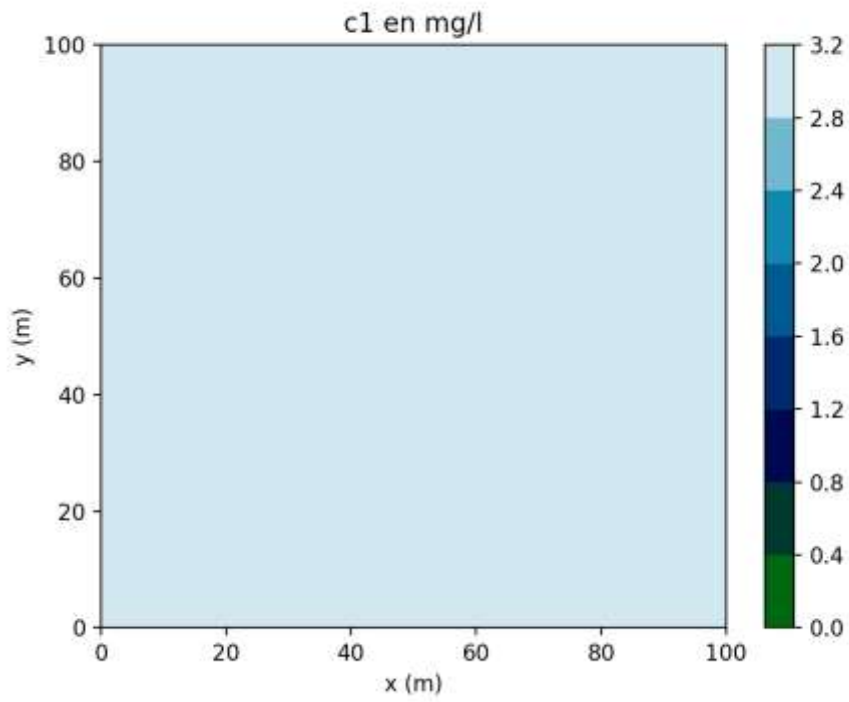


Figura 4-13. Vista 2D de c_1 para un $t = 0$ días

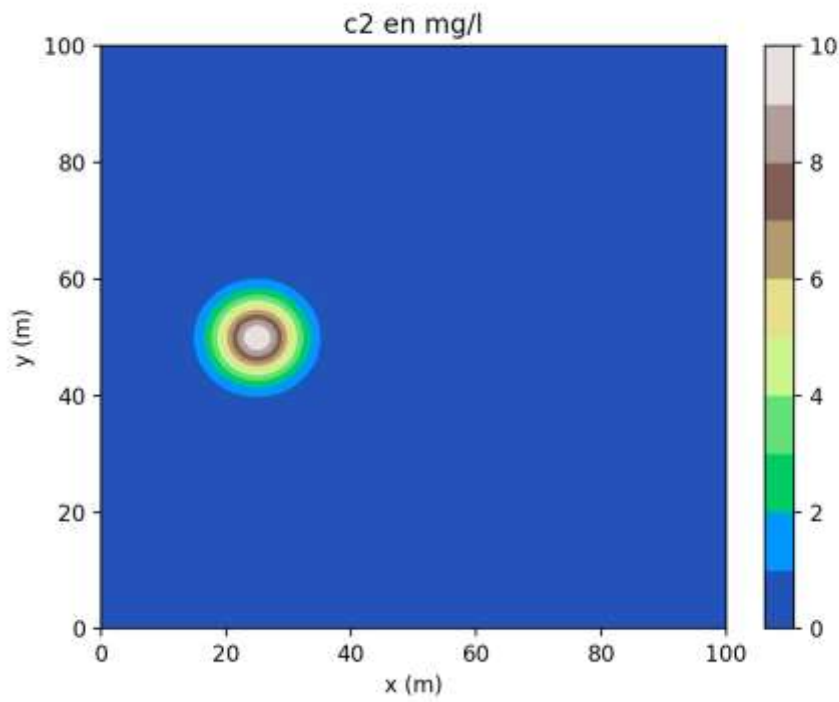


Figura 4-14. Vista 2D de c_2 para un $t = 0$ días

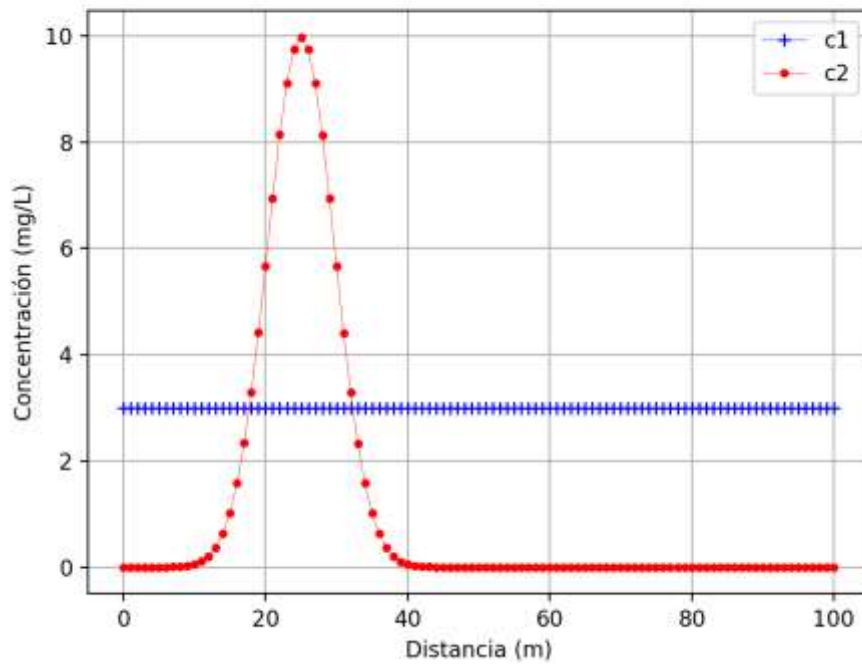


Figura 4-15. Corte transversal de c_1 y c_2 para un $t = 0$ días con $c_{2\text{máx}} = 10\text{mg/l}$

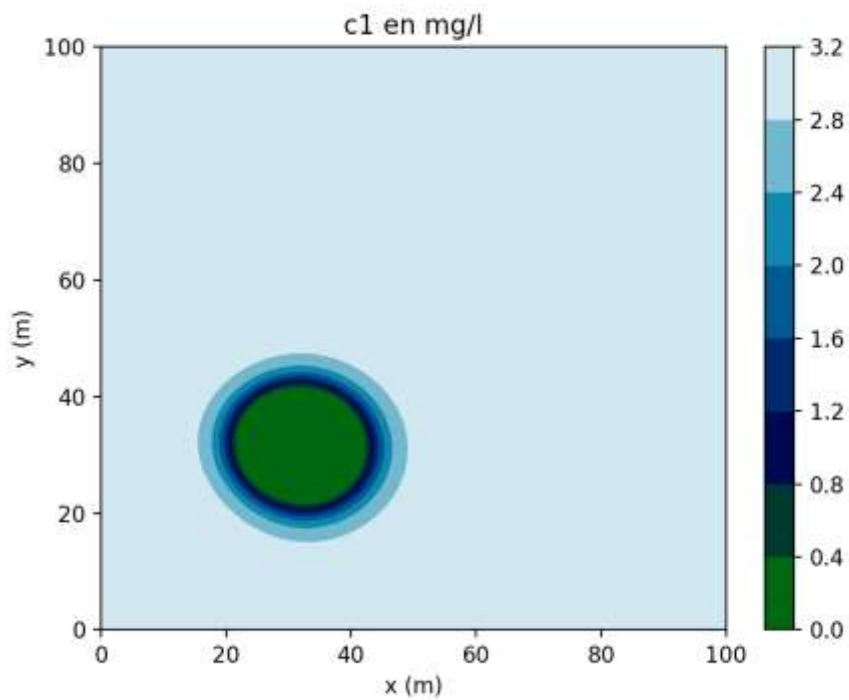


Figura 4-16. Vista 2D de c_1 para un $t = 20$ días

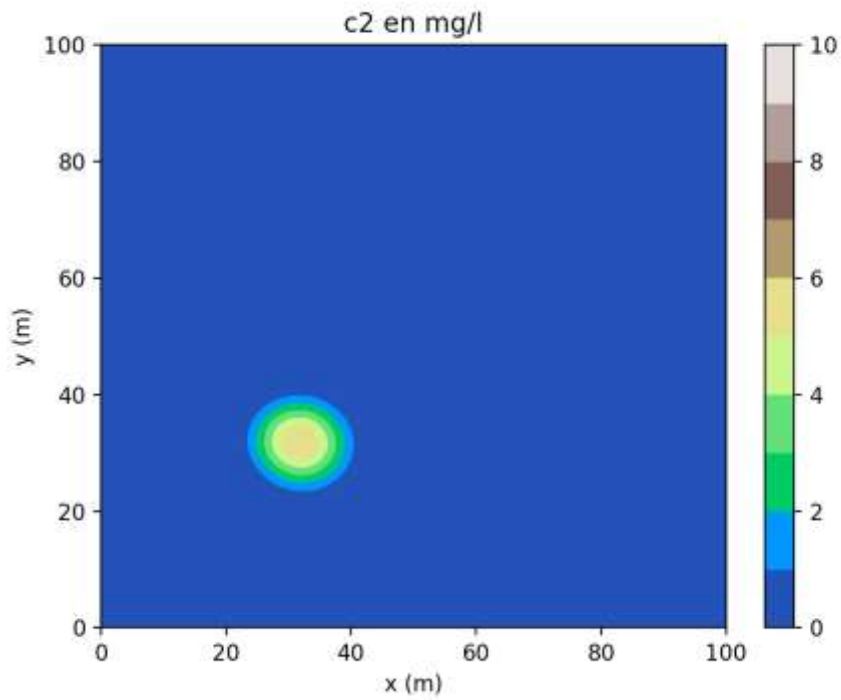


Figura 4-17. Vista 2D de c_2 para un $t = 20$ días

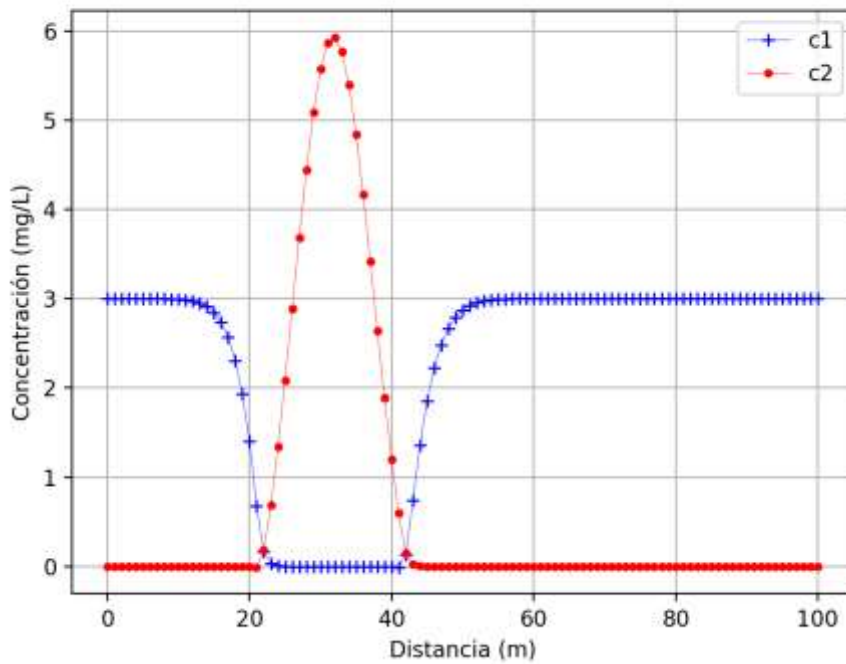


Figura 4-18. Corte transversal de c_1 y c_2 para un $t = 20$ días con $c_{2\text{máx}} = 5.93\text{mg/l}$

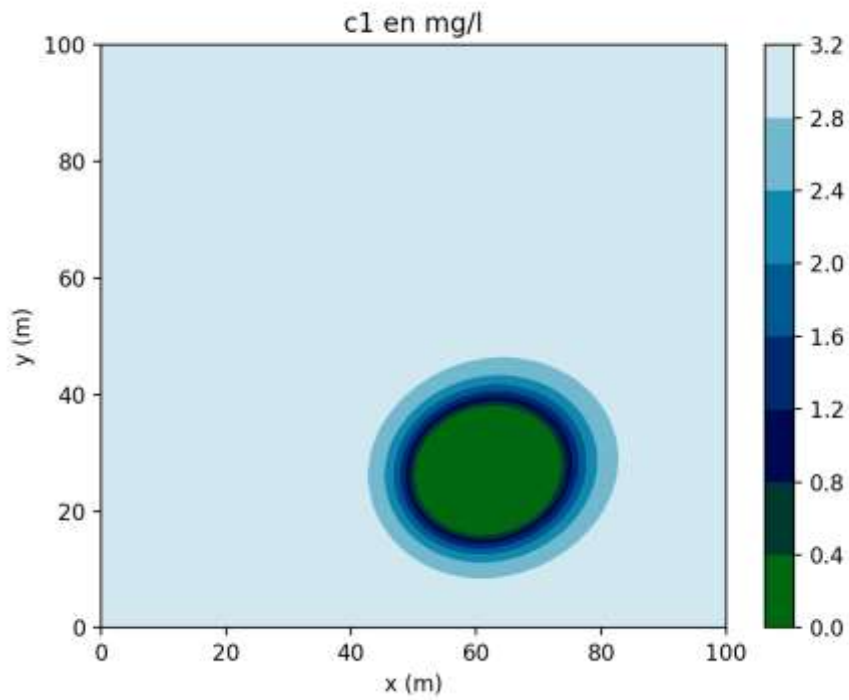


Figura 4-19. Vista 2D de c_1 para un $t = 50$ días

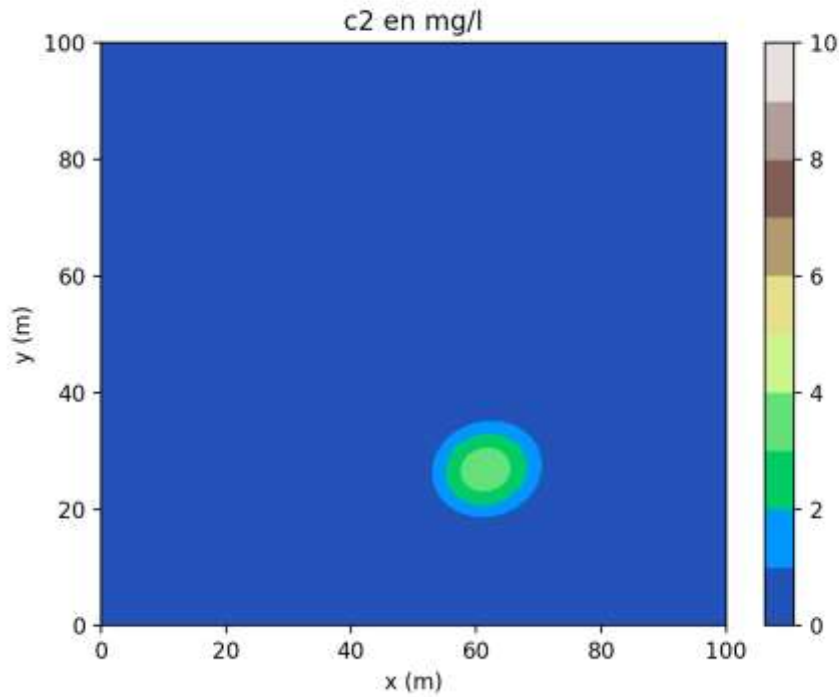


Figura 4-20. Vista 2D de c_2 para un $t = 50$ días

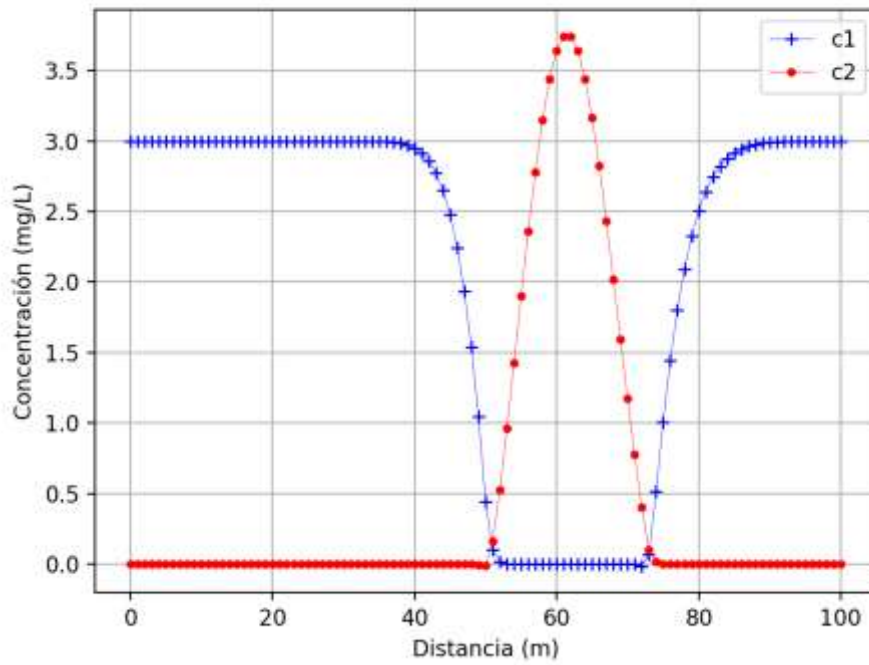


Figura 4-21. Corte transversal de c_1 y c_2 para un $t = 50$ días con $c_{2\text{máx}} = 3.74\text{mg/l}$

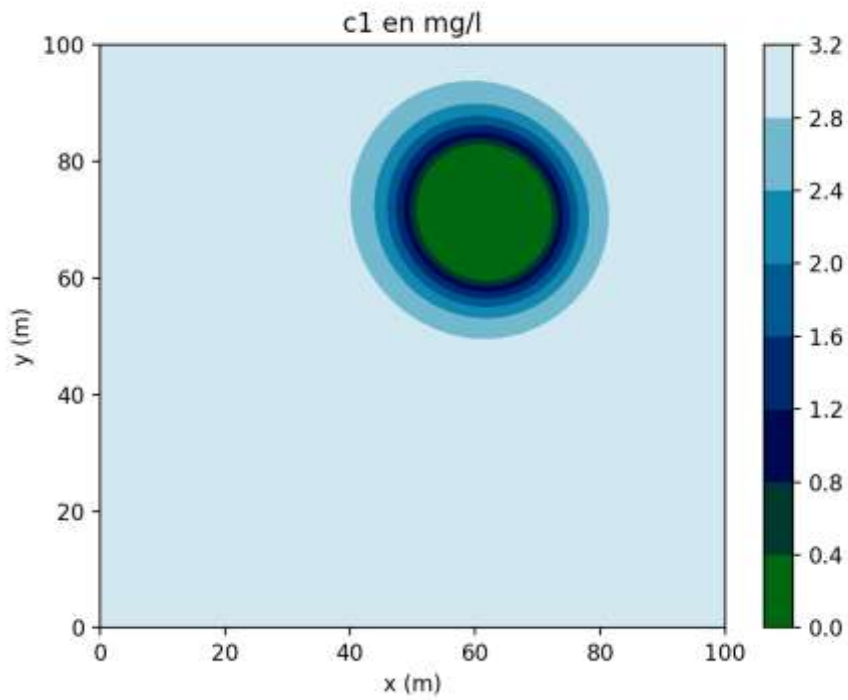


Figura 4-22. Vista 2D de c_1 para un $t = 100$ días

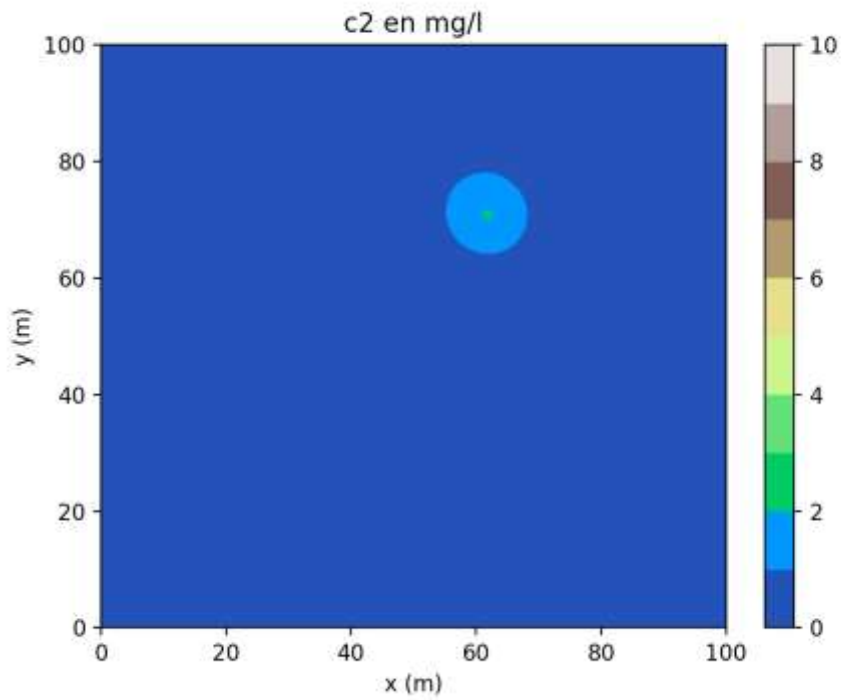


Figura 4-23. Vista 2D de c_2 para un $t = 100$ días

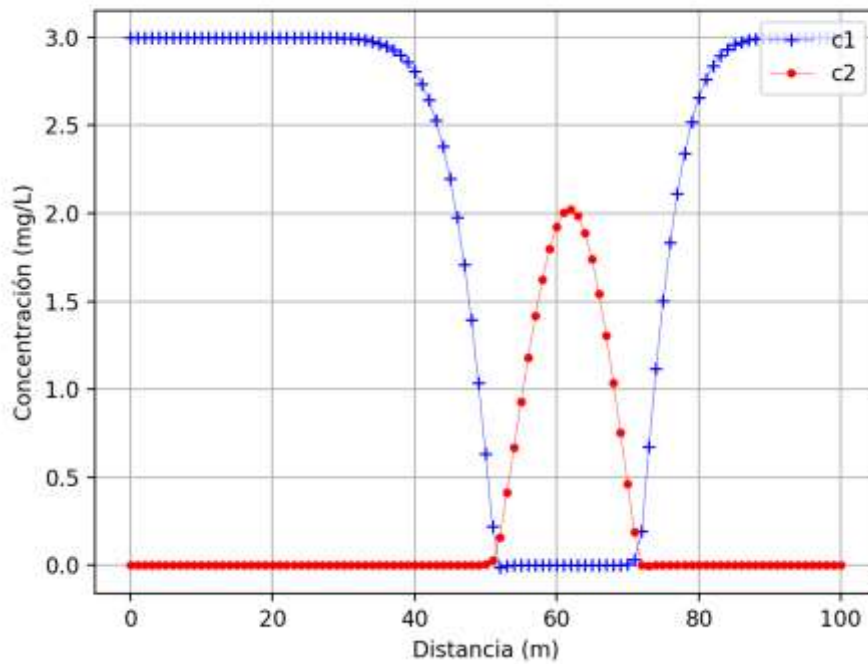


Figura 4-24. Corte transversal de c_1 y c_2 para un $t = 100$ días con $c_{2 \text{ máx}} = 2.02 \text{ mg/l}$

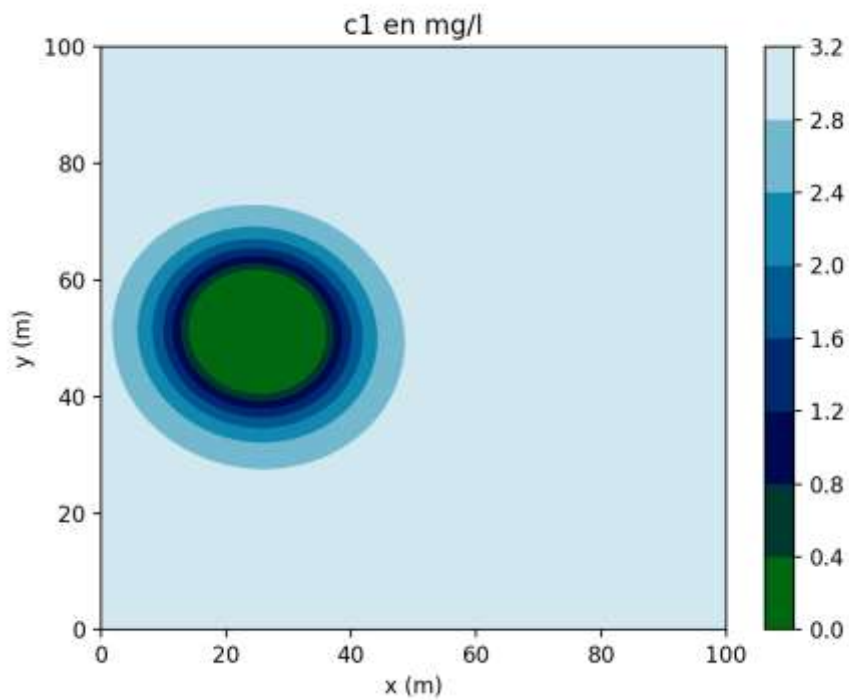


Figura 4-25. Vista 2D de c_1 para un $t = 150$ días

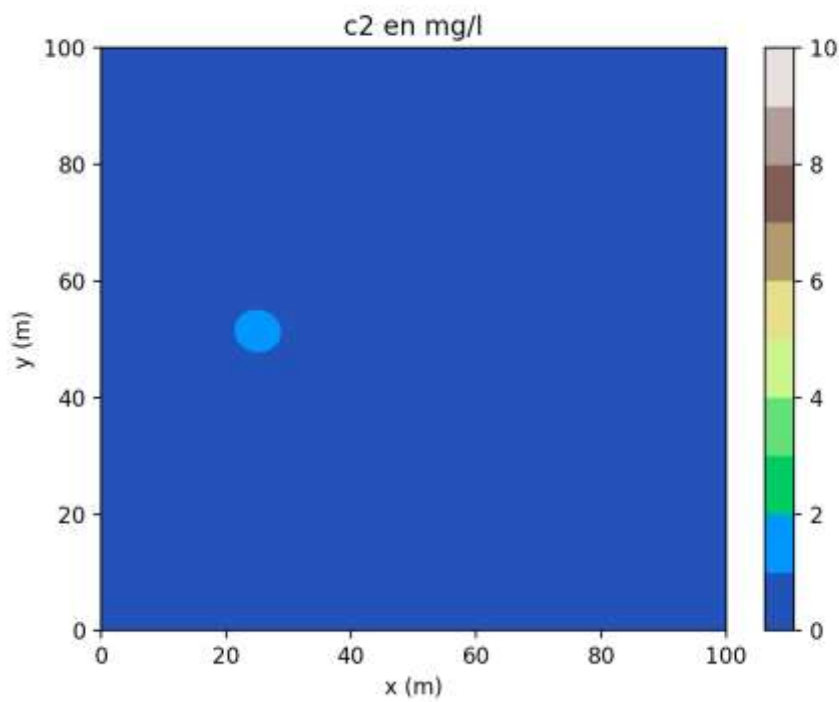


Figura 4-26. Vista 2D de c_2 para un $t = 150$ días

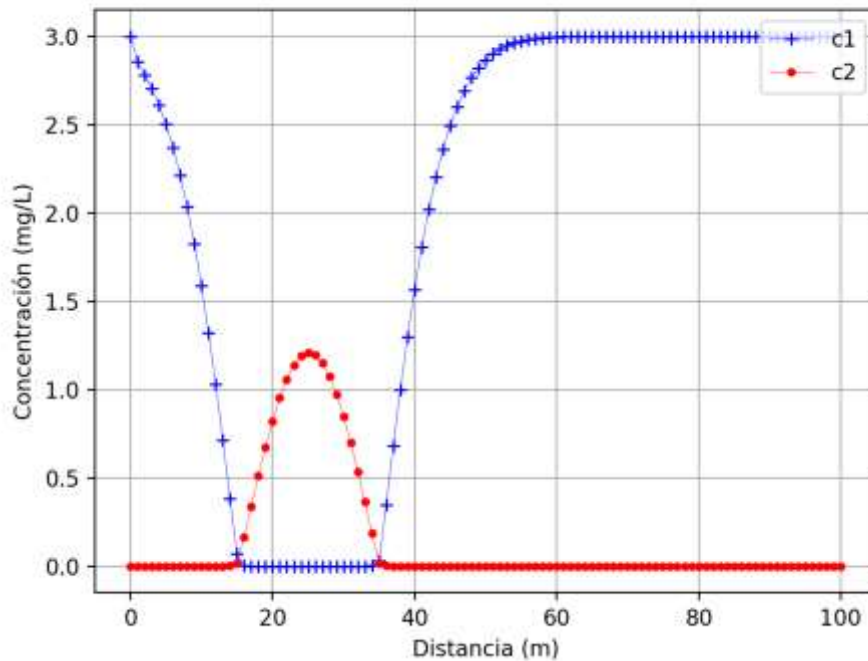


Figura 4-27. Corte transversal de c_1 y c_2 para un $t = 150$ días con $c_{2 \text{ máx}} = 1.20 \text{ mg/l}$

El valor máximo del número de Courant del modelo numérico es el siguiente:

$$Cr = \frac{(2)(0.1)}{1} = \frac{1}{5} = 0.2$$

$$0.2 < 1$$

Por lo tanto, la estabilidad del modelo numérico es aceptable.

Y el número de Peclet tiene valores entre 0 a 10:

De acuerdo a los resultados se observa que el modelo numérico se comporta de manera satisfactoria en todos los tiempos de análisis, por lo tanto, se tiene un buen comportamiento del modelo numérico para la solución de este tipo de problemas.

Tabla 4-2. Valores mínimos y máximos de c_1 con diferentes tiempos de análisis

t (días)	$c_{1 \text{ mín}} (mg/l)$	$c_{1 \text{ máx}} (mg/l)$
0	3	3
20	0	3
50	0	3
100	0	3
150	0	3

Tabla 4-3. Valores mínimos y máximos de c_2 con diferentes tiempos de análisis

t (días)	$c_{2 \text{ mín}} (mg/l)$	$c_{2 \text{ máx}} (mg/l)$
0	0	10.00
20	0	5.93
50	0	3.74
100	0	2.02
150	0	1.20

5 Aplicación del modelo numérico a un sistema de lagunas de estabilización aerobia en 3D

Durante años, los sistemas de aguas residuales de lagunas han brindado un desempeño de tratamiento a muchas comunidades pequeñas y medianas. Los atributos de estos procesos han sido atractivos como opciones rentables para el tratamiento de aguas residuales. El segmento de aireación en estos sistemas es el componente más crítico y es el núcleo de su proceso de tratamiento biológico (DEP, 2003).

Una laguna de estabilización es, básicamente una excavación en el suelo donde el agua residual se almacena para su tratamiento por medio de la actividad bacteriana (CONAGUA, 2007).

Cuando el agua residual es descargada en una laguna de estabilización se realiza en forma espontánea un proceso de auto purificación o estabilización natural, en el que tienen lugar fenómenos de tipo físico, químico y biológico. En esta simple descripción se establecen los aspectos fundamentales del proceso de tratamiento del agua que se lleva a cabo en las lagunas de estabilización (CONAGUA, 2007):

- Es un proceso natural de autodepuración.
- La estabilización de materia orgánica se realiza mediante la acción simbiótica de bacterias, algas y otros organismos superiores.
- Se presentan procesos físicos de remoción de materia suspendida.
- Se efectúan cambios químicos en la calidad del agua que, entre otros aspectos, mantienen las condiciones adecuadas para que los organismos puedan realizar la estabilización, transformación, y remoción de contaminantes orgánicos biodegradables y en algunos casos nutrientes.
- Se establecen cadenas tróficas y redes de competencia que permiten la eliminación de gran cantidad de microorganismos patógenos que se encuentran presentes en las aguas residuales. Por lo tanto, las lagunas de estabilización se consideran y se pueden proyectarse como un método de

tratamiento de la materia orgánica y de remoción de los patógenos presentes en el agua residual.

5.1 Ventajas y desventajas del sistema de lagunas de estabilización

El Departamento de Protección Medioambiental (DEP) del estado de Maine, USA 2003 menciona las siguientes:

Ventajas:

- Los sistemas de lagunas pueden ser rentables para diseñar y construir en áreas donde la tierra no es costosa.
- Utilizan menos energía que la mayoría de los métodos de tratamiento de aguas residuales.
- Son sencillos de operar y mantener, y generalmente requieren solo personal de medio tiempo.
- Son muy efectivos para eliminar organismos causantes de enfermedades (patógenos) de las aguas residuales.
- El efluente de los sistemas de lagunas puede ser adecuado para el riego (para cierto tipo de cultivos), debido a su alto contenido de nutrientes y bajo contenido de patógenos.

Desventajas:

- Los sistemas de lagunas requieren más tierra que otros métodos de tratamiento.
- Son menos eficientes en climas fríos y pueden requerir tierras adicionales o tiempos de detención más prolongados en estas áreas.
- El olor puede convertirse en una molestia durante la proliferación de algas o el deshielo primaveral en climas fríos.

- Las lagunas pueden proporcionar un área de reproducción para mosquitos y otros insectos.
- No son muy eficaces para eliminar metales pesados de las aguas residuales.
- El efluente de algunos tipos de lagunas contiene algas y, a menudo, requiere tratamiento adicional o “pulido” para cumplir con el estándar de descarga local.
- El sistema de lagunas de estabilización debe diseñarse individualmente para adaptarse a un sitio y uso específico como el tipo de aguas residuales a tratar y el nivel de tratamiento requerido por las regulaciones.

5.2 Tipos de lagunas de estabilización aerobia

5.2.1 De baja tasa

Los sistemas de lagunas de estabilización aerobia de baja tasa son grandes depósitos de poca profundidad donde los microorganismos se encuentran en suspensión y prevalecen condiciones aerobias. El oxígeno es suministrado de forma natural por la aireación de la superficie o por la fotosíntesis de las algas. La función es el tratamiento de desechos orgánicos solubles o el tratamiento de efluentes secundarios (CONAGUA, 2007).

En general, estas lagunas son más adecuadas para climas cálidos y soleados, donde es menos probable que se congelen. Las aguas residuales generalmente deben permanecer en lagunas aeróbicas de 3 a 50 días para recibir un tratamiento adecuado. Debido a que son tan poco profundos deben ser pavimentados o forrados con materiales que eviten que las malezas crezcan en la laguna (DEP, 2003).

5.2.2 De alta tasa

Una laguna de estabilización aerobia de alta tasa se caracteriza por utilizar aireadores para mezclar el contenido del estanque y agregar oxígeno a las aguas residuales. Estas lagunas se les puede conocer como lagunas de mezcla parcial o completa, según el grado de aireación. Con la excepción de los diseños impulsados por el viento, la mayoría de los aireadores requieren energía para funcionar. Sin embargo, los costos de energía son casi siempre considerablemente menores que los de otros sistemas de tratamiento mecánico. La aireación hace que el tratamiento sea más eficiente, lo que compensa los costos de energía en algunos casos. Las lagunas aireadas requieren menos superficie de terreno y un menor tiempo de tratamiento. Los tiempos de tratamiento para las aguas residuales en las lagunas se basan en factores como el diseño particular, la cantidad de aguas residuales a tratar y el nivel de tratamiento deseado (DEP, 2003).



Figura 5-1. Vista aérea del sistema de lagunas aireadas del Distrito Sanitario de Veazie, Maine, USA (DEP, 2003).

5.3 Descripción general del problema de estudio

En este apartado se presentará la utilidad del modelo numérico con procesos de biodegradación aerobia en el sistema de lagunas de estabilización aerobia de alta tasa, para la solución hidrodinámica se utilizará OpenFOAM 2019 y para la solución de las ecuaciones de difusión, advección y reacción se ocupará Python 2019.

Cabe destacar que se empleó SALOME 2019 en el pre - procesamiento (generación de mallas) y ParaView 2019 para la visualización de resultados.

Todas estas herramientas para la simulación numérica pretenden mostrar un camino para avanzar en el entendimiento de sistemas acuíferos. Los datos del problema se mostrarán a continuación.

Se decidió utilizar el sistema de lagunas de estabilización aerobia del pueblo de Dover – Foxcroft del estado de Maine, USA debido a la gran cantidad de información disponible (ubicación, descripción de la zona de estudio, operación, funcionamiento, caudal de entrada, caudal de salida, análisis de calidad de agua, entre otros).

El sistema de lagunas de estabilización aerobia recibe aguas residuales de aproximadamente 4,000 usuarios residenciales y comerciales. El sistema de recolección es de aproximadamente 32.2 kilómetros de longitud.

En el poblado se implementó un programa para separar la recolección de aguas residuales de las aguas pluviales, de tal manera que las aguas residuales llegan a las instalaciones del tratamiento de aguas residuales sin aguas pluviales.

El sistema de las lagunas se compone de tres lagunas operadas en serie con una superficie total de 36,830 metros cuadrados, una capacidad de 87.8 millones de litros con un tiempo de detención de 59 días. La aireación en las tres lagunas de tratamiento es proporcionada por aireación difusa y aireadores mecánicos.

Antes de que las aguas residuales entren al sistema de lagunas se realiza un tratamiento preliminar donde se remueven los sólidos de gran tamaño. El efluente se descarga al río Piscataquils a través de una tubería de 18 pulgadas de diámetro. El esquema general del diagrama de flujo se muestra en la Figura 5-3.

El sitio de estudio se localiza en las coordenadas latitud norte: 45°11'17.4" y longitud oeste: 69°10'58.6" (Figura 5-2).



Figura 5-2. Vista satelital de la zona de estudio

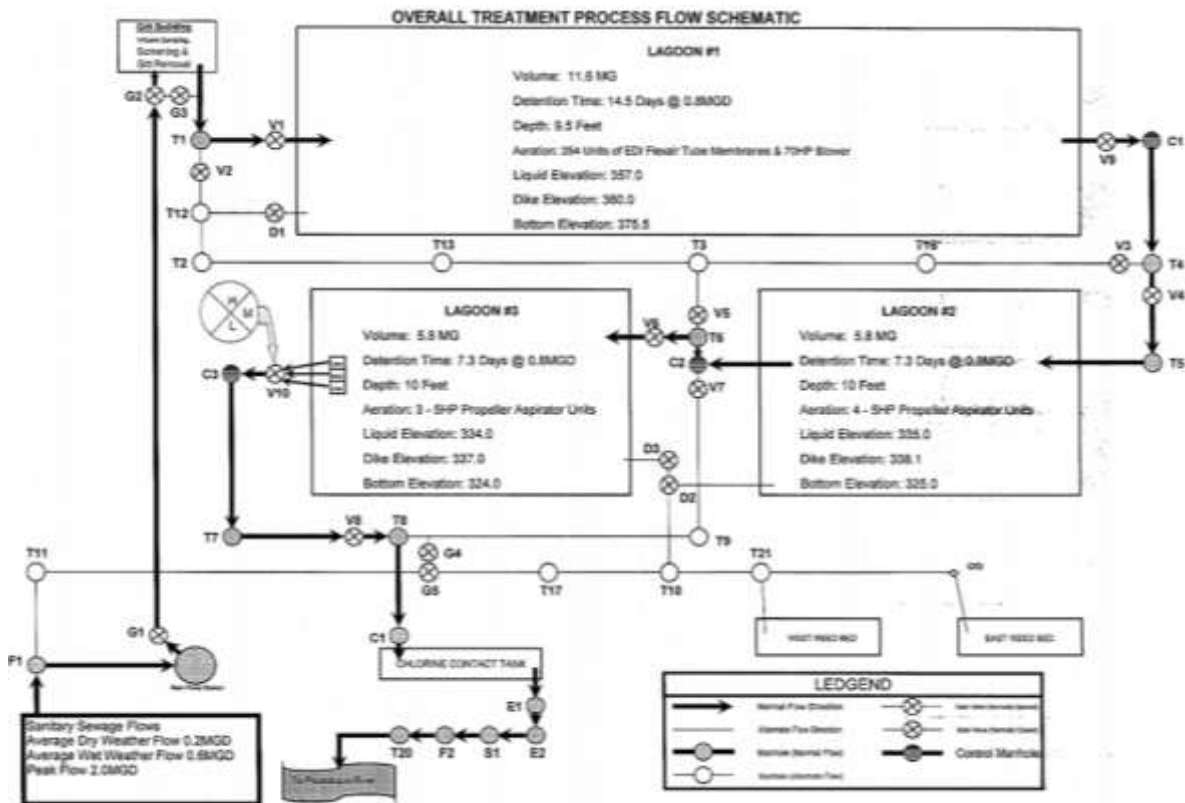


Figura 5-3 Diagrama de flujo del proceso de tratamiento general (DEP, 2003)

Tabla 5-1 Especificaciones generales del sistema de lagunas de estabilización

Laguna	1	2	3
Dimensiones de la laguna (ft)	900 x 240 x 9.5	480 x 230 x 10	480 x 230 x 10
Dimensiones de la laguna (m)	274.32x73.152 x 2.9	146.30 x 70.1 x 3.0	146.30 x 70.1 x 3.0
Superficie de la laguna (acres)	4.95	2.53	2.53
Superficie de la laguna (m ²)	20,067.05	10,255.63	10,255.63
Elevación al centro del conducto de entrada (m)	2.4	2.5	2.5
Elevación al centro del conducto de salida (m)	0.5	0.5	0.5

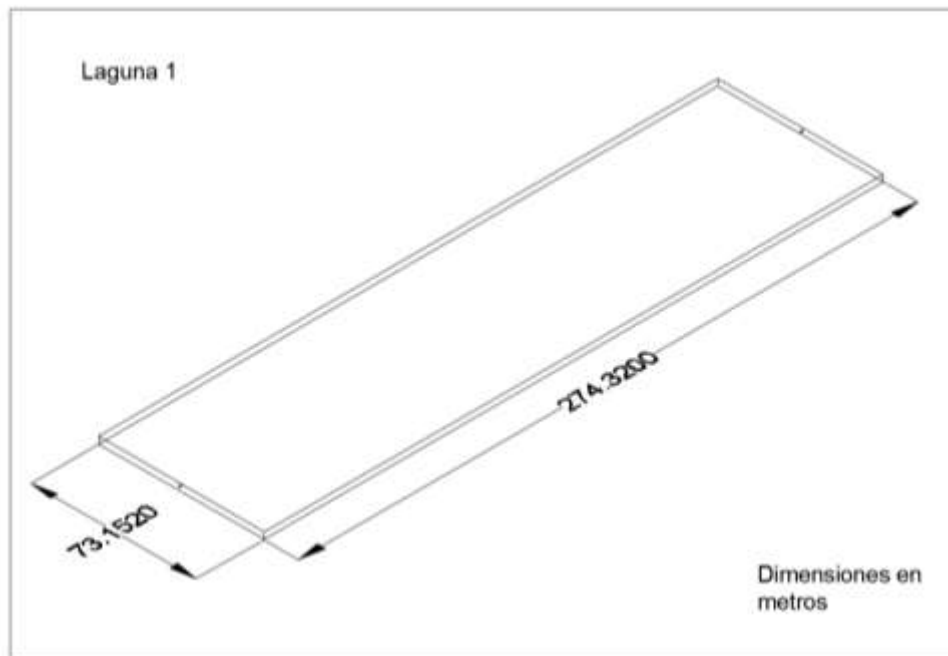


Figura 5-4 Dimensiones de la laguna 1 de estabilización aerobia

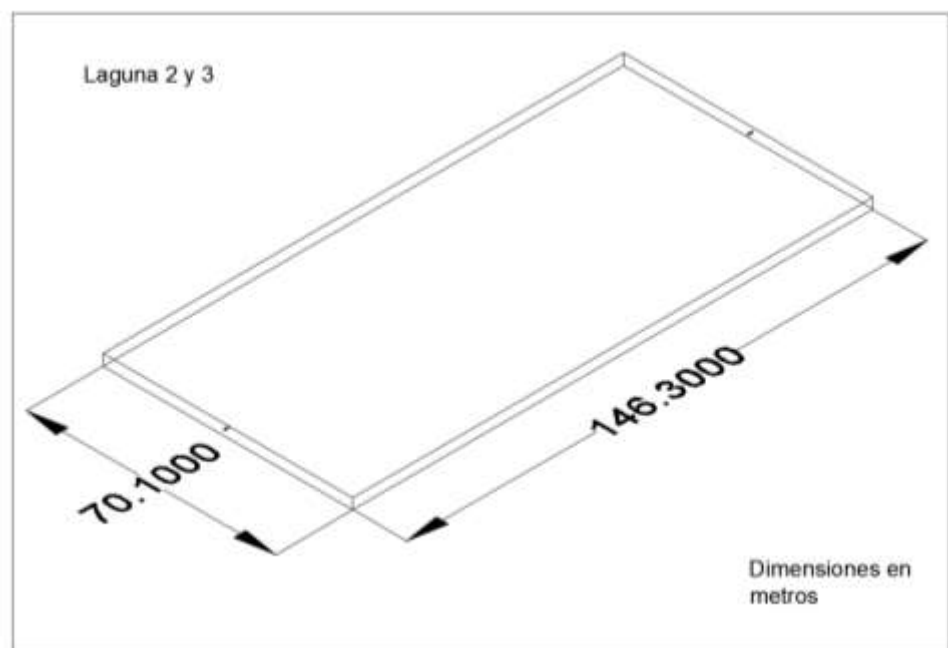


Figura 5-5 Dimensiones de la laguna 2 y 3 de estabilización aerobia

5.4 Determinación de datos hidrodinámicos

5.4.1 Pre - procesamiento de la simulación numérica

Para la generación de mallas en 3D se utilizó el software SALOME 2019 mediante el Módulo de Geometría (*Geometry Module*) y el Módulo de Malla (*Mesh Module*). Para crear la malla se utilizó el algoritmo NETGEN 1D – 2D – 3D, el cual produce una malla de forma predeterminada con relación a los bordes y la geometría de estudio, este algoritmo se eligió porque hace más eficiente la simulación (en bordes, la malla es más fina y en donde la geometría es regular, la malla es más grande).

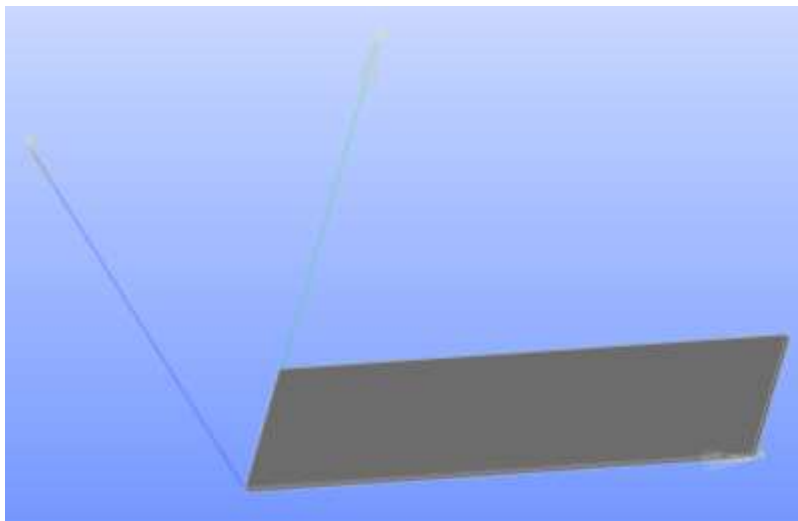


Figura 5-6 Geometría de la laguna 1 de estabilización aerobia

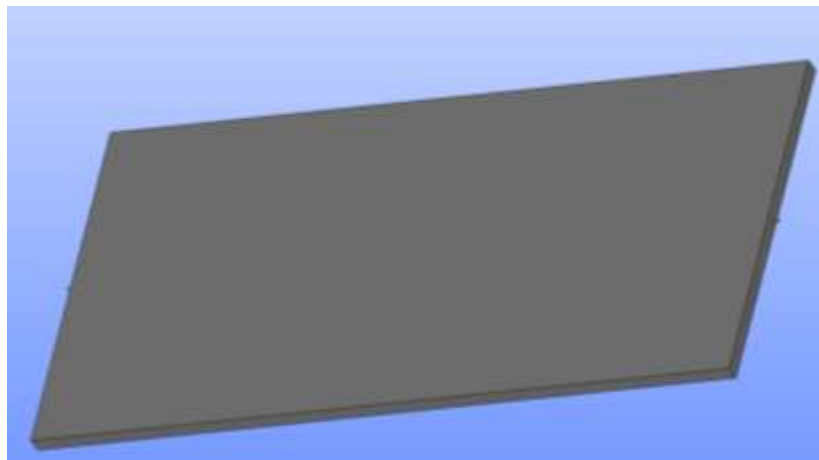


Figura 5-7 Geometría de la laguna 2 y 3 de estabilización aerobia

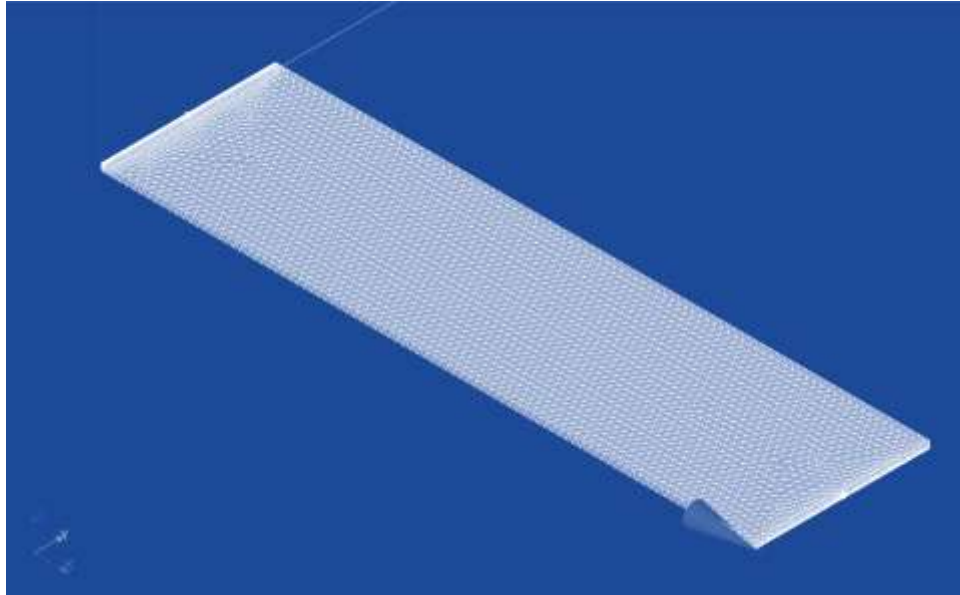


Figura 5-8 Malla de la laguna 1 de estabilización aerobia

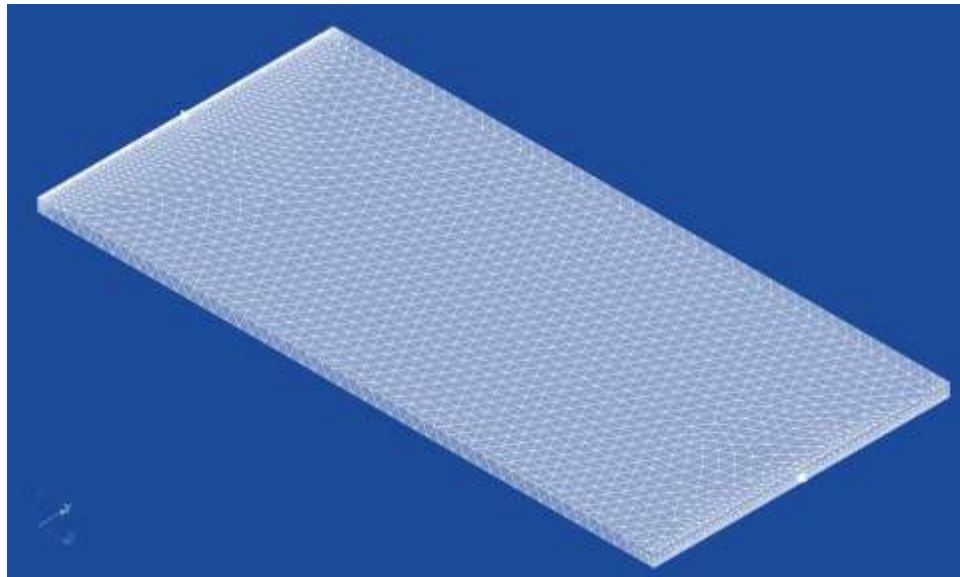


Figura 5-9 Malla de la laguna 2 y 3 de estabilización aerobia

En la generación de mallas se crearon grupos de las caras que servirán para la simulación hidrodinámica con OpenFOAM de la siguiente manera:

- *inlet* (entrada de caudal)
- *outlet* (salida de caudal)
- *wall* (paredes de la laguna y fondo de la laguna)
- *SLA* (superficie libre del agua)

5.4.2 Solución hidrodinámica con OpenFOAM

Las mallas creadas con SALOME se convierten al lenguaje de OpenFOAM utilizando el comando *ideasUnvToFoam* (Figura 5-10 y Figura 5-11).



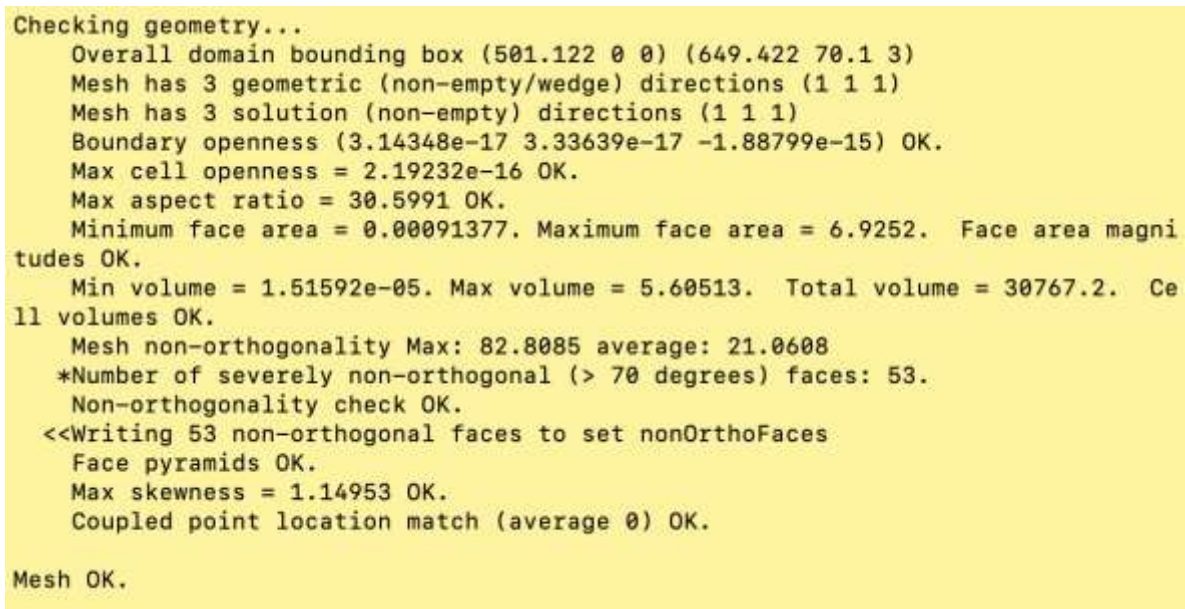
```
benignoduranaguilar — ofuser@6884e8605e18:~/workingDir/Documents/doc...
[ofuser@6884e8605e18 Laguna1]$ ls
0 Laguna1.unv constant system
[ofuser@6884e8605e18 Laguna1]$ ideasUnvToFoam Laguna1.unv
```

Figura 5-10. Conversión de mallas de SALOME a OpenFOAM para la laguna 1



```
benignoduranaguilar — ofuser@6884e8605e18:~/workingDir/Documents/doc...
[ofuser@6884e8605e18 Laguna2y3]$ ls
0 Laguna2y3.unv constant system
[ofuser@6884e8605e18 Laguna2y3]$ ideasUnvToFoam Laguna2y3.unv
```

Figura 5-11. Conversión de mallas de SALOME a OpenFOAM para la laguna 2 y 3
Una vez creada las mallas se utilizó el comando *checkMesh*, para analizar y evaluar estadísticas de malla y parámetros de calidad (Figura 5-11).



```
Checking geometry...
Overall domain bounding box (501.122 0 0) (649.422 70.1 3)
Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
Mesh has 3 solution (non-empty) directions (1 1 1)
Boundary openness (3.14348e-17 3.33639e-17 -1.88799e-15) OK.
Max cell openness = 2.19232e-16 OK.
Max aspect ratio = 30.5991 OK.
Minimum face area = 0.00091377. Maximum face area = 6.9252. Face area magnitudes OK.
Min volume = 1.51592e-05. Max volume = 5.60513. Total volume = 30767.2. Cell volumes OK.
Mesh non-orthogonality Max: 82.8085 average: 21.0608
*Number of severely non-orthogonal (> 70 degrees) faces: 53.
Non-orthogonality check OK.
<<Writing 53 non-orthogonal faces to set nonOrthoFaces
Face pyramids OK.
Max skewness = 1.14953 OK.
Coupled point location match (average 0) OK.

Mesh OK.
```

Figura 5-12. Comando *checkMesh* para verificar la validez de la malla

Se utilizó el solucionador *pisoFoam* que utiliza el algoritmo PISO (*Pressure Implicit with Splitting of Operators*), este es un solucionador para un flujo transitorio, incompresible y turbulento.

El algoritmo PISO fue propuesto por Issa en 1986 y es una extensión del algoritmo SIMPLE (*Semi-Implicit Method for Pressure Linked Equations*) utilizado en dinámica de fluidos computacional (CFD) para resolver las ecuaciones de Navier – Stokes. PISO implica un paso de predictor y dos pasos de corrector y está diseñado para satisfacer la conservación de masa utilizando pasos de predictor – corrector (OpenFOAM, 2018).

Se utilizará el modelo de turbulencia *Large Eddy simulation* (LES) que tiene la función de reducir el costo computacional al ignorar las escalas de longitud más pequeñas, que son las más costosas de resolver desde el punto de vista computacional, mediante el filtrado espacial bajo las ecuaciones de Navier – Stokes (OpenFOAM, 2018).

- Estructura de OpenFOAM

En resumen, la carpeta *0* representa las condiciones de contorno de los parámetros en todos los límites de la malla junto con los valores iniciales de los parámetros. La carpeta *constant* contiene una descripción completa de las propiedades físicas de transporte y de turbulencia. En la carpeta *system* se configuran los parámetros asociados con el procedimiento de solución, como el paso de tiempo, esquemas de discretización y el tipo de solucionador.

Tabla 5-2 Estructura de OpenFOAM para la simulación con el solver *pisoFOAM*

Carpeta	Archivos
<i> 0 </i>	<i> k, nut, nuTilda, p, s y U </i>
<i> constant </i>	<i> transportProperties y turbulenceProperties </i>
<i> system </i>	<i> blockMeshDict, controlDict, fvSchemes y fvSolution </i>

Los datos requeridos para la simulación hidrodinámica son los siguientes:

- Energía cinética turbulenta (k). Se utilizaron valores predeterminados de OpenFOAM.
- Viscosidad cinemática de Eddy (ν_t). Se utilizaron valores predeterminados de OpenFOAM.
- Viscosidad cinemática turbulenta (ν_{tilda}). Se utilizaron valores predeterminados de OpenFOAM.
- Presión cinemática (p).

Para problemas de densidad constante, el sistema de ecuaciones se puede simplificar. Los solucionadores incompresibles transforman la presión estática, p_s a la presión cinemática p_k .

$$p_k = \frac{p_s}{\rho} \quad [m^2/s^2] \quad (5.1)$$

Nota: Se tendrá en cuenta que la presión cinemática tiene el nombre de p en los archivos de OpenFOAM. En consecuencia, para interpretar los resultados de los solucionadores incompresibles, la presión cinemática debe multiplicarse por el valor de la densidad referente (constante).

- Parámetro de turbulencia (s). Se utilizaron valores predeterminados de OpenFOAM
- Velocidad de flujo (U)

El gasto en la laguna 1 permanece constante con un valor de 0.8 MGD (millones de galones estadounidenses por día), lo cual equivale a $0.035 \text{ m}^3/\text{s}$; la tubería en la entrada y en la salida de la laguna tiene un diámetro de 18 pulgadas, equivale a

0.4572 m. Por lo tanto, la velocidad en la entrada a la laguna 1 se calcula con la ecuación de continuidad:

$$A = \frac{\pi D^2}{4} \tag{5.2}$$

$$A = \frac{(3.1416)(0.4572)^2}{4} = 0.1642 \text{ m}^2 \tag{5.3}$$

$$U = \frac{Q}{A} \tag{5.4}$$

$$U = \frac{0.0350}{0.1642} = 0.2132 \text{ m/s} \tag{5.5}$$

Para mejorar el proceso de simulación hidrodinámica, se asignó un valor de velocidad constante en el eje x en el dominio de aplicación, esto fue para reducir el tiempo de simulación debido a que se necesita conocer únicamente el campo de velocidades en condición de estabilidad en las lagunas de estudio, el análisis se realizó con la ecuación de continuidad de la siguiente manera:

- Laguna 1

$$U = \frac{0.0350}{(73.152)(2.9)} = \frac{0.0350}{212.1408} = 0.000164985 \text{ m/s}$$

- Laguna 2 y 3

$$U = \frac{0.0350}{(70.1)(3.0)} = \frac{0.0350}{210.3} = 0.000166429 \text{ m/s}$$

A continuación, se muestran las condiciones iniciales y de frontera para las simulaciones hidrodinámicas de las lagunas 1, 2 y 3.

- Condiciones de frontera para la Laguna 1.

Tabla 5-3 Condiciones de frontera para la Laguna 1.

Concepto	Superficie de contorno	Tipo	Valor
<i>k</i>	<i>inlet</i>	<i>fixedValue</i>	2×10^5
	<i>outlet</i>	<i>inletOutlet</i>	0
	<i>wall</i>	<i>fixedValue</i>	0
	<i>SLA</i>	<i>inletOutlet</i>	0
<i>nut</i>	<i>inlet</i>	<i>zeroGradient</i>	0
	<i>outlet</i>	<i>zeroGradient</i>	-
	<i>wall</i>	<i>zeroGradient</i>	-
	<i>SLA</i>	<i>zeroGradient</i>	-
<i>nuTilda</i>	<i>inlet</i>	<i>fixedValue</i>	0
	<i>outlet</i>	<i>inletOutlet</i>	0
	<i>wall</i>	<i>fixedValue</i>	0
	<i>SLA</i>	<i>inletOutlet</i>	0
<i>p</i>	<i>inlet</i>	<i>zeroGradient</i>	-
	<i>outlet</i>	<i>fixedValue</i>	0
	<i>wall</i>	<i>zeroGradient</i>	-
	<i>SLA</i>	<i>totalPressure</i>	0
<i>s</i>	<i>inlet</i>	<i>fixedValue</i>	1
	<i>outlet</i>	<i>inletOutlet</i>	0
	<i>wall</i>	<i>zeroGradient</i>	-
	<i>SLA</i>	<i>zeroGradient</i>	-
<i>U</i>	<i>internalField (0.000164985 0 0)</i>		
	<i>inlet</i>	<i>turbulentInlet</i>	(0.2132 0 0)
	<i>outlet</i>	<i>turbulentInlet</i>	(0.2132 0 0)
	<i>wall</i>	<i>noSlip</i>	-
	<i>SLA</i>	<i>pressureInletOutletVelocity</i>	0

- Condiciones de frontera para la Laguna 2 y 3.

Tabla 5-4 Condiciones de frontera para la Laguna 2 y 3.

Concepto	Superficie de contorno	Tipo	Valor
<i>k</i>	<i>inlet</i>	<i>fixedValue</i>	2×10^5
	<i>outlet</i>	<i>inletOutlet</i>	0
	<i>wall</i>	<i>fixedValue</i>	0
	<i>SLA</i>	<i>inletOutlet</i>	0
<i>nut</i>	<i>inlet</i>	<i>zeroGradient</i>	0
	<i>outlet</i>	<i>zeroGradient</i>	-
	<i>wall</i>	<i>zeroGradient</i>	-
	<i>SLA</i>	<i>zeroGradient</i>	-
<i>nuTilda</i>	<i>inlet</i>	<i>fixedValue</i>	0
	<i>outlet</i>	<i>inletOutlet</i>	0
	<i>wall</i>	<i>fixedValue</i>	0
	<i>SLA</i>	<i>inletOutlet</i>	0
<i>p</i>	<i>inlet</i>	<i>zeroGradient</i>	-
	<i>outlet</i>	<i>fixedValue</i>	0
	<i>wall</i>	<i>zeroGradient</i>	-
	<i>SLA</i>	<i>totalPressure</i>	0
<i>s</i>	<i>inlet</i>	<i>fixedValue</i>	1
	<i>outlet</i>	<i>inletOutlet</i>	0
	<i>wall</i>	<i>zeroGradient</i>	-
	<i>SLA</i>	<i>zeroGradient</i>	-
<i>U</i>	<i>internalField (0.000166429 0 0)</i>		
	<i>inlet</i>	<i>turbulentInlet</i>	(0.2132 0 0)
	<i>outlet</i>	<i>turbulentInlet</i>	(0.2132 0 0)
	<i>wall</i>	<i>noSlip</i>	-
	<i>SLA</i>	<i>pressureInletOutletVelocity</i>	0

En la siguiente tabla se muestra la descripción de los tipos de frontera que se utilizaron para la simulación hidrodinámica en OpenFOAM.

Tabla 5-5 Descripción de los tipos de frontera utilizados en OpenFOAM.

Tipo	Descripción
<i>fixedValue</i>	Aplica un valor fijo a la frontera y es del tipo Dirichlet.
<i>zeroGradient</i>	Aplica un gradiente cero a la frontera y es del tipo Neumann.
<i>inletOutlet</i>	Cuando la dirección del flujo es hacia el exterior del dominio, funciona como una condición de frontera de gradiente cero y cuando el flujo se dirige hacia el interior del dominio funciona como una condición de frontera de valor fijo.
<i>totalPressure</i>	Funciona como una condición fija para establecer la presión estática.
<i>internalField</i>	Define la condición inicial en el campo de dominio.
<i>turbulentInlet</i>	Esta condición de frontera genera una condición de entrada fluctuante al agregar un componente aleatorio a un campo de referencia.
<i>noSlip</i>	Esta condición de frontera fija a la velocidad en cero en las paredes.
<i>pressureInletOutletVelocity</i>	Esta condición de frontera de entrada/salida de velocidad se aplica a los límites de presión. Se aplica una condición de gradiente cero para el flujo de salida (como lo define el flujo); para el flujo de entrada, la velocidad se obtiene del componente normal de la cara del campo del valor de la celda interna.

Nota: Se utilizaron los mismos parámetros de simulación para la laguna 2 y 3 debido a que sus características geométricas y su funcionamiento es el mismo.

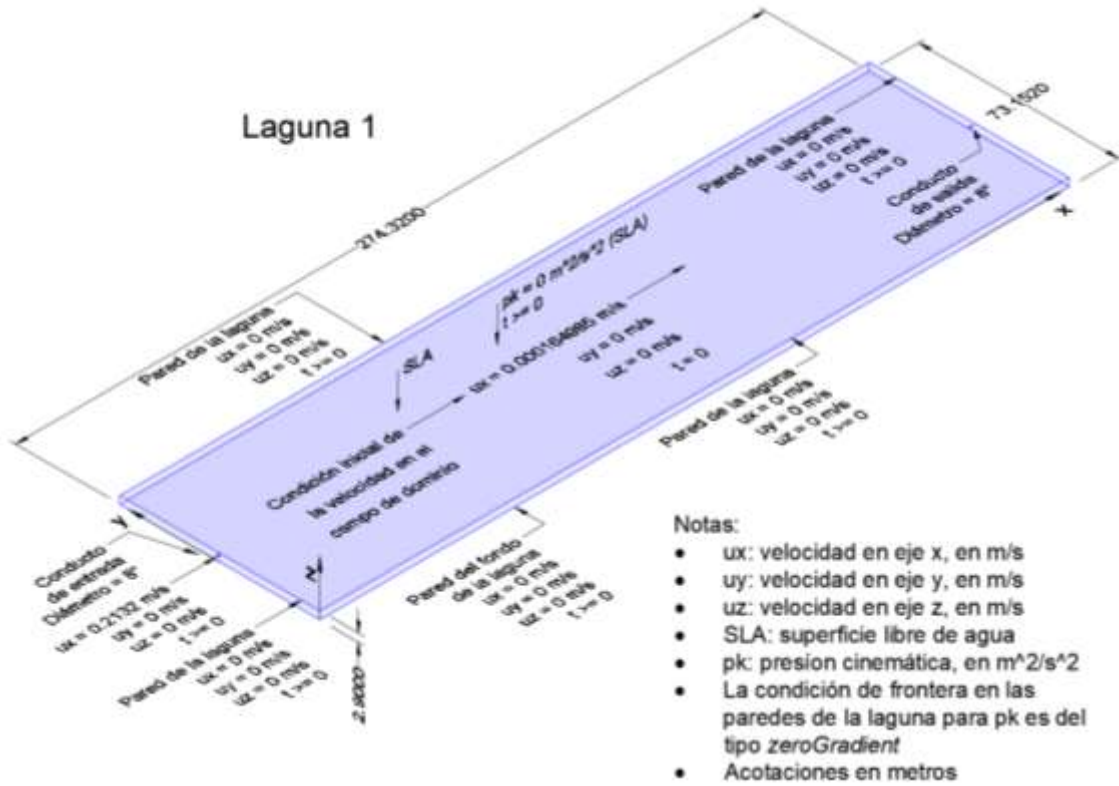


Figura 5-13. Condiciones de frontera de la laguna 1 para OpenFOAM.

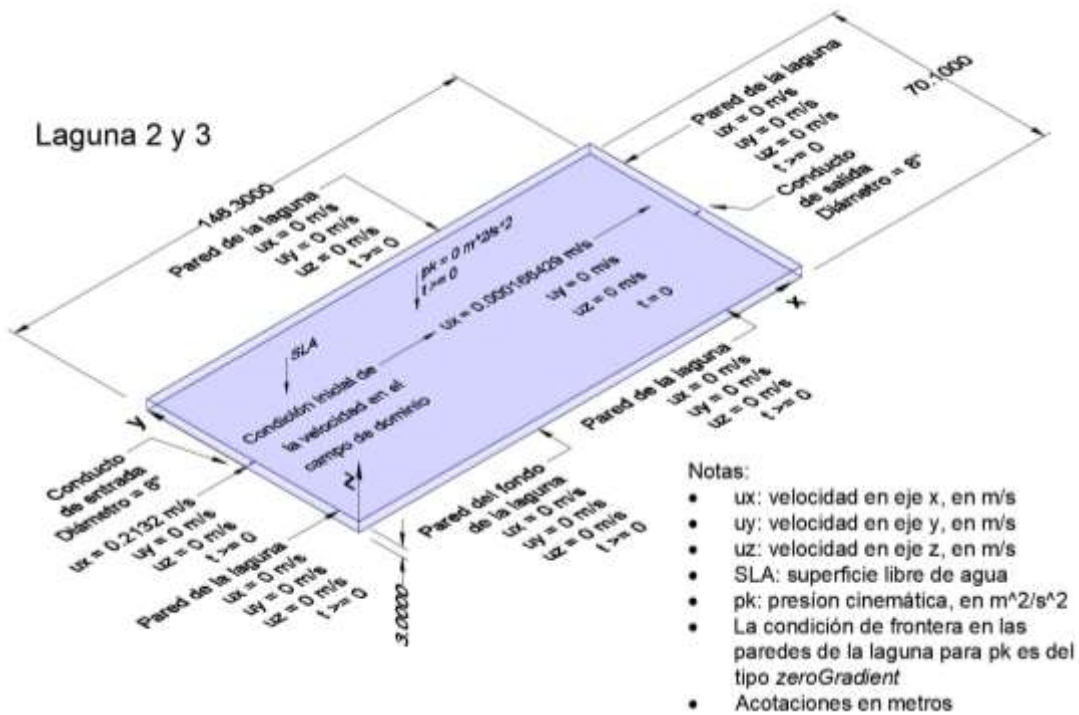


Figura 5-14. Condiciones de frontera de la laguna 2 y 3 para OpenFOAM

En el Anexo 8.3 se muestra a detalle las condiciones iniciales, condiciones de frontera, propiedades físicas, propiedades de turbulencia y parámetros de solución para la realización de la simulación hidrodinámica con OpenFOAM.

A continuación, se presenta la visualización de resultados en ParaView.

- Visualización del campo de velocidades en la laguna 1

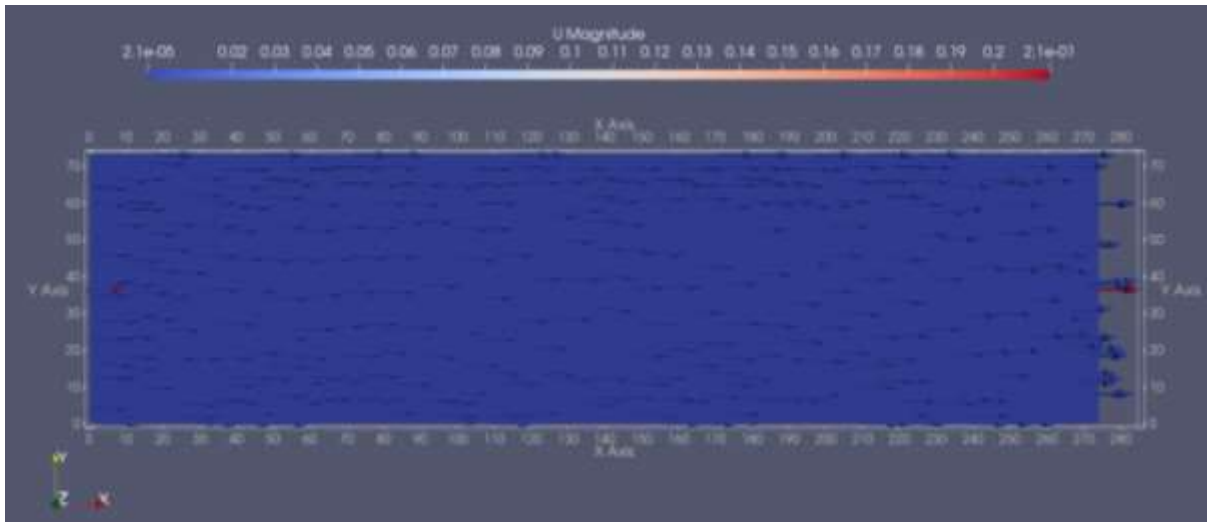


Figura 5-15. Vista en planta del campo de velocidades en la laguna 1, con $z = 2.9$ m y $t = 7,200$ s.

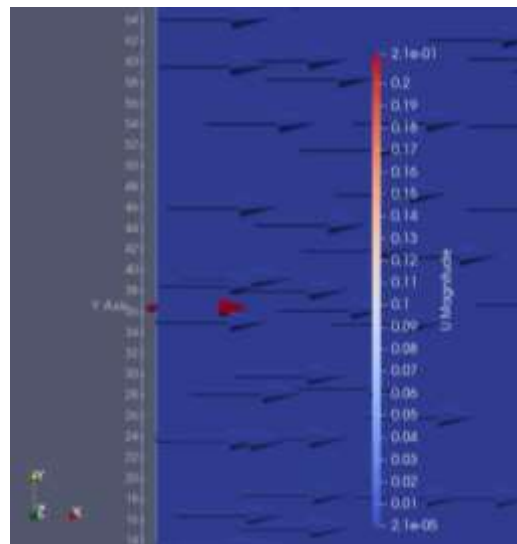


Figura 5-16. Vista ampliada en planta del campo de velocidades en la entrada de la laguna 1, con $z = 2.9$ m y $t = 7,200$ s.

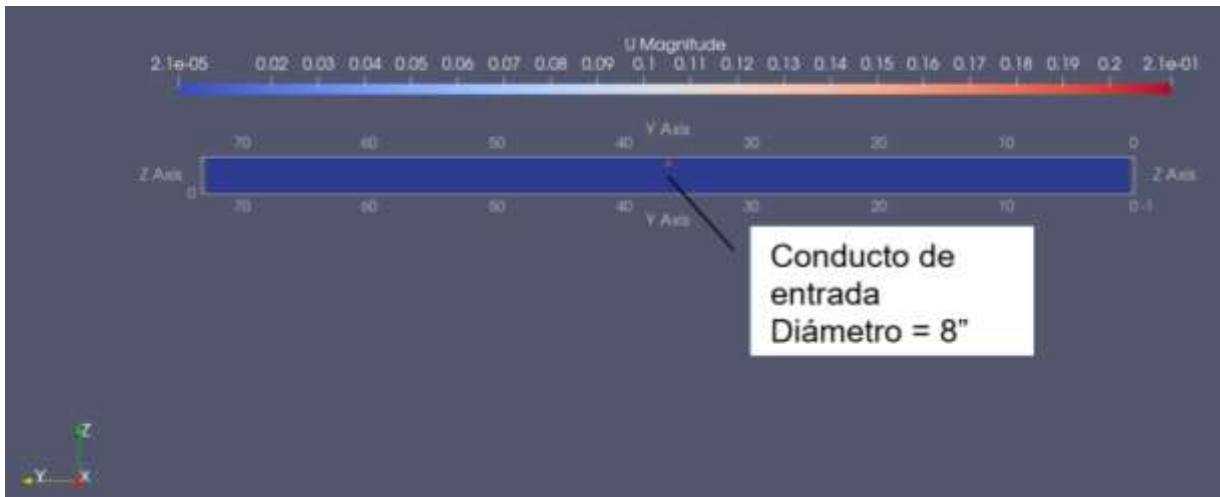


Figura 5-17. Vista en la sección de entrada del flujo en la laguna 1, $t = 7,200$ s.

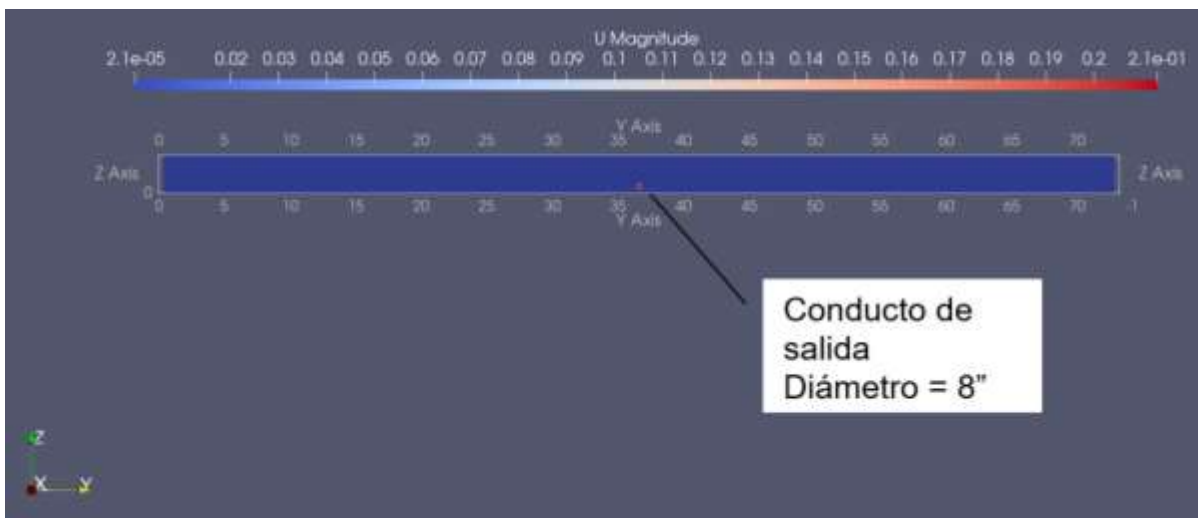


Figura 5-18. Vista en la sección de salida del flujo en la laguna 1, $t = 7,200$ s.

- Visualización del campo de velocidades en la laguna 2 y 3.

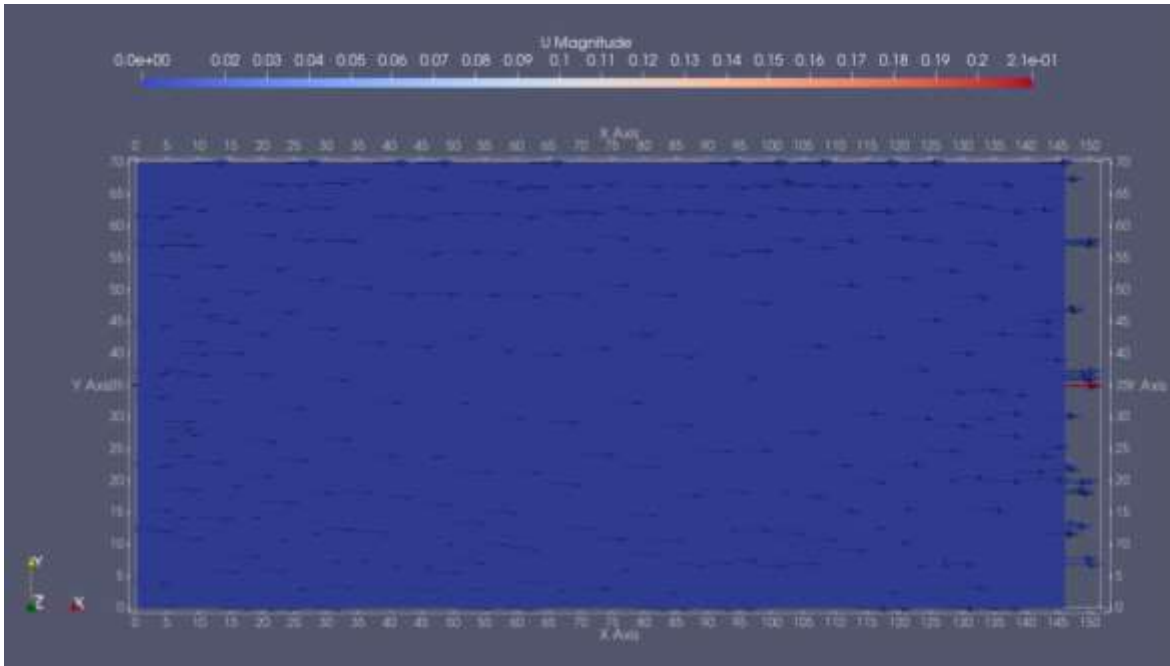


Figura 5-19. Vista en planta del campo de velocidades en la laguna 2 y 3, con $z = 3.0$ m y $t = 7,200$ s.

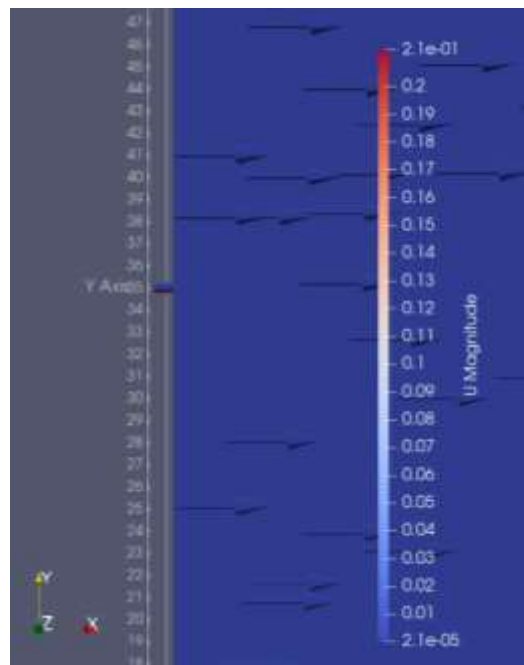


Figura 5-20. Vista ampliada en planta del campo de velocidades en la laguna 2 y 3, con $z = 3.0$ m y $t = 7,200$ s.

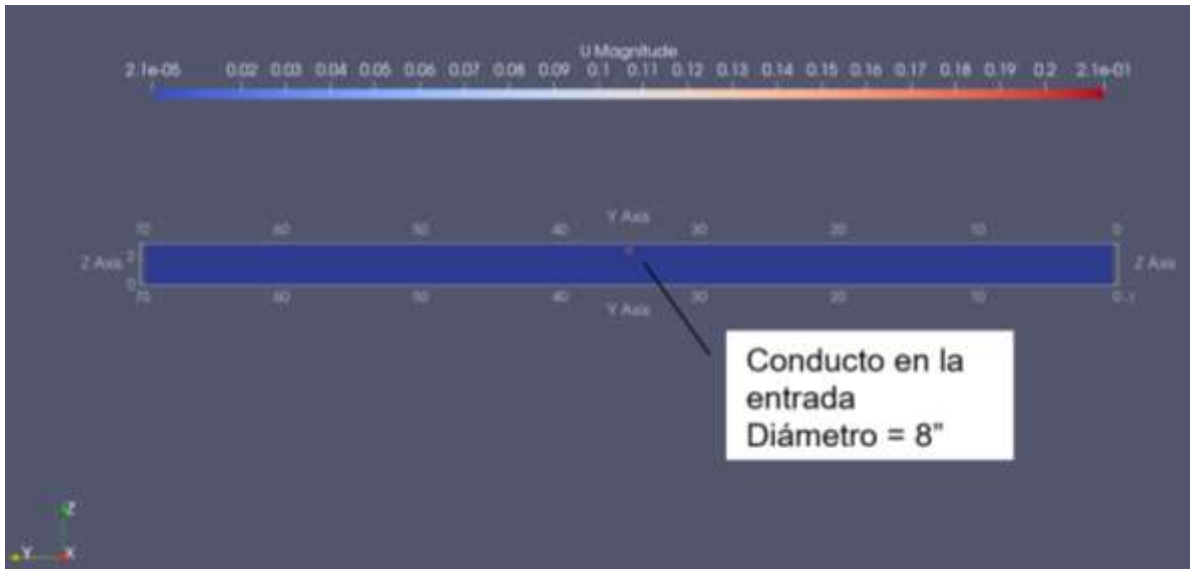


Figura 5-21. Vista en la sección de entrada del flujo en la laguna 2 y 3, $t = 7,200$ s.

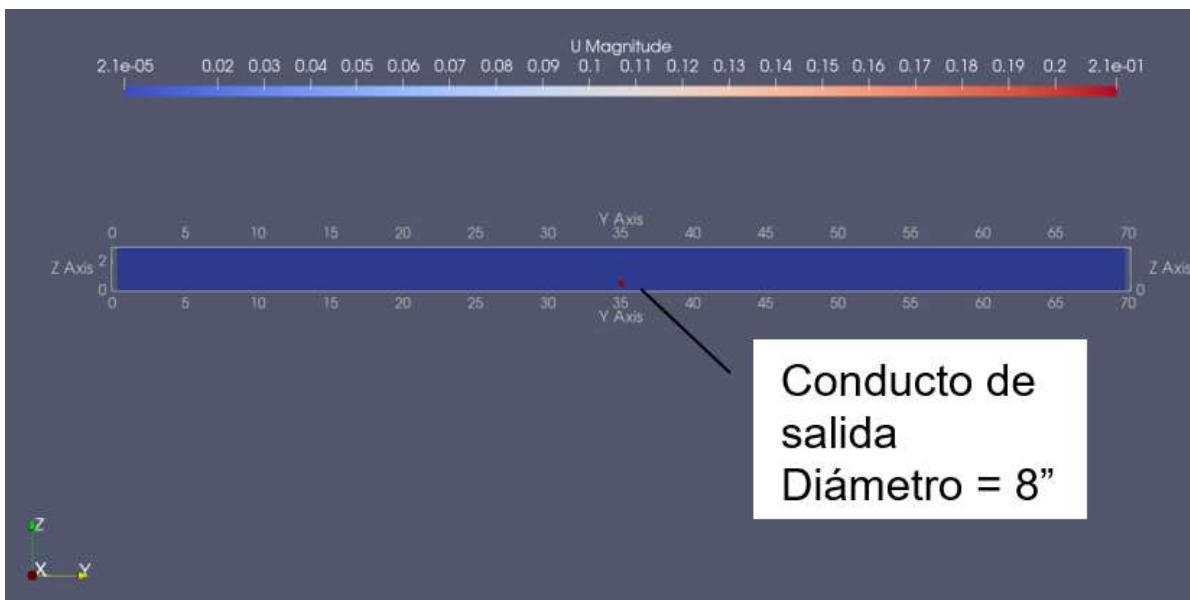


Figura 5-22. Vista en la sección de salida del flujo en la laguna 2 y 3, $t = 7,200$ s.

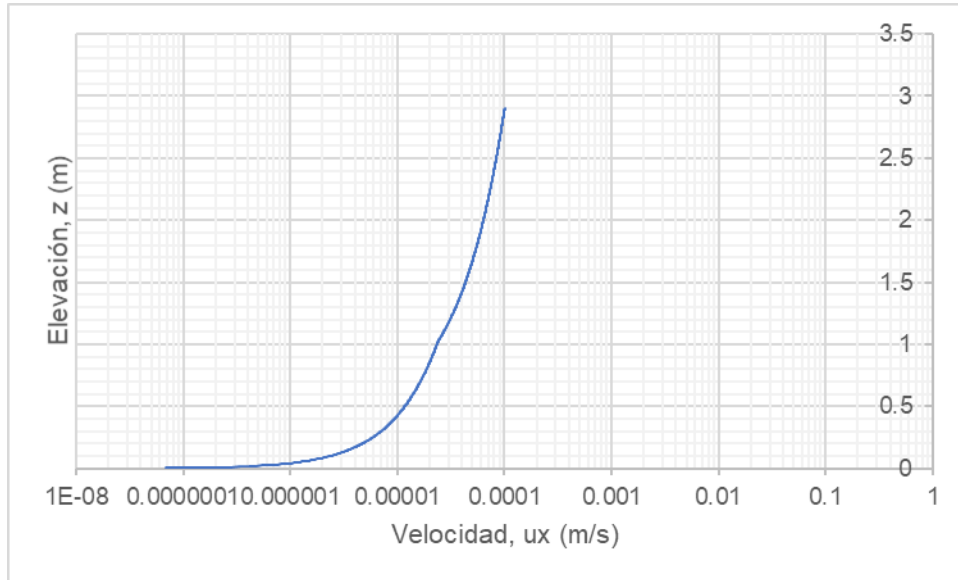


Figura 5-23. Perfil de velocidad en el centro de la Laguna 1, $t = 7,200$ s.

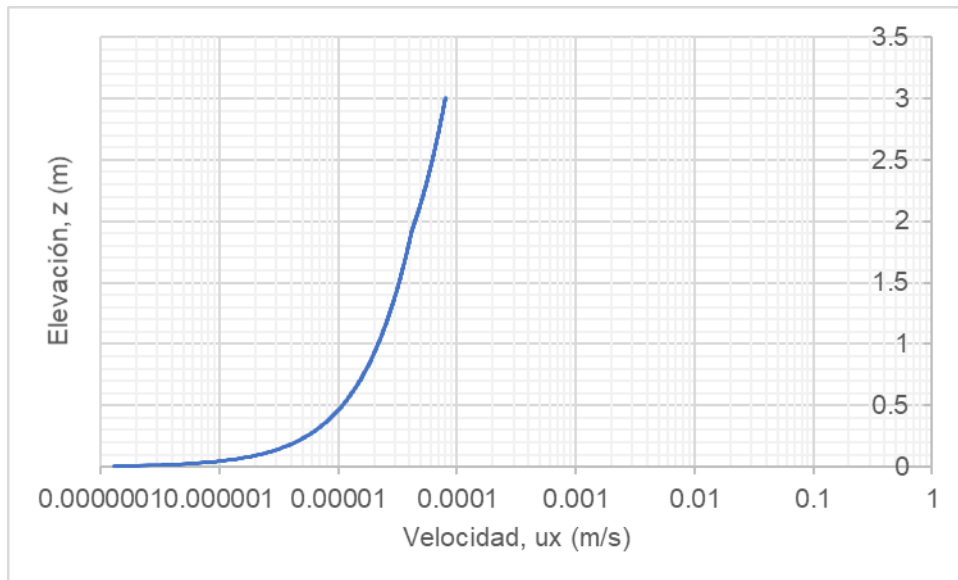


Figura 5-24. Perfil de velocidad en el centro de la Laguna 2 y 3, $t = 7,200$ s.

Los datos de velocidad para la simulación de la ecuación de difusión – advección – reacción en las lagunas 1, 2 y 3 serán U_x , U_y y U_z , los cuales se guardaron en archivos en formato *txt* para su utilización correspondiente. En las siguientes figuras se muestran, a manera de ejemplo, estos ~~los~~ archivos *txt* para incorporarlos a Python.

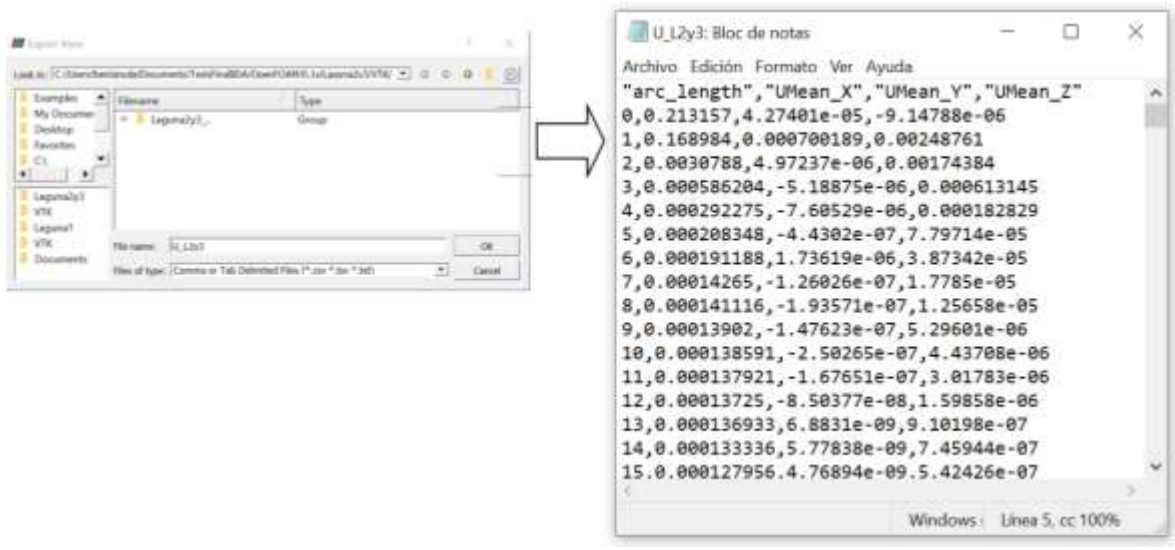


Figura 5-25. Exportación de datos Ux, Uy y Uz de ParaView a archivo txt

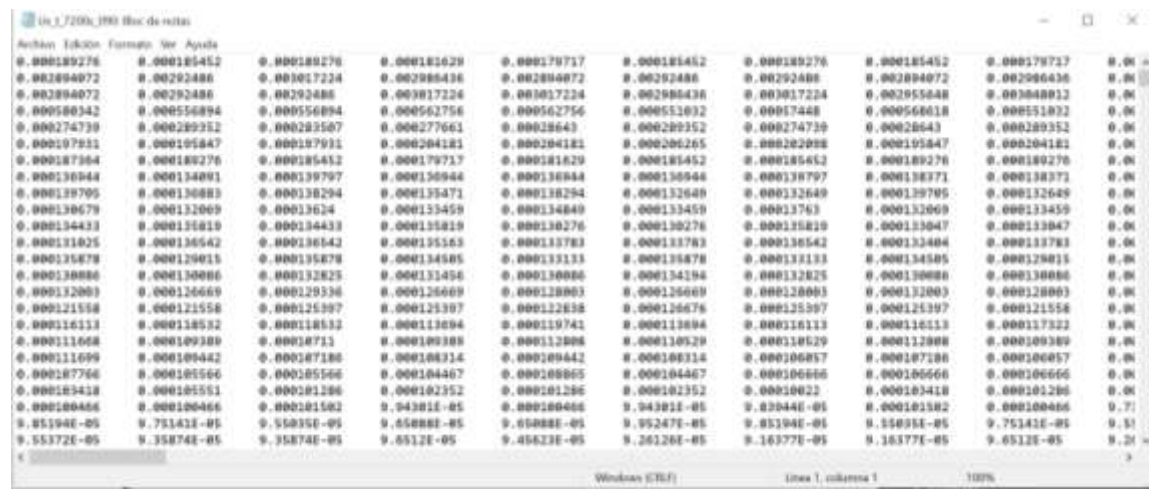


Figura 5-26. Procesamiento de los datos de velocidad para su utilización en Python en archivo txt.

5.5 Simulación del proceso de biodegradación aerobia 3D

En este caso la concentración de oxígeno disuelto permanecerá constante con un valor de 6 mg/l , este valor es el requerido por normatividad de la Agencia de Protección Ambiental (EPA, USA), lo cual se logrará con la utilización automática de aireadores en la laguna 1, 2 y 3. Los datos serán los siguientes:

$$V_m^i = 1.0 \text{ días}^{-1}$$

$$i = 1,2$$

$$K_h^i = 0.1 \text{ mg/l}$$

$$i = 1,2$$

$$k_{12} = 2.0$$

$$k_{21} = 0.5$$

c_1 : Oxígeno disuelto

c_2 : Contaminante orgánico

Nota: Los valores corresponden al estudio de Celia M. A. y J. S. Kindred (1989), los cuales se adecuan a las condiciones del ejemplo de estudio.

El contaminante orgánico será el sustrato de carbono orgánico, se estimó mediante la correlación de Demanda Biológica de Oxígeno (DBO_5) y Carbono Orgánico Disuelto (COD), la expresión es la siguiente (Day, 2011):

$$DBO_5 = 2.9COD$$

El afluente tiene un valor inicial de $DBO = 200 \text{ mg/l}$

Por lo tanto, el valor inicial de $COD = \frac{200}{2.9} = 69 \text{ mg/l}$

La especie estacionaria se mantendrá constante debido a una concentración óptima de oxígeno disuelto y se fija en:

$$X_1 = 3.0 \text{ mg/l}$$

Tiempo de simulación en la laguna 1, 2 y 3:

- Laguna 1

$$t = 29.5 \text{ días}$$

- Laguna 2 y 3

$$t = 14.75 \text{ días}$$

Nota: El tiempo de simulación corresponde a la política de operación de las lagunas.

Se utilizó una malla de:

$$\Delta x = 1 \text{ m}$$

Con un diferencial de tiempo:

$$\Delta t = 0.01 \text{ días}$$

Los valores de velocidad U_x , U_y y U_z se tomaron de la simulación de OpenFOAM.

La condición inicial y las condiciones de frontera se representan en las siguientes figuras:

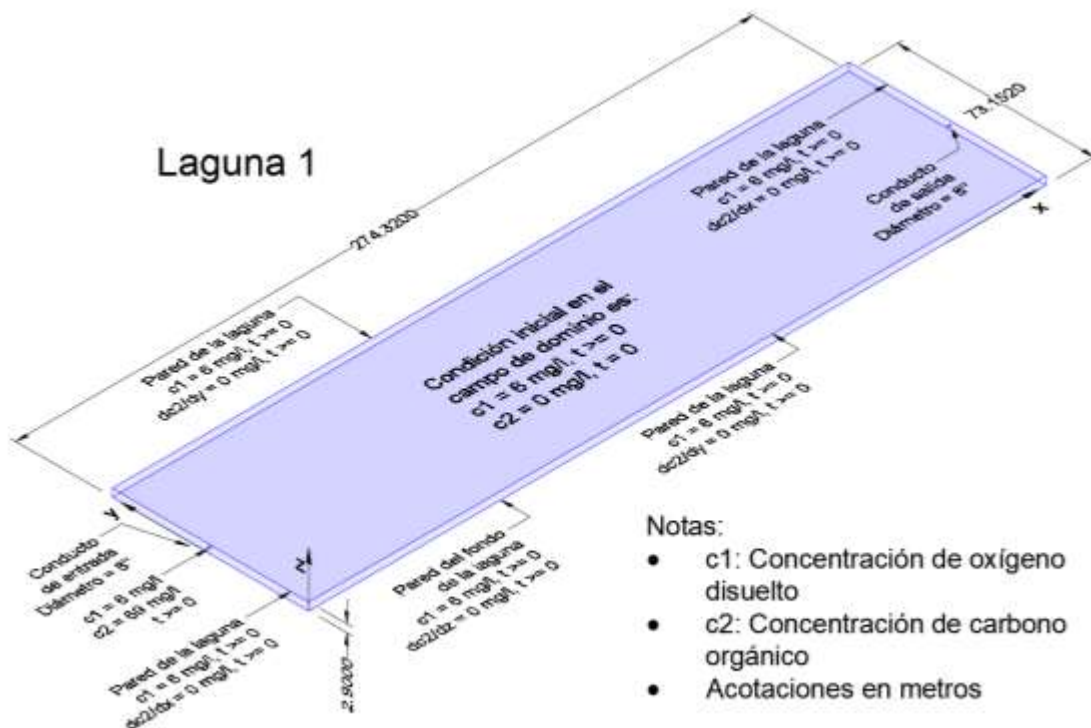


Figura 5-27. Condición inicial y condiciones de frontera para la laguna 1 del proceso de biodegradación aerobia.

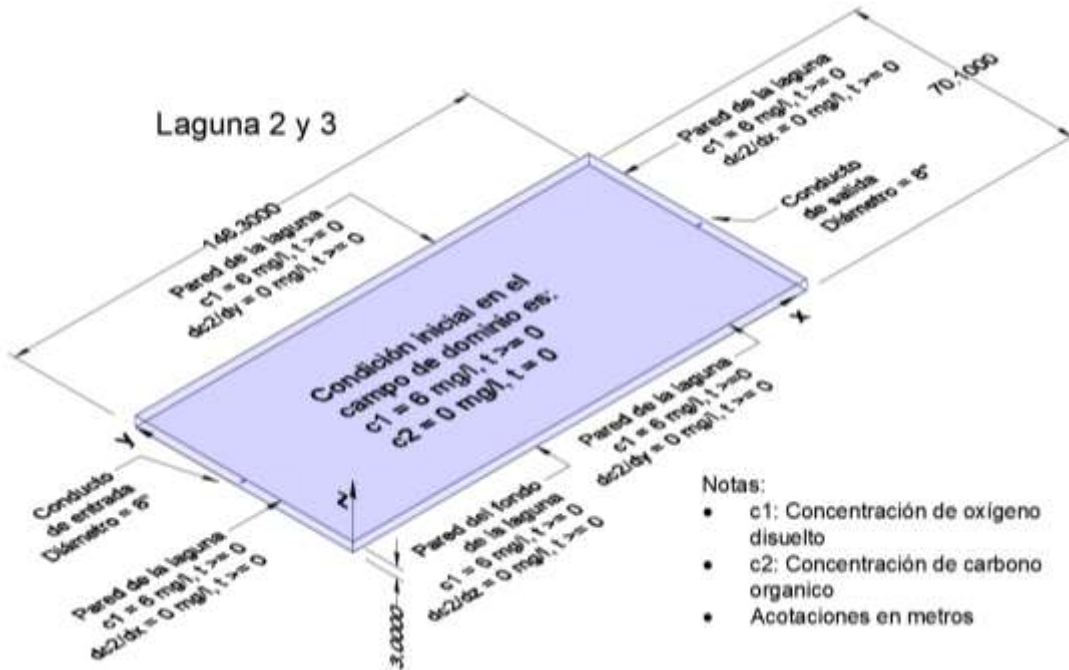


Figura 5-28. Condición inicial y condiciones de frontera para la laguna 2 y 3 del proceso de biodegradación aerobia.

El coeficiente de difusión para contaminantes en aguas residuales se determinó con la siguiente expresión (Lagod, Sobczuk, Suchorab, & Widomski, 2009):

$$D = 7.25HU_* \left(\frac{U}{U_*} \right)^{0.25} \quad (5.6)$$

Donde:

H : es la profundidad hidráulica media (m)

U : velocidad media en la sección transversal $\left(\frac{m}{s} \right)$

U_* : velocidad de corte o velocidad de fricción $\left(\frac{m}{s} \right)$

$$U_* = \sqrt{gR_h S_f} \quad (5.7)$$

Donde:

g : aceleración de la gravedad $\left(\frac{m^2}{s}\right)$

R_h : radio hidráulico (m)

S_f : pendiente de fricción (m), formula de Manning

$$S_f = \left(\frac{nQ}{AR_h^{\frac{2}{3}}} \right)^2$$

(5.8)

Donde:

n : coeficiente de Manning

Q : caudal (m^3/s)

R_h : radio hidráulico

El coeficiente de Manning corresponderá a 0.013, lo cual equivale a concreto terminado con llana metálica (Chow, 1994).

De las ecuaciones (5.6), (5.7) y (5.8) se determinó el coeficiente de difusión para la Laguna 1, 2 y 3.

- Laguna 1

$$S_f = \left(\frac{nQ}{AR_h^{\frac{2}{3}}} \right)^2$$

$$S_f = \left(\frac{(0.0145)(0.035)}{(212.14) \left(\frac{212.14}{78.952} \right)^{\frac{2}{3}}} \right)^2 = 1.532 \times 10^{-12}$$

$$U_* = \sqrt{gR_h S_f}$$

$$U_* = \sqrt{(9.81)(2.687)(1.532 \times 10^{-12})} = 6.355 \times 10^{-6} \text{ m/s}$$

Por lo tanto, se tiene:

$$D_{L1} = 7.25HU_* \left(\frac{U}{U_*} \right)^{0.25}$$

$$D_{L1} = 7.25(2.9)(6.355 \times 10^{-6}) \left(\frac{0.000165}{6.355 \times 10^{-6}} \right)^{0.25} = 0.0003016 \times 10^{-4} \text{ m}^2/\text{s}$$

$$D_{L1} = 26.06 \text{ m}^2/\text{día}$$

- Laguna 2 y 3

$$S_f = \left(\frac{nQ}{AR_h^{\frac{2}{3}}} \right)^2$$

$$S_f = \left(\frac{(0.0145)(0.035)}{(210.3) \left(\frac{210.3}{76.1} \right)^{\frac{2}{3}}} \right)^2 = 1.5017 \times 10^{-12}$$

$$U_* = \sqrt{gR_h S_f}$$

$$U_* = \sqrt{(9.81)(2.7634)(1.5017 \times 10^{-12})} = 6.38 \times 10^{-6} \text{ m/s}$$

Por lo tanto, se tiene:

$$D_{L2yL3} = 7.25HU_* \left(\frac{U}{U_*} \right)^{0.25}$$

$$D_{L2yL3} = (7.25)(3.0)(6.38 \times 10^{-6}) \left(\frac{0.0001664}{6.38 \times 10^{-6}} \right)^{0.25} = 0.00031361 \times 10^{-4} \text{ m}^2/\text{s}$$

$$D_{L2yL3} = 27.09 \text{ m}^2/\text{día}$$

Ahora se calcula el coeficiente de difusión mediante la media para la simulación:

$$D = \frac{D_{L1} + D_{L2yL3}}{2}$$

$$D = \frac{26.06 + 27.09}{2} = 26.575 \text{ m}^2/\text{día}$$

Se realizó un promedio de los coeficientes de difusión debido a que las lagunas tienen las mismas características de funcionamiento.

Una vez definidos los parámetros y datos del problema se determinó la simulación para la laguna 1, 2 y 3 en los tiempos de simulación de 10, 30 y 59 días. Cabe destacar que la simulación se desarrolló utilizando las tres lagunas en el mismo sistema de solución.

5.5.1 Resultados

5.5.1.1 Simulación numérica para concentración de oxígeno disuelto (c_1)

Cabe destacar que el oxígeno disuelto permanece constante en todos los tiempos de simulación debido a que es la condición óptima para el proceso de biodegradación aerobia.

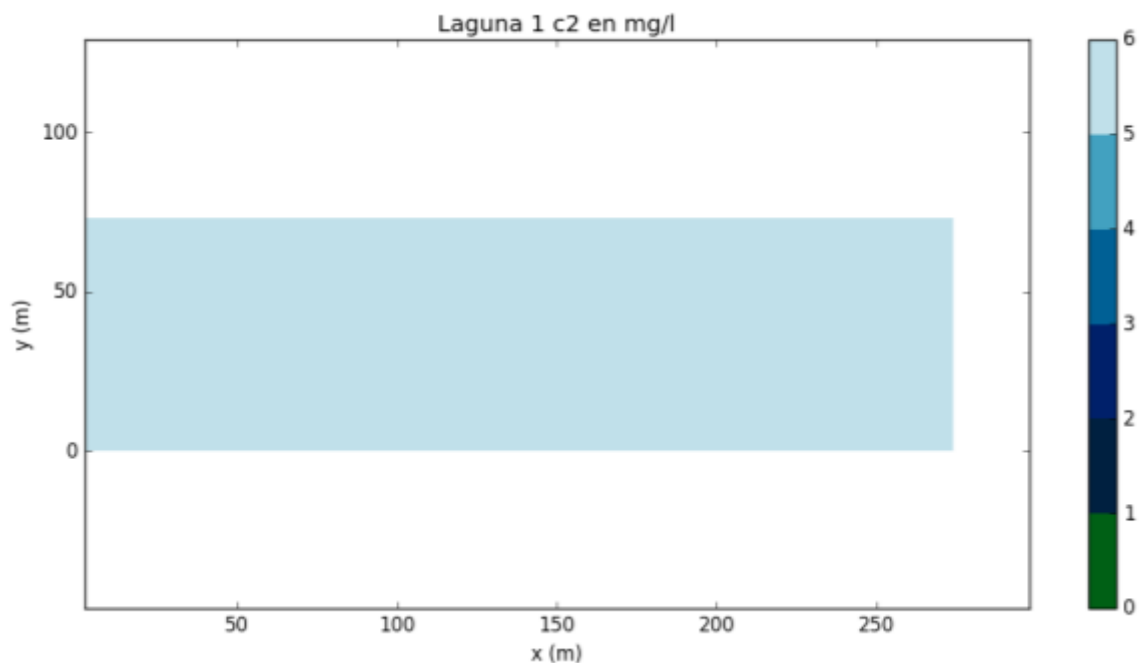


Figura 5-29 Oxígeno disuelto (c_1) para la laguna 1 en todos los tiempos de simulación.

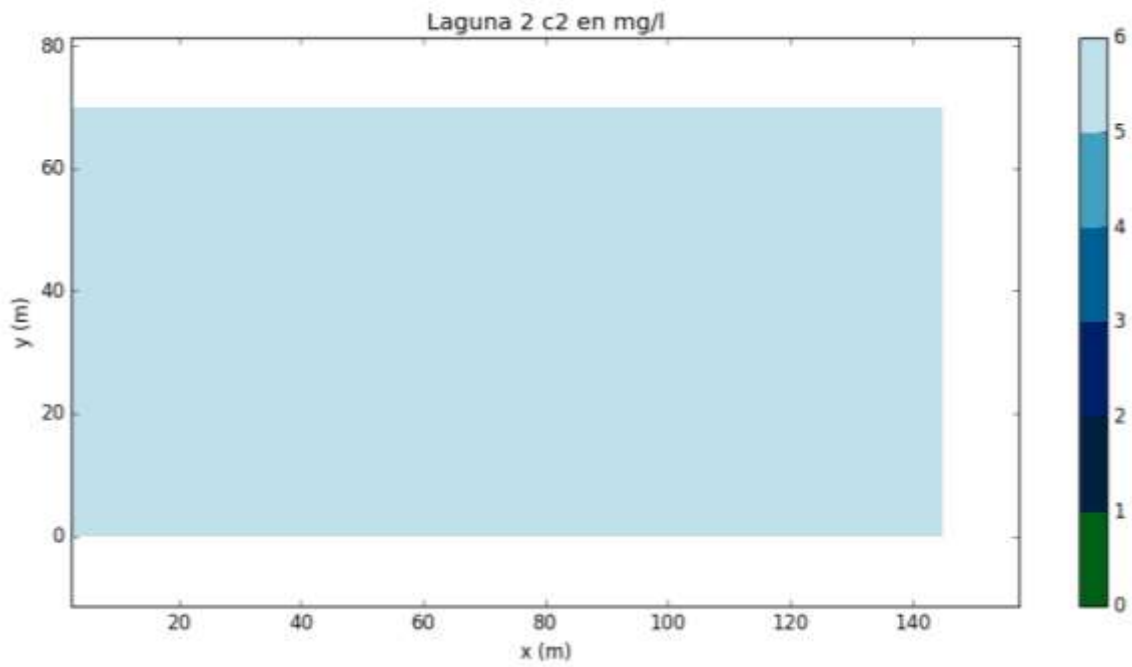


Figura 5-30 Oxígeno disuelto (c1) para la laguna 2 en todos los tiempos de simulación.

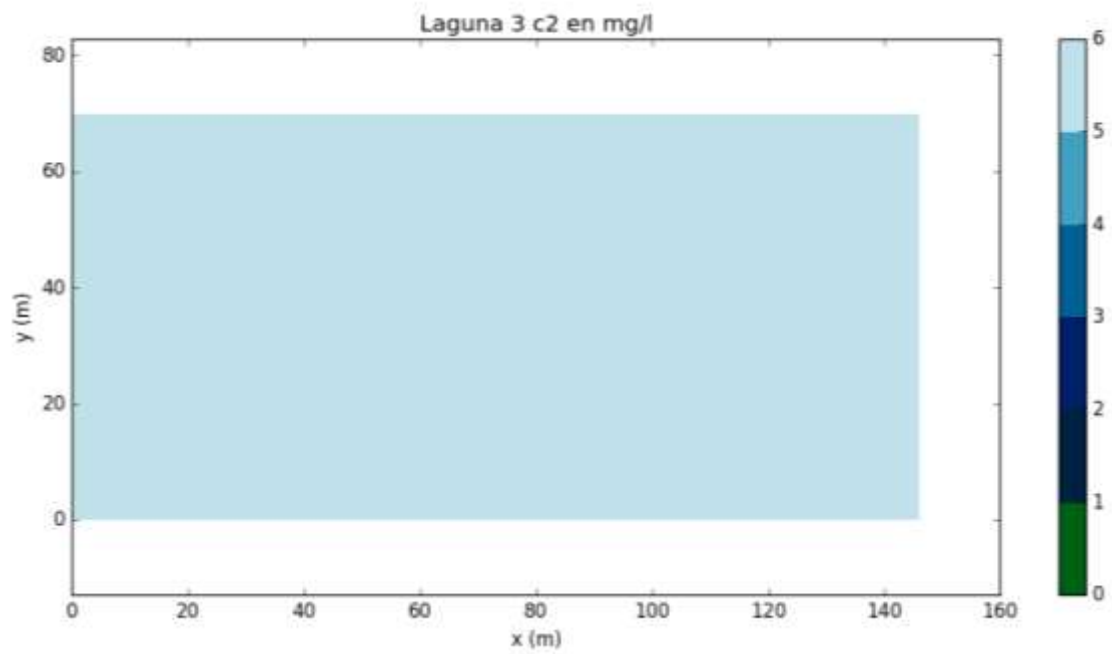


Figura 5-31 Oxígeno disuelto (c1) para la laguna 3 en todos los tiempos de simulación.

5.5.1.2 Simulación numérica para concentración de carbono orgánico disuelto (c_2)

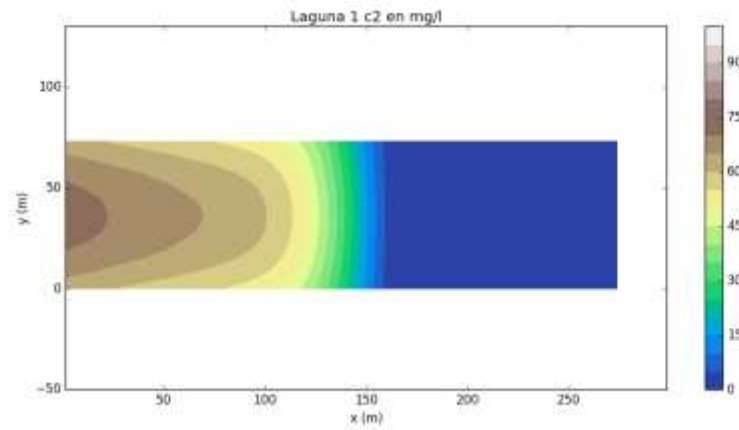


Figura 5-32 Concentración de c_2 en la laguna 1, elevación = 0.9 m y $t = 10$ días.

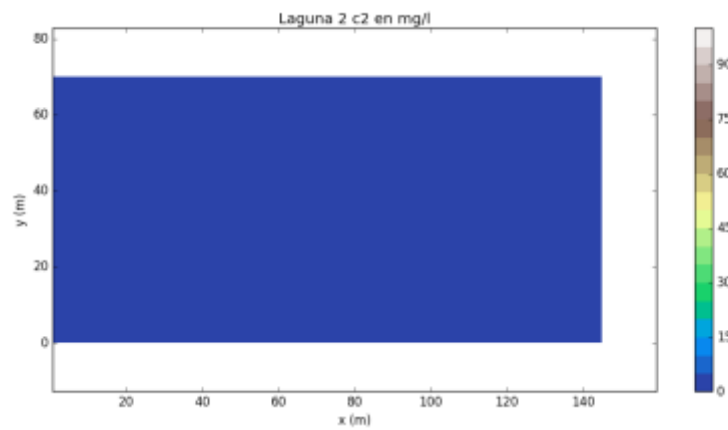


Figura 5-33 Concentración de c_2 en la laguna 2, elevación = 0.9 m y $t = 10$ días.

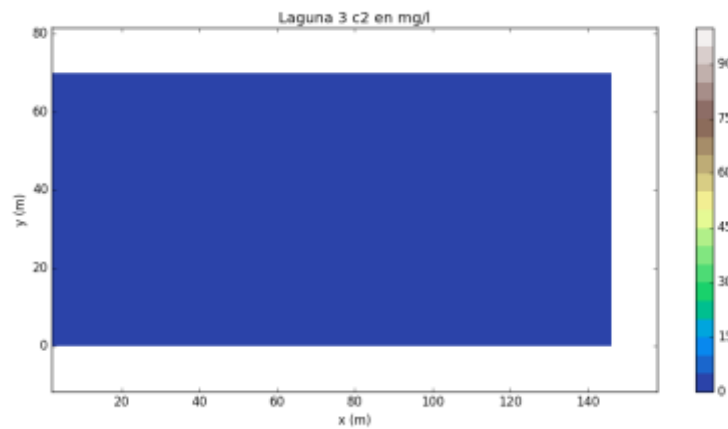


Figura 5-34 Concentración de c_2 en la laguna 3, elevación = 0.9 m y $t = 10$ días.

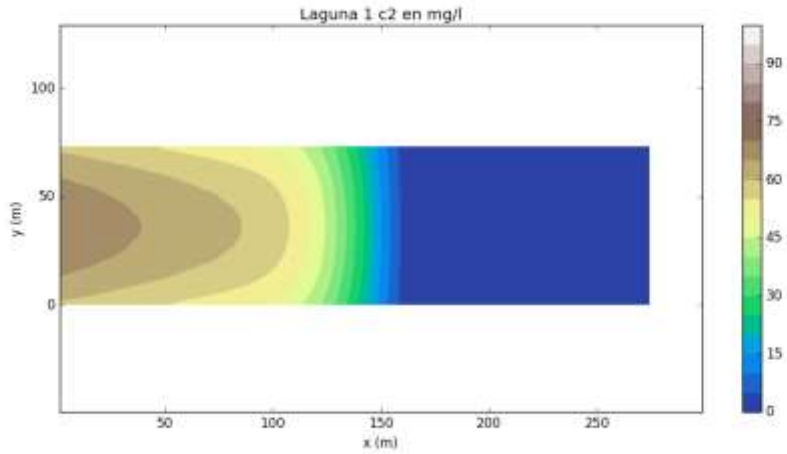


Figura 5-35 Concentración de c_2 en la laguna 1, elevación = 1.9 m y $t = 10$ días.

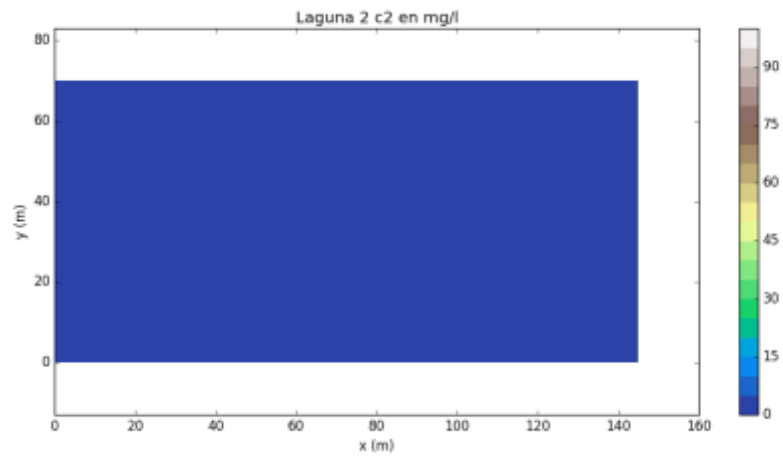


Figura 5-36 Concentración de c_2 en la laguna 2, elevación = 1.9 m y $t = 10$ días.

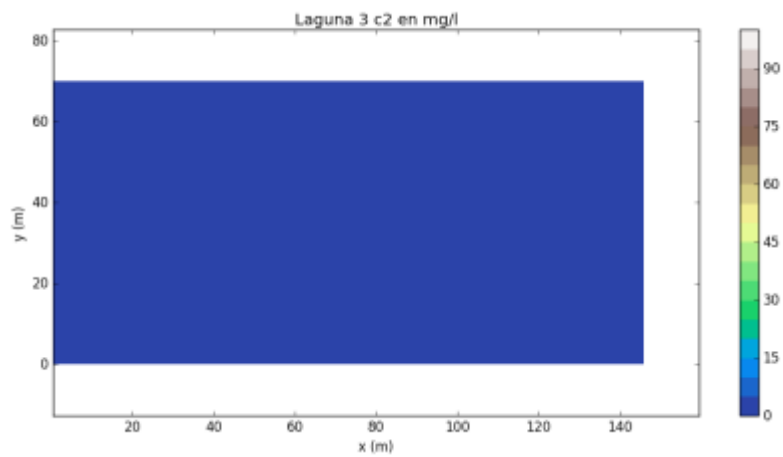


Figura 5-37 Concentración de c_2 en la laguna 3, elevación = 1.9 m y $t = 10$ días.

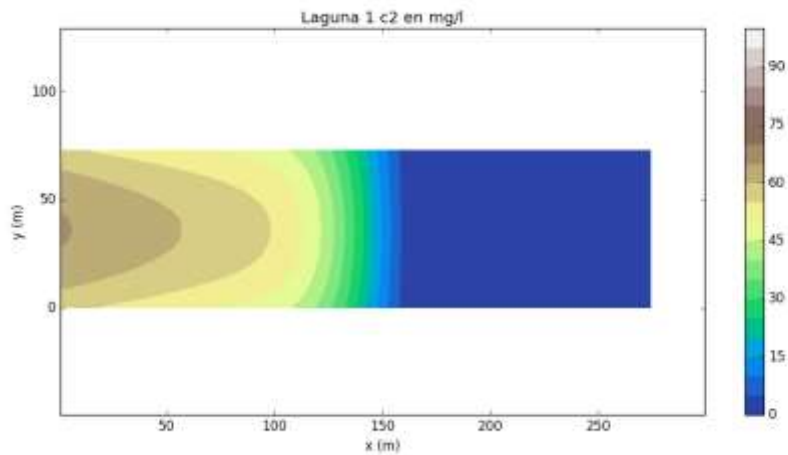


Figura 5-38 Concentración de c_2 en la laguna 1, elevación = 2.9 m y t = 10 días

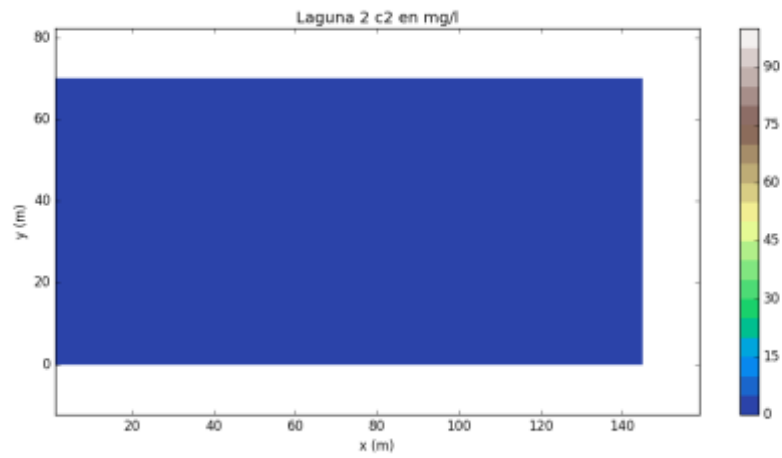


Figura 5-39 Concentración de c_2 en la laguna 2, elevación = 2.9 m y t = 10 días

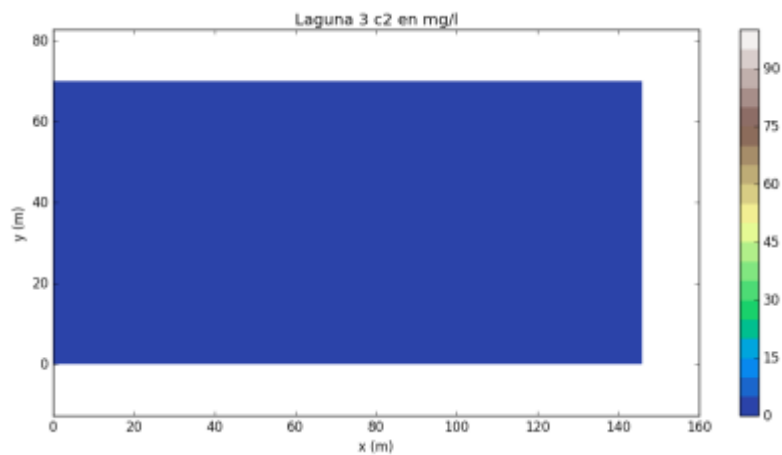


Figura 5-40 Concentración de c_2 en la laguna 3, elevación = 2.9 m y t = 10 días.

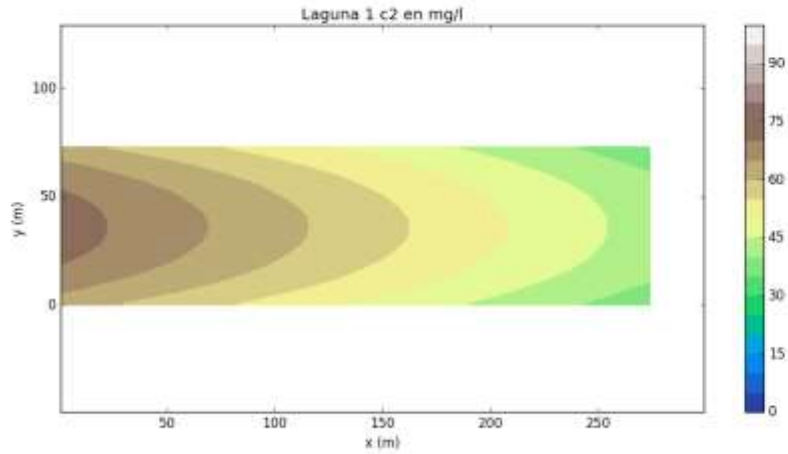


Figura 5-41 Concentración de c_2 en la laguna 1, elevación = 0.9 m y t = 30 días.

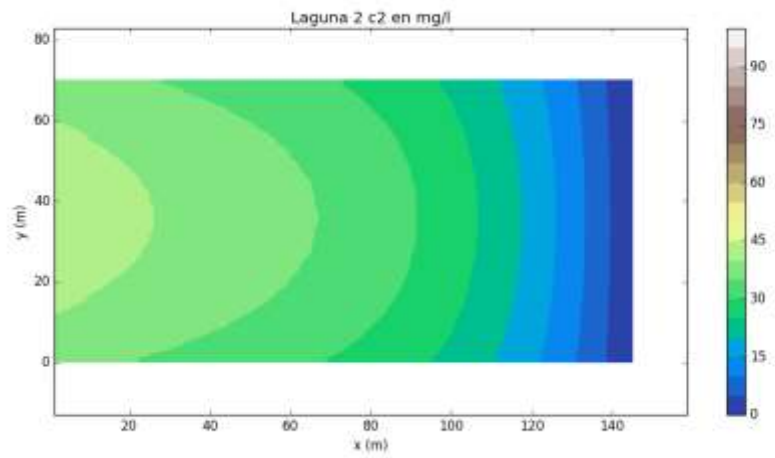


Figura 5-42 Concentración de c_2 en la laguna 2, elevación = 0.9 m y t = 30 días.

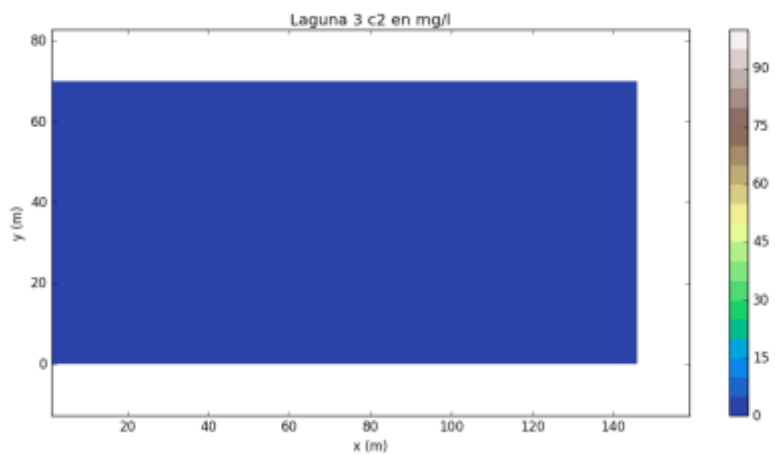


Figura 5-43 Concentración de c_2 en la laguna 3, elevación = 0.9 m y t = 30 días.

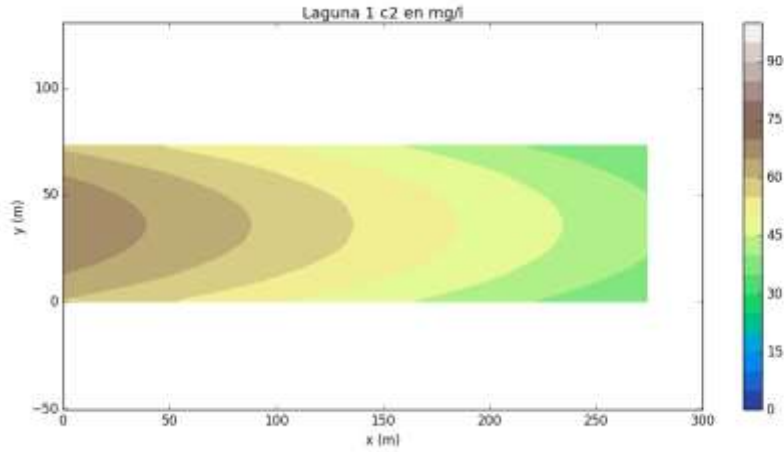


Figura 5-44 Concentración de c_2 en la laguna 1, elevación = 1.9 m y t = 30 días.

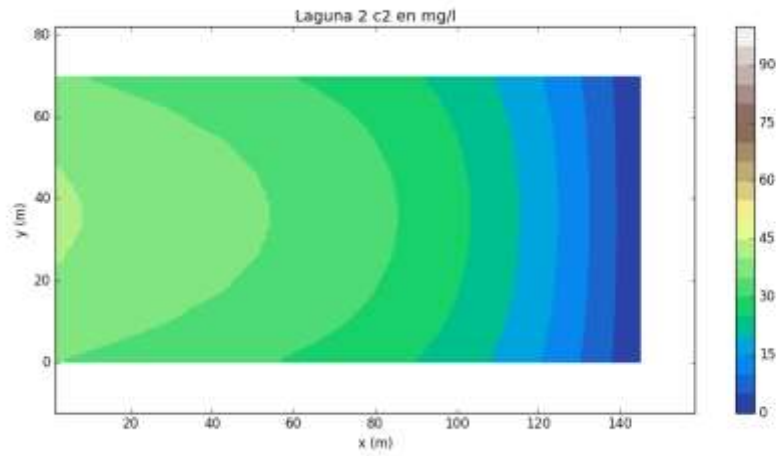


Figura 5-45 Concentración de c_2 en la laguna 2, elevación = 1.9 m y t = 30 días.

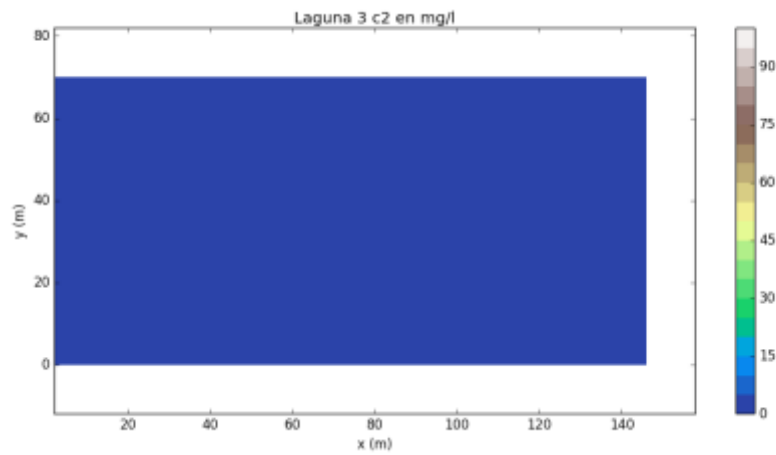


Figura 5-46 Concentración de c_2 en la laguna 3, elevación = 1.9 m y t = 30 días.

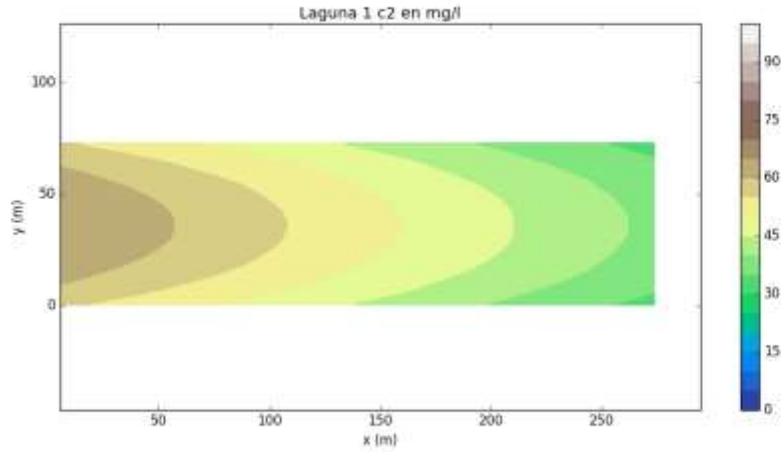


Figura 5-47 Concentración de c_2 en la laguna 1, elevación = 2.9 m y $t = 30$ días.

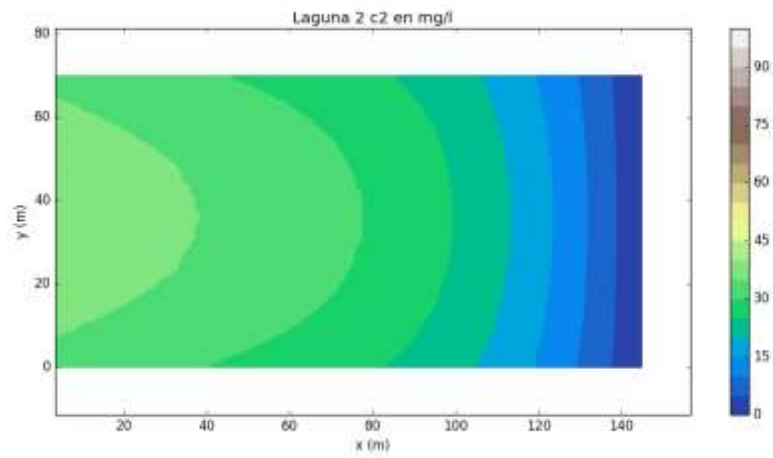


Figura 5-48 Concentración de c_2 en la laguna 2, elevación = 2.9 m y $t = 30$ días.

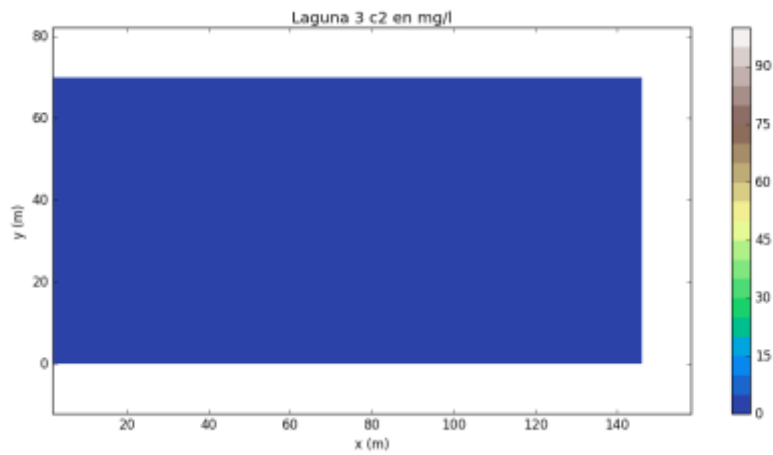


Figura 5-49 Concentración de c_2 en la laguna 3, elevación = 2.9 m y $t = 30$ días

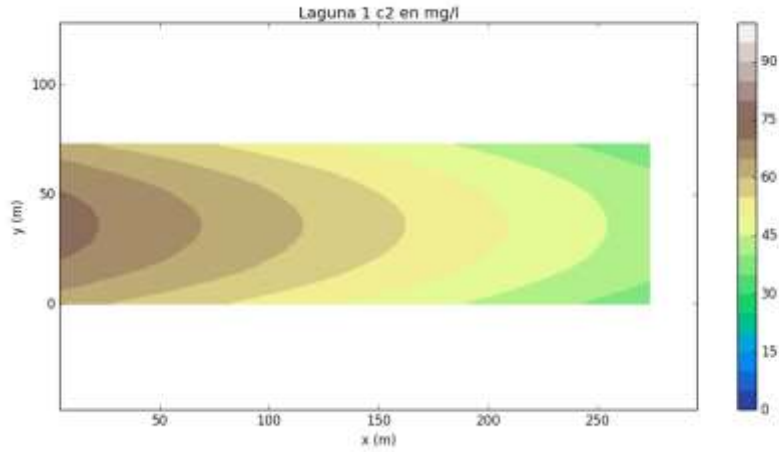


Figura 5-50 Concentración de c_2 en la laguna 1, elevación = 0.9 m y $t = 59$ días.

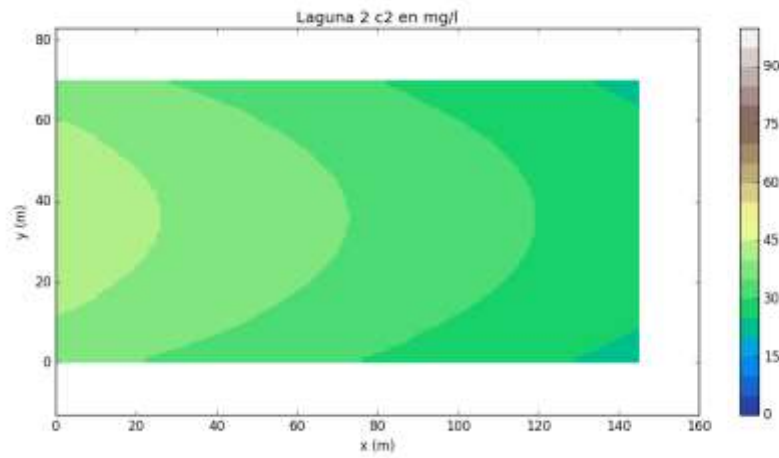


Figura 5-51 Concentración de c_2 en la laguna 2, elevación = 0.9 m y $t = 59$ días.

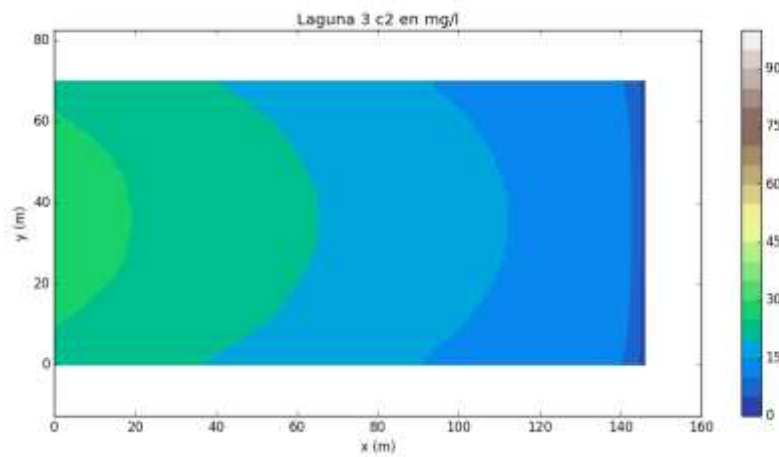


Figura 5-52 Concentración de c_2 en la laguna 3, elevación = 0.9 m y $t = 59$ días.]

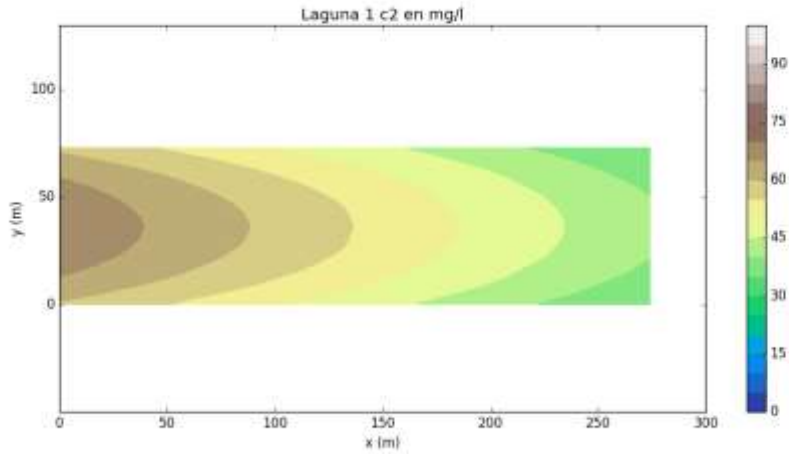


Figura 5-53 Concentración de c_2 en la laguna 1, elevación = 1.9 m y t = 59 días

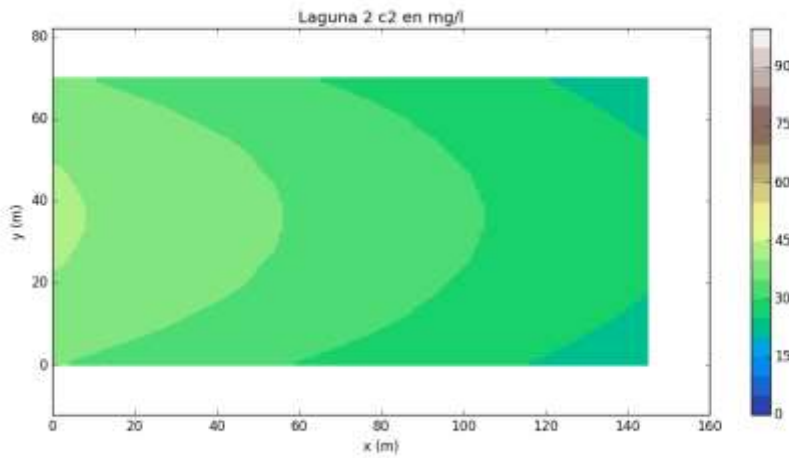


Figura 5-54 Concentración de c_2 en la laguna 2, elevación = 1.9 m y t = 59 días

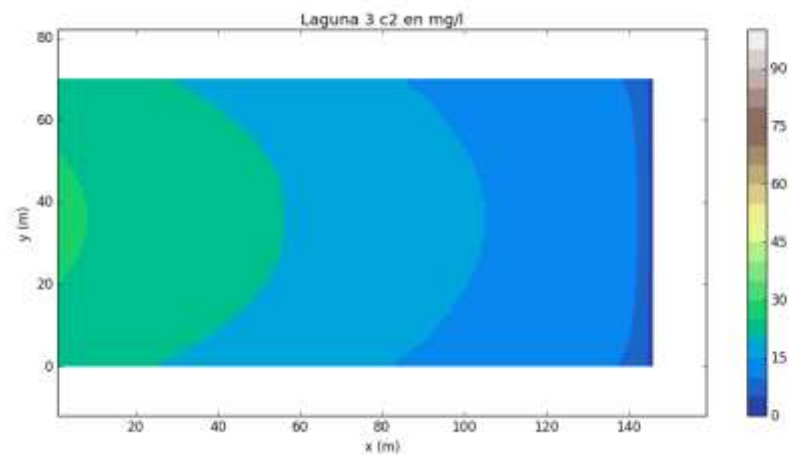


Figura 5-55 Concentración de c_2 en la laguna 3, elevación = 1.9 m y t = 59 días.

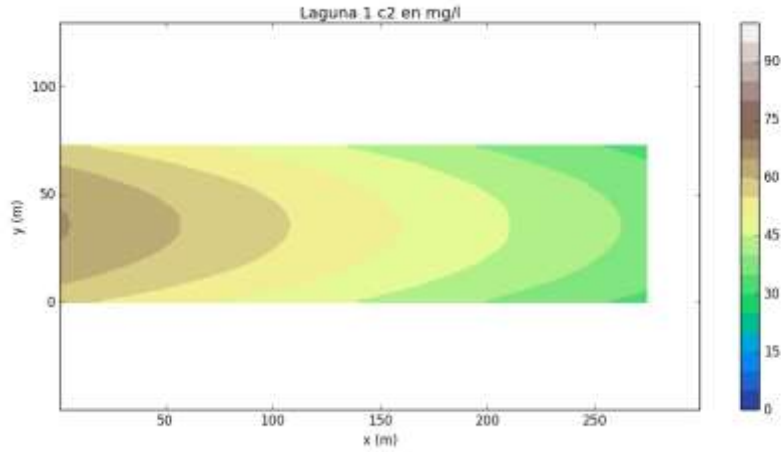


Figura 5-56 Concentración de c_2 en la laguna 1, elevación = 2.9 m y t = 59 días

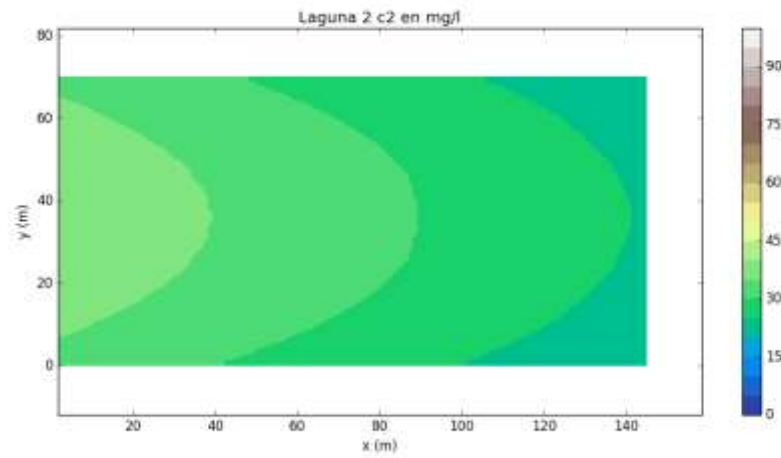


Figura 5-57 Concentración de c_2 en la laguna 2, elevación = 2.9 m y t = 59 días

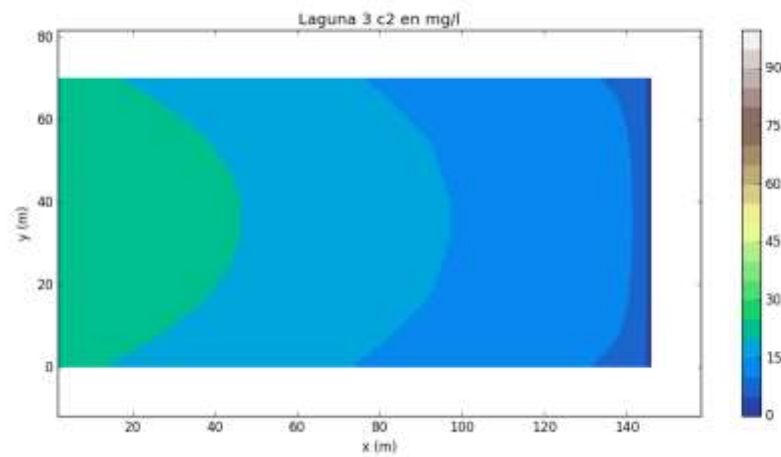


Figura 5-58 Concentración de c_2 en la laguna 3, elevación = 2.9 m y t = 59 días

En las figuras se observa una disminución en la concentración de carbono orgánico disuelto (c_2) desde la laguna 1 hasta la laguna 3 para las tres elevaciones de estudio (0.9 m, 1.9 m y 2.9 m) durante el tiempo de simulación, dando como resultado una concentración en la sección de salida de la laguna 3 de 0 a 8 mg/l, lo cual es equivalente a un rango de 0 a 30 mg/l de DBO_5 , se calculó la equivalencia con la expresión de Day, 2011; cabe destacar que esta concentración de salida se considera aceptable para la descarga al río Piscataquils de acuerdo a normas del Departamento de Protección Medioambiental (EPA, por sus siglas en inglés) de Estados Unidos.

Para la concentración de oxígeno disuelto (c_1) se determinó que se mantuviera constante con un valor de 6 mg/l para tener un óptimo funcionamiento de biodegradación aerobia en las tres lagunas.

El valor máximo del número de Courant del modelo numérico es el siguiente:

$$Cr = \frac{(14.38)(0.01)}{1} = 0.1438$$
$$0.1438 < 1$$

Por lo tanto, la estabilidad del modelo numérico es aceptable.

6 Conclusiones

Se consiguió la solución del modelo numérico CFD unidimensional (1D), bidimensional (2D) y tridimensional (3D) para el proceso de biodegradación aerobia y también con su respectiva representación en diferentes ejemplos.

Cada componente de las ecuaciones del proceso de biodegradación aerobia se estudió de manera particular y detallada para comprender su funcionamiento e importancia en las ecuaciones.

Inicialmente se desarrollaron simulaciones 1D y 2D para visualizar el comportamiento de las ecuaciones, posteriormente se desarrolló el modelo 3D con un caso real utilizando softwares libres de código abierto desde la generación de mallas 3D hasta la visualización interactiva de resultados.

Para el uso adecuado del modelo numérico CFD se efectuó la verificación del modelo en 1D obteniendo una comparación satisfactoria con los resultados de Celia M. A. y J. S. Kindred (1989).

Se obtuvieron resultados bastante aceptables con respecto a la solución analítica en el ejemplo 2D que consistió en el transporte de un contaminante conservativo.

En el ejemplo 2D del transporte de dos especies con términos de biodegradación aerobia, se observó un buen comportamiento numérico para todos los tiempos de simulación, cabe destacar que el campo de velocidades tiene una dirección rotacional, lo cual puede servir para la aplicación en una planta de tratamiento rotativa.

Para la aplicación del modelo numérico CFD tridimensional en caso real se desarrolló con base al tratamiento de aguas residuales mediante lagunas de estabilización aerobia. Para los datos de simulación se realizó una correlación de la

demanda biológica de oxígeno cinco días de reacción (DBO_5) con carbono orgánico disuelto (c_2). El valor del oxígeno disuelto (c_1) se consideró constante para un funcionamiento óptimo durante el proceso de biodegradación aerobia. Los resultados de este ejemplo concuerdan satisfactoriamente con los datos del caso de estudio y además se tiene estabilidad numérica con base al criterio del número de Courant.

El número de Peclet se utilizó para evaluar el grado de difusión en dirección con el flujo de agua con respecto a la advección, con $Pe > 1$ el transporte de difusión es más pequeño en comparación con la advección.

Con el presente trabajo se logran implementar herramientas para la simulación de procesos de biodegradación aerobia con un análisis numérico y comparativo con resultados satisfactorios. Este estudio también demuestra la aplicación de modelos numéricos para el diseño de plantas de tratamiento de aguas residuales mediante lagunas de estabilización aerobia y puede ser aplicado en diferentes poblados de México debido a sus grandes ventajas en la operación y funcionamiento del sistema.

Mediante este modelo numérico es posible evaluar cómo se desarrollan los procesos de biodegradación aerobia y con esto poder manipular el proceso para fines útiles y de aplicación.

Además, se observa la gran importancia de la utilización de softwares libres que se complementan para la simulación numérica como SALOME, OpenFOAM, ParaVIEW y Python.

Para trabajo a futuro queda pendiente incorporar procesos de biodegradación más complejos como por ejemplo donde se incluyan diferentes consorcios de biomasa variable consumiendo diferentes compuestos.

7 Referencias

- Aguilar Chavez, A., & Laurel Castillo, J. (2011). *Simulación Física y Matemática del flujo en vertedores escalonados*. Jiutepec, Morelos: Instituto Mexicano de Tecnología del Agua.
- Andersson, B., R. A., L. H., M. M., R. S., & B. W. (2012). *Computational Fluid Dynamics for Engineers*. New York: Cambridge University Press.
- Arroyo Correa, V. M. (2005). *Modelación Bidimensional de Flujo y Transporte no Lineales en Medios Porosos aplicando ELLAM*. Jiutepec, Morelos: Instituto Mexicano de Tecnología del Agua.
- Blanco, P. G. (2010). *Eliminación de Clorofenoles mediante la combinación de ozonación previa y biodegradación aerobia*. Ciudad de México : Instituto Politécnico Nacional.
- Blazek, J. (2001). *Computational Fluid Dynamics: Principles and Applications*. Oxford, UK: ELSEVIER.
- Celia, M. A., Kindred, J. S., & I. H. (June 1989). *Contaminant Transport and Biodegradation A Numerical Model for Reactive Transport in Porous Media*. Cambridge USA: Water Resources Research.
- Chow, V. T. (1994). *Hidráulica de canales abiertos*. Bogotá, Colombia: McGRAW-HILL INTERAMERICANA S. A.
- CONAGUA. (2007). *Manual de Agua Potable, Alcantarillado y Saneamiento*. Ciudad de México: Comisión Nacional del Agua.
- Covarrubias, M. I., Velázquez, J. F., Bustamante, W. O., Delgado, C. D., & Escalante, R. M. (2015). *Comparación de resultados experimentales de un Venturi con simulación de dinámica de fluidos computacional*. Jiutepec: Tecnología y Ciencias del Agua.
- Cussler, E. L. (2007). *Diffusion, Mass Transfer in Fluid Systems*. New York: Cambridge University Press.
- Date, A. W. (2005). *Introduction to Computational Fluid Dynamics*. USA: Cambridge University Press.

- Day, P. R. (2011). *Winery Wastewater Management and Recycling*. Australia: Australian Government.
- DEP, M. (2003). *An Informational Resource for Operators of Lagoon Systems*. Maine, USA: Department of Environmental Protection State of Maine.
- Eskander, S. B., & Saleh, H. M. (2017). *Biodegradation: Process Mechanism*. Giza, Egypt: ResearchGate.
- F. G., C. K., T. F., F. L., & R. S. (2016). *Assessment of particle-tracking models for dispersed particle-laden flows implemented in OpenFOAM and ANSYS FLUENT*. Graz: Taylor & Francis FGroup.
- H. W., Sharpley, R. C., & S. M. (1996). *An ELLAM Scheme for Advection-Diffusion Equations in Multi-Dimensions*. Columbia, Carolina del Sur: University of South Carolina.
- Hamilton, D. (2019). *DYRESM-CAEDYM*. Australia: DYRESM-CAEDYM.
- Hodges, B. R. (2006). *CWR-ELCOM*. Texas, USA: The University of Texas at Austin.
- I. Essaid, H., & A. Bekins, B. (1997). *BIOMOC, A Multispecies Solute-Transport Model with Biodegradation USGS*. Menlo Park, Cal.: Water Resources Investigations Report.
- Lagod, G., Sobczuk, H., Suchorab, Z., & Widomski, M. (2009). *Advection-Dispersion Pollutant and Dissolved Oxygen Transport as a part of Sewage Biodegradation Model*. Lublin, Poland: Lublin University of Technology.
- Millán Barrera, C., Arroyo Correa, V., & Laurel Castillo, J. A. (2013). *Modeling contaminant transport with aerobic biodegradation in a shallow water body*. Beijing, China: Tsinghua University Press.
- OpenFOAM. (2018). *Software Open Field Operation and Manipulation*. Londres, Inglaterra: Imperial College London.
- ParaView. (2018). *Software ParaView*. USA: ParaView.
- Rivas, A. (2007). *Análisis Diferencial de Flujos*. San Sebastián, España: Tecnun, Escuela Superior de Ingenieros de la Universidad de Navarra.
- Román, F. S. (2017). *Hidrología Superficial y Subterránea*. Salamanca, España: Createspace Independent Pub.

SALOME. (2018). *SALOME*. Francia.

Socolofsky, S. A., & Jirka, G. H. (2002). *Environmental Fluid Mechanics* .

Karlsruhe: Karlsruhe University.

Versteeg, H. K., & W. M. (2007). *An Introduction to Computational Fluid Dynamics*.

Harlow, England: Pearson Education Limited.

Wang, Y., M. P., & L. W. (2018). *Spillway jet regime and total dissolved gas prediction with a multiphase flow model*. Iowa, USA: Journal of Hydraulic Research.

Research.

8 Anexos

8.1 Anexo 1. Modelo numérico para la solución unidimensional (1D) con términos de biodegradación aerobia

Los pasos para el modelo unidimensional son los siguientes:

Paso 1. Discretización de los términos de la ecuación (4.1)

- Diferenciación temporal hacia adelante

$$\frac{\partial c_1}{\partial t} \cong \frac{C_{1i}^{n+1} - C_{1i}^n}{\Delta t}$$

- Diferenciación espacial central

$$\frac{\partial c_1}{\partial x} \cong \frac{C_{1i+1}^n - C_{1i-1}^n}{2\Delta x}$$

- Discretización de la segunda derivada espacial

$$\frac{\partial^2 c_1}{\partial x^2} \cong \frac{C_{1i+1}^n - 2C_{1i}^n + C_{1i-1}^n}{(\Delta x)^2}$$

Paso 2. Discretización de los términos de la ecuación (4.2)

- Diferenciación temporal hacia adelante

$$\frac{\partial c_2}{\partial t} \cong \frac{C_{2i}^{n+1} - C_{2i}^n}{\Delta t}$$

- Diferenciación espacial central

$$\frac{\partial c_2}{\partial x} \cong \frac{C_{2i+1}^n - C_{2i-1}^n}{2\Delta x}$$

- Discretización de la segunda derivada espacial

$$\frac{\partial^2 c_2}{\partial x^2} \cong \frac{C_{2i+1}^n - 2C_{2i}^n + C_{2i-1}^n}{(\Delta x)^2}$$

Paso 3. Sustitución de las ecuaciones del Paso 1 y Paso 2 en las ecuaciones (4.1) y (4.2).

La sustitución queda de la siguiente manera:

Para la ecuación (4.1) se tiene:

$$\begin{aligned} \frac{C_{1i}^{n+1} - C_{1i}^n}{\Delta t} + U \left(\frac{C_{1i+1}^n - C_{1i-1}^n}{2\Delta x} \right) - D \left(\frac{C_{1i+1}^n - 2C_{1i}^n + C_{1i-1}^n}{(\Delta x)^2} \right) \\ + \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n = -k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n \end{aligned}$$

Para la ecuación (4.2) se tiene:

$$\begin{aligned} \frac{C_{2i}^{n+1} - C_{2i}^n}{\Delta t} + U \left(\frac{C_{2i+1}^n - C_{2i-1}^n}{2\Delta x} \right) - D \left(\frac{C_{2i+1}^n - 2C_{2i}^n + C_{2i-1}^n}{(\Delta x)^2} \right) \\ + \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n = -k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n \end{aligned}$$

Paso 4. Presentación del modelo discreto para encontrar la solución de las ecuaciones (4.1) y (4.2).

Para la ecuación (4.1) se despeja C_{1i}^{n+1} :

$$\begin{aligned} \frac{C_{1i}^{n+1} - C_{1i}^n}{\Delta t} = -U \left(\frac{C_{1i+1}^n - C_{1i-1}^n}{2\Delta x} \right) + D \left(\frac{C_{1i+1}^n - 2C_{1i}^n + C_{1i-1}^n}{(\Delta x)^2} \right) \\ - \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n - k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n \\ C_{1i}^{n+1} - C_{1i}^n = \left[-U \left(\frac{C_{1i+1}^n - C_{1i-1}^n}{2\Delta x} \right) + D \left(\frac{C_{1i+1}^n - 2C_{1i}^n + C_{1i-1}^n}{(\Delta x)^2} \right) \right. \\ \left. - \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n - k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n \right] \Delta t \\ C_{1i}^{n+1} = \left[-U \left(\frac{C_{1i+1}^n - C_{1i-1}^n}{2\Delta x} \right) + D \left(\frac{C_{1i+1}^n - 2C_{1i}^n + C_{1i-1}^n}{(\Delta x)^2} \right) - \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n \right. \\ \left. - k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n \right] \Delta t + C_{1i}^n \end{aligned} \tag{A1.1}$$

Para la ecuación (4.2) se despeja C_{2i}^{n+1} :

$$\begin{aligned} \frac{C_{2i}^{n+1} - C_{2i}^n}{\Delta t} + U \left(\frac{C_{2i+1}^n - C_{2i-1}^n}{2\Delta x} \right) - D \left(\frac{C_{2i+1}^n - 2C_{2i}^n + C_{2i-1}^n}{(\Delta x)^2} \right) \\ + \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n = -k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n \\ C_{2i}^{n+1} - C_{2i}^n = \left[-U \left(\frac{C_{2i+1}^n - C_{2i-1}^n}{2\Delta x} \right) + D \left(\frac{C_{2i+1}^n - 2C_{2i}^n + C_{2i-1}^n}{(\Delta x)^2} \right) \right. \\ \left. - \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n - k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n \right] \Delta t \\ C_{2i}^{n+1} - C_{2i}^n = \left[-U \left(\frac{C_{2i+1}^n - C_{2i-1}^n}{2\Delta x} \right) + D \left(\frac{C_{2i+1}^n - 2C_{2i}^n + C_{2i-1}^n}{(\Delta x)^2} \right) \right. \\ \left. - \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n - k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n \right] \Delta t \\ C_{2i}^{n+1} = \left[-U \left(\frac{C_{2i+1}^n - C_{2i-1}^n}{2\Delta x} \right) + D \left(\frac{C_{2i+1}^n - 2C_{2i}^n + C_{2i-1}^n}{(\Delta x)^2} \right) \right. \\ \left. - \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i}^n} \right) \delta_2 C_{2i}^n - k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i}^n} \right) \delta_1 C_{1i}^n \right] \Delta t + C_{2i}^n \end{aligned} \tag{A1.2}$$

Para el modelo de solución de (A1.1) y (A1.2) se establece que si $C_{2i,j}^n > C_{1i,j}^n$ se tendrá $\delta_1 = 1$ y $\delta_2 = 0$ mientras que cuando $C_{1i,j}^n > C_{2i,j}^n$ se tendrá $\delta_1 = 0$ y $\delta_2 = 1$.

8.2 Anexo 2. Modelo numérico para la solución bidimensional (2D) con términos de biodegradación aerobia

Los pasos para el modelo bidimensional son los siguientes:

Paso 1. Discretización de los términos de la ecuación (4.1), donde se tiene un parámetro i para los cambios en x y j para los cambios en y .

- Diferenciación temporal hacia adelante

$$\frac{\partial c_1}{\partial t} \cong \frac{C_{1,i,j}^{n+1} - C_{1,i,j}^n}{\Delta t}$$

- Diferenciación espacial central

$$\frac{\partial c_1}{\partial x} \cong \frac{C_{1,i+1,j}^n - C_{1,i-1,j}^n}{2\Delta x}$$

$$\frac{\partial c_1}{\partial y} \cong \frac{C_{1,i,j+1}^n - C_{1,i,j-1}^n}{2\Delta y}$$

- Discretización de la segunda derivada espacial

$$\frac{\partial^2 c_1}{\partial x^2} \cong \frac{C_{1,i+1,j}^n - 2C_{1,i,j}^n + C_{1,i-1,j}^n}{(\Delta x)^2}$$

$$\frac{\partial^2 c_1}{\partial y^2} \cong \frac{C_{1,i,j+1}^n - 2C_{1,i,j}^n + C_{1,i,j-1}^n}{(\Delta y)^2}$$

Paso 2. Discretización de los términos de la ecuación (4.2)

- Diferenciación temporal hacia adelante

$$\frac{\partial c_2}{\partial t} \cong \frac{C_{2,i,j}^{n+1} - C_{2,i,j}^n}{\Delta t}$$

- Diferenciación espacial central

$$\frac{\partial c_2}{\partial x} \cong \frac{C_{2,i+1,j}^n - C_{2,i-1,j}^n}{2\Delta x}$$

$$\frac{\partial c_2}{\partial y} \cong \frac{C_{2,i,j+1}^n - C_{2,i,j-1}^n}{2\Delta y}$$

- Discretización de la segunda derivada espacial

$$\frac{\partial^2 c_2}{\partial x^2} \cong \frac{C_{2,i+1,j}^n - 2C_{2,i,j}^n + C_{2,i-1,j}^n}{(\Delta x)^2}$$

$$\frac{\partial^2 c_2}{\partial y^2} \cong \frac{C_{2,i,j+1}^n - 2C_{2,i,j}^n + C_{2,i,j-1}^n}{(\Delta y)^2}$$

Paso 3. Sustitución de las ecuaciones del Paso 1 y Paso 2 en las ecuaciones (4.1) y (4.2).

La sustitución queda de la siguiente manera:

Para la ecuación (4.1) se tiene:

$$\begin{aligned} & \frac{C_{1,i,j}^{n+1} - C_{1,i,j}^n}{\Delta t} + Ux \frac{C_{1,i+1,j}^n - C_{1,i-1,j}^n}{2\Delta x} + Uy \frac{C_{1,i,j+1}^n - C_{1,i,j-1}^n}{2\Delta y} \\ & - D \frac{C_{1,i+1,j}^n - 2C_{1,i,j}^n + C_{1,i-1,j}^n}{(\Delta x)^2} - D \frac{C_{1,i,j+1}^n - 2C_{1,i,j}^n + C_{1,i,j-1}^n}{(\Delta y)^2} \\ & + \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j}^n} \right) \delta_1 C_{1,i,j}^n = -k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j}^n} \right) \delta_2 C_{2,i,j}^n \end{aligned}$$

Para la ecuación (4.2) se tiene:

$$\begin{aligned} & \frac{C_{2,i,j}^{n+1} - C_{2,i,j}^n}{\Delta t} + Ux \frac{C_{2,i+1,j}^n - C_{2,i-1,j}^n}{2\Delta x} + Uy \frac{C_{2,i,j+1}^n - C_{2,i,j-1}^n}{2\Delta y} \\ & - D \frac{C_{2,i+1,j}^n - 2C_{2,i,j}^n + C_{2,i-1,j}^n}{(\Delta x)^2} - D \frac{C_{2,i,j+1}^n - 2C_{2,i,j}^n + C_{2,i,j-1}^n}{(\Delta y)^2} \\ & + \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j}^n} \right) \delta_2 C_{2,i,j}^n = -k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j}^n} \right) \delta_1 C_{1,i,j}^n \end{aligned}$$

Paso 4. Presentación del modelo discreto para la solución de las ecuaciones (4.1) y (4.2)

Para la ecuación (4.1) se despeja $C_{1,i,j}^{n+1}$:

$$\begin{aligned} & \frac{C_{1,i,j}^{n+1} - C_{1,i,j}^n}{\Delta t} + Ux \frac{C_{1,i+1,j}^n - C_{1,i-1,j}^n}{2\Delta x} + Uy \frac{C_{1,i,j+1}^n - C_{1,i,j-1}^n}{2\Delta y} \\ & - D \frac{C_{1,i+1,j}^n - 2C_{1,i,j}^n + C_{1,i-1,j}^n}{(\Delta x)^2} - D \frac{C_{1,i,j+1}^n - 2C_{1,i,j}^n + C_{1,i,j-1}^n}{(\Delta y)^2} \\ & + \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j}^n} \right) \delta_1 C_{1,i,j}^n = -k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j}^n} \right) \delta_2 C_{2,i,j}^n \end{aligned}$$

$$\begin{aligned} & \frac{C_{1,i,j}^{n+1} - C_{1,i,j}^n}{\Delta t} \\ & = -Ux \frac{C_{1,i+1,j}^n - C_{1,i-1,j}^n}{2\Delta x} - Uy \frac{C_{1,i,j+1}^n - C_{1,i,j-1}^n}{2\Delta y} \\ & + D \frac{C_{1,i+1,j}^n - 2C_{1,i,j}^n + C_{1,i-1,j}^n}{(\Delta x)^2} + D \frac{C_{1,i,j+1}^n - 2C_{1,i,j}^n + C_{1,i,j-1}^n}{(\Delta y)^2} \\ & - \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j}^n} \right) \delta_1 C_{1,i,j}^n - k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j}^n} \right) \delta_2 C_{2,i,j}^n \end{aligned}$$

$$\begin{aligned} C_{1,i,j}^{n+1} = & \left[-Ux \frac{C_{1,i+1,j}^n - C_{1,i-1,j}^n}{2\Delta x} - Uy \frac{C_{1,i,j+1}^n - C_{1,i,j-1}^n}{2\Delta y} \right. \\ & + D \frac{C_{1,i+1,j}^n - 2C_{1,i,j}^n + C_{1,i-1,j}^n}{(\Delta x)^2} + D \frac{C_{1,i,j+1}^n - 2C_{1,i,j}^n + C_{1,i,j-1}^n}{(\Delta y)^2} \\ & \left. - \left(\frac{V_m^1 X_1}{K_h^1 + C_{1,i,j}^n} \right) \delta_1 C_{1,i,j}^n - k_{12} \left(\frac{V_m^2 X_1}{K_h^2 + C_{2,i,j}^n} \right) \delta_2 C_{2,i,j}^n \right] \Delta t + C_{1,i,j}^n \end{aligned}$$

(A2.1)

Para la ecuación (4.2) se despeja $C_{2i,j}^{n+1}$:

$$\begin{aligned}
& \frac{C_{2i,j}^{n+1} - C_{2i,j}^n}{\Delta t} + Ux \frac{C_{2i+1,j}^n - C_{2i-1,j}^n}{2\Delta x} + Uy \frac{C_{2i,j+1}^n - C_{2i,j-1}^n}{2\Delta y} \\
& - D \frac{C_{2i+1,j}^n - 2C_{2i,j}^n + C_{2i-1,j}^n}{(\Delta x)^2} - D \frac{C_{2i,j+1}^n - 2C_{2i,j}^n + C_{2i,j-1}^n}{(\Delta y)^2} \\
& + \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i,j}^n} \right) \delta_2 C_{2i,j}^n = -k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i,j}^n} \right) \delta_1 C_{1i,j}^n \\
& \frac{C_{2i,j}^{n+1} - C_{2i,j}^n}{\Delta t} \\
& = -Ux \frac{C_{2i+1,j}^n - C_{2i-1,j}^n}{2\Delta x} - Uy \frac{C_{2i,j+1}^n - C_{2i,j-1}^n}{2\Delta y} \\
& + D \frac{C_{2i+1,j}^n - 2C_{2i,j}^n + C_{2i-1,j}^n}{(\Delta x)^2} + D \frac{C_{2i,j+1}^n - 2C_{2i,j}^n + C_{2i,j-1}^n}{(\Delta y)^2} \\
& - \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i,j}^n} \right) \delta_2 C_{2i,j}^n - k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i,j}^n} \right) \delta_1 C_{1i,j}^n \\
C_{2i,j}^{n+1} = & \left[-Ux \frac{C_{2i+1,j}^n - C_{2i-1,j}^n}{2\Delta x} - Uy \frac{C_{2i,j+1}^n - C_{2i,j-1}^n}{2\Delta y} \right. \\
& + D \frac{C_{2i+1,j}^n - 2C_{2i,j}^n + C_{2i-1,j}^n}{(\Delta x)^2} + D \frac{C_{2i,j+1}^n - 2C_{2i,j}^n + C_{2i,j-1}^n}{(\Delta y)^2} \\
& \left. - \left(\frac{V_m^2 X_1}{K_h^2 + C_{2i,j}^n} \right) \delta_2 C_{2i,j}^n - k_{21} \left(\frac{V_m^1 X_1}{K_h^1 + C_{1i,j}^n} \right) \delta_1 C_{1i,j}^n \right] \Delta t + C_{2i,j}^n
\end{aligned} \tag{A2.1}$$

Para el modelo de solución de las ecuaciones (A2.1) y (A2.2) se establece que si $C_{2i,j}^n > C_{1i,j}^n$ se tendrá $\delta_1 = 1$ y $\delta_2 = 0$ mientras que cuando $C_{1i,j,k}^n > C_{2i,j,k}^n$ se tendrá $\delta_1 = 0$ y $\delta_2 = 1$.

8.3 Anexo 3. Directorio de archivos OpenFOAM para la simulación hidrodinámica de las lagunas de estabilización

Laguna 1, para condición inicial $t = 0$ s (carpeta: 0)

- Archivo k

```
/*-----* C++ *-----*\
|=====|                                     |
|\ / F i e l d   | OpenFOAM: The Open Source CFD Toolbox |
|\ / O p e r a t i o n   | Version: v1812                 |
|\ / A n d       | Web:   www.OpenFOAM.com               |
|\ \ M a n i p u l a t i o n   |                             |
\*-----*/
FoamFile
{
  version 2.0;
  format  ascii;
  class  volScalarField;
  object  k;
}
// ***** //

dimensions  [0 2 -2 0 0 0];

internalField  uniform 0;

boundaryField
{
  inlet
  {
    type      fixedValue;
    value     uniform 2e-05;
  }

  outlet
  {
    type      inletOutlet;
    inletValue  uniform 0;
    value     uniform 0;
  }

  wall
  {
    type      fixedValue;
    value     uniform 0;
  }

  SLA
  {

```

```

        type      inletOutlet;
        inletValue  uniform 0;
        value      uniform 0;
    }
}

// ***** //

```

- Archivo nut

```

/*-----* C++ *-----*/
|=====|
| \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ / O p e r a t i o n | Version: v1812 |
| \ / A n d | Web: www.OpenFOAM.com |
| \ \ M a n i p u l a t i o n |
/*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object nut;
}
// ***** //

```

```

dimensions [0 2 -1 0 0 0];

```

```

internalField uniform 0;

```

```

boundaryField
{
    inlet
    {
        type zeroGradient;
    }

    outlet
    {
        type zeroGradient;
    }

    wall
    {
        type zeroGradient;
    }

    SLA
    {
        type calculated;
        value uniform 0;
    }
}

```

```
// ***** //
```

- Archivo nuTilda

```
/*-----* C++ *-----*\
|=====|
|\ / Field | OpenFOAM: The Open Source CFD Toolbox |
|\ / Operation | Version: v1812 |
|\ / And | Web: www.OpenFOAM.com |
|\ \ Manipulation |
\*-----*/
```

```
FoamFile
{
  version 2.0;
  format ascii;
  class volScalarField;
  object nuTilda;
}
// ***** //
```

```
dimensions [0 2 -1 0 0 0];
```

```
internalField uniform 0;
```

```
boundaryField
```

```
{
  inlet
  {
    type fixedValue;
    value uniform 0;
  }

  outlet
  {
    type inletOutlet;
    inletValue uniform 0;
    value uniform 0;
  }

  wall
  {
    type fixedValue;
    value uniform 0;
  }

  SLA
  {
    type inletOutlet;
    inletValue uniform 0;
    value uniform 0;
  }
}
```

```
// ***** //
```

- Archivo p

```
/*-----* C++ *-----*\
|=====|
|\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
|\ / O p e r a t i o n | Version: v1812 |
|\ / A n d | Web: www.OpenFOAM.com |
|\ \ M a n i p u l a t i o n |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object p;
}
// ***** //

dimensions [0 2 -2 0 0 0];

internalField uniform 0;

boundaryField
{
    inlet
    {
        type zeroGradient;
    }

    outlet
    {
        type fixedValue;
        value uniform 0;
    }

    wall
    {
        type zeroGradient;
    }

    SLA
    {
        type totalPressure;
        p0 uniform 0;
    }
}
// ***** //
```

- Archivos

```

/*-----*- C++ -*-----*\
|=====|
|\ / Field | OpenFOAM: The Open Source CFD Toolbox |
|\ / Operation | Version: v1812 |
|\ / And | Web: www.OpenFOAM.com |
|\ \ Manipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object s;
}
// ***** //

dimensions [0 0 0 0 0 0];

internalField uniform 0;

boundaryField
{
    inlet
    {
        type fixedValue;
        value uniform 1;
    }

    outlet
    {
        type inletOutlet;
        inletValue uniform 0;
        value uniform 0;
    }

    wall
    {
        type zeroGradient;
    }

    SLA
    {
        type zeroGradient;
    }
}

// ***** //

```

- Archivo U

```

/*-----* C++ *-----*\
|=====|
|\ / Field | OpenFOAM: The Open Source CFD Toolbox |
|\ / Operation | Version: v1812 |
|\ / And | Web: www.OpenFOAM.com |
|\ \ Manipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class volVectorField;
    object U;
}
// ***** //

dimensions [0 1 -1 0 0 0];

internalField uniform (0.000164985 0 0);

boundaryField
{
    inlet
    {
        type turbulentInlet;
        referenceField uniform (0.2132 0 0);
        fluctuationScale (0.02 0.01 0.01);
        value uniform (0.2132 0 0);
    }

    outlet
    {
        type turbulentInlet;
        referenceField uniform (0.2132 0 0);
        fluctuationScale (0.02 0.01 0.01);
        value uniform (0.2132 0 0);
        //type inletOutlet;
        //inletValue uniform (0.2132 0 0);
        //value uniform (0.2132 0 0);
    }

    wall
    {
        type noSlip;
    }

    SLA
    {
        type pressureInletOutletVelocity;
        value uniform (0 0 0);
    }
}
// ***** //

```


Laguna 1, datos del sistema (carpeta: system)

- Archivo ControlDict

```
/*-----* C++ *-----*\
|=====|
|\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
|\ / O peration | Version: v1812 |
|\ / A nd | Web: www.OpenFOAM.com |
|\ \ M anipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object controlDict;
}
// ***** //

application pisoFoam;

startFrom startTime;

startTime 0;

stopAt endTime;

endTime 7200;

deltaT 0.5;

writeControl timeStep;

writeInterval 2400;

purgeWrite 0;

writeFormat ascii;

writePrecision 6;

writeCompression off;

timeFormat general;

timePrecision 6;

runTimeModifiable true;

functions
{
```

```

probes
{
    type        probes;
    libs        ("libsampling.so");
    writeControl  timeStep;
    writeInterval 1;

    fields
    (
        p
    );

    probeLocations
    (
        (0.0254 0.0253 0)
        (0.0508 0.0253 0)
        (0.0762 0.0253 0)
        (0.1016 0.0253 0)
        (0.127 0.0253 0)
        (0.1524 0.0253 0)
        (0.1778 0.0253 0)
    );
}

fieldAverage1
{
    type        fieldAverage;
    libs        ("libfieldFunctionObjects.so");
    writeControl  writeTime;

    fields
    (
        U
        {
            mean    on;
            prime2Mean on;
            base    time;
        }

        p
        {
            mean    on;
            prime2Mean on;
            base    time;
        }
    );
}

surfaceSampling
{
    // Sample near-wall velocity

    type surfaces;

    // Where to load it from (if not already in solver)

```

```

libs      ("libsampling.so");
writeControl  writeTime;

interpolationScheme cellPoint;

surfaceFormat vtk;

// Fields to be sampled
fields
(
    U
);

surfaces
(
    nearWall
    {
        type      patchInternalField;
        patches    ( lowerWall );
        distance    1E-6;
        interpolate true;
        triangulate false;
    }
);

#includeFunc scalarTransport
}

// ***** //

```

- Archivo fvSchemes

```

/*-----* C++ *-----*\
|=====|
|\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
|\ / O peration | Version: v1812 |
|\ / A nd | Web: www.OpenFOAM.com |
|\ \ M anipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object fvSchemes;
}
// ***** //

ddtSchemes
{
    default backward;
}

```

```

gradSchemes
{
  default      Gauss linear;
}

divSchemes
{
  default      none;
  div(phi,U)   Gauss LUST grad(U);
  div(phi,k)   Gauss limitedLinear 1;
  div(phi,s)   bounded Gauss limitedLinear 1;
  div((nuEff*dev2(T(grad(U)))) Gauss linear;
}

laplacianSchemes
{
  default      Gauss linear corrected;
}

interpolationSchemes
{
  default      linear;
}

snGradSchemes
{
  default      corrected;
}

// ***** //

```

- Archivo fvSolution

```

/*-----* C++ -*-----*\
|=====|
|\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
|\ / O peration | Version: v1812 |
|\ / A nd | Web: www.OpenFOAM.com |
|\ \ M anipulation |
\*-----*/
FoamFile
{
  version 2.0;
  format ascii;
  class dictionary;
  location "system";
  object fvSolution;
}

```

```

// ***** //

solvers
{
  p
  {
    solver      GAMG;
    tolerance   1e-06;
    relTol      0.1;
    smoother    GaussSeidel;
  }

  pFinal
  {
    $p;
    smoother    DICGaussSeidel;
    tolerance   1e-06;
    relTol      0;
  }

  "(U|k|B|nuTilda|s)"
  {
    solver      smoothSolver;
    smoother    GaussSeidel;
    tolerance   1e-05;
    relTol      0;
  }
}

PISO
{
  nCorrectors 2;
  nNonOrthogonalCorrectors 0;
}

// ***** //

```

Laguna 1, datos de constantes (carpeta: constant)

- Archivo transportProperties

```

/*----- C++ -----*\
|=====|
|\  / F ield   | OpenFOAM: The Open Source CFD Toolbox |
|\  / O peration | Version: v1812 |
|\  / A nd      | Web: www.OpenFOAM.com |
|\  \ M anipulation |
\*-----*/
FoamFile
{
  version 2.0;

```

```

format    ascii;
class    dictionary;
location "constant";
object   transportProperties;
}
// ***** //

transportModel Newtonian;

nu        1e-06;

// ***** //

```

- Archivo turbulenceProperties

```

/*-----* C++ *-----*\
|=====| | |
|\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
|\ / O p e r a t i o n | Version: v1812 |
|\ / A n d | Web: www.OpenFOAM.com |
|\ \ M a n i p u l a t i o n | |
\*-----*/
FoamFile
{
    version 2.0;
    format  ascii;
    class   dictionary;
    location "constant";
    object  turbulenceProperties;
}
// ***** //

simulationType LES;

LES
{
    LESModel    dynamicKEqn;

    turbulence  on;

    printCoeffs on;

    delta       cubeRootVol;

    dynamicKEqnCoeffs
    {
        filter simple;
    }

    cubeRootVolCoeffs
    {
        deltaCoeff 1;
    }
}

```

```

}

PrandtlCoeffs
{
  delta      cubeRootVol;
  cubeRootVolCoeffs
  {
    deltaCoeff  1;
  }

  smoothCoeffs
  {
    delta      cubeRootVol;
    cubeRootVolCoeffs
    {
      deltaCoeff  1;
    }

    maxDeltaRatio  1.1;
  }

  Cdelta      0.158;
}

vanDriestCoeffs
{
  delta      cubeRootVol;
  cubeRootVolCoeffs
  {
    deltaCoeff  1;
  }

  smoothCoeffs
  {
    delta      cubeRootVol;
    cubeRootVolCoeffs
    {
      deltaCoeff  1;
    }

    maxDeltaRatio  1.1;
  }

  Aplus      26;
  Cdelta      0.158;
}

smoothCoeffs
{
  delta      cubeRootVol;
  cubeRootVolCoeffs
  {
    deltaCoeff  1;
  }

  maxDeltaRatio  1.1;
}

```

```
}  
}
```

```
// ***** //
```

Laguna 2 y 3, para la condición inicial $t = 0$ s (carpeta: 0)

- Archivo k

```
/*-----*- C++ -*-----*\n|=====\n|\\ / Field | OpenFOAM: The Open Source CFD Toolbox | \n|\\ / Operation | Version: v1812 | \n|\\ / And | Web: www.OpenFOAM.com | \n|\\ Manipulation | \n|-----*\nFoamFile\n{\n  version 2.0;\n  format ascii;\n  class volScalarField;\n  object k;\n}\n// ***** //\n\ndimensions [0 2 -2 0 0 0];\n\ninternalField uniform 0;\n\nboundaryField\n{\n  inlet\n  {\n    type fixedValue;\n    value uniform 2e-05;\n  }\n\n  outlet\n  {\n    type inletOutlet;\n    inletValue uniform 0;\n    value uniform 0;\n  }\n\n  wall\n  {\n    type fixedValue;\n    value uniform 0;\n  }\n\n  SLA
```



```

    {
        type      inletOutlet;
        inletValue  uniform 0;
        value      uniform 0;
    }
}

// ***** //

```

- Archivo nut

```

/*----- C++ -----*\
|=====|
|\ / Field | OpenFOAM: The Open Source CFD Toolbox |
|\ / Operation | Version: v1812 |
|\ / And | Web: www.OpenFOAM.com |
|\ / Manipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object nut;
}
// ***** //

dimensions [0 2 -1 0 0 0];

internalField uniform 0;

boundaryField
{
    inlet
    {
        type zeroGradient;
    }

    outlet
    {
        type zeroGradient;
    }

    wall
    {
        type zeroGradient;
    }

    SLA
    {
        type calculated;
        value uniform 0;
    }
}

```

```
}
```

```
// ***** //
```

- Archivo nuTilda

```
/*----- C++ -----*\
|=====|
| \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ / O p e r a t i o n | Version: v1812 |
| \ / A n d | Web: www.OpenFOAM.com |
| \ / M a n i p u l a t i o n |
|-----*/
FoamFile
{
  version 2.0;
  format ascii;
  class volScalarField;
  object nuTilda;
}
// ***** //

dimensions [0 2 -1 0 0 0];

internalField uniform 0;

boundaryField
{
  inlet
  {
    type fixedValue;
    value uniform 0;
  }

  outlet
  {
    type inletOutlet;
    inletValue uniform 0;
    value uniform 0;
  }

  wall
  {
    type fixedValue;
    value uniform 0;
  }

  SLA
  {
    type inletOutlet;
    inletValue uniform 0;
    value uniform 0;
  }
}
```

```
}
```

```
// ***** //
```

- Archivo p

```
/*----- C++ -----*\
|=====|
| \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ / O p e r a t i o n | Version: v1812 |
| \ / A n d | Web: www.OpenFOAM.com |
| \ / M a n i p u l a t i o n |
|-----*/
```

```
FoamFile
```

```
{
  version 2.0;
  format  ascii;
  class   volScalarField;
  object  p;
}
```

```
// ***** //
```

```
dimensions [0 2 -2 0 0 0];
```

```
internalField uniform 0;
```

```
boundaryField
```

```
{
  inlet
  {
    type zeroGradient;
  }

```

```
  outlet
  {
    type    fixedValue;
    value   uniform 0;
  }

```

```
  wall
  {
    type zeroGradient;
  }

```

```
  SLA
  {
    type    totalPressure;
    p0     uniform 0;
  }

```

```
}
```

```
// ***** //
```

- Archivos

```

/*-----*- C++ -*-----*\
|=====|
|\ / Field | OpenFOAM: The Open Source CFD Toolbox |
|\ / Operation | Version: v1812 |
|\ / And | Web: www.OpenFOAM.com |
|\ \ Manipulation |
\*-----*/
FoamFile
{
  version 2.0;
  format ascii;
  class volScalarField;
  object s;
}
// ***** //

dimensions [0 0 0 0 0 0];

internalField uniform 0;

boundaryField
{
  inlet
  {
    type fixedValue;
    value uniform 1;
  }

  outlet
  {
    type inletOutlet;
    inletValue uniform 0;
    value uniform 0;
  }

  wall
  {
    type zeroGradient;
  }

  SLA
  {
    type zeroGradient;
  }
}

// ***** //

```

- Archivo U

```

/*-----* C++ *-----*\
|=====| | |
|\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
|\ / O p e r a t i o n | Version: v1812 |
|\ / A n d | Web: www.OpenFOAM.com |
|\ \ M a n i p u l a t i o n | |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class volVectorField;
    object U;
}
// ***** //

dimensions [0 1 -1 0 0 0];

internalField uniform (0.000166429 0 0);

boundaryField
{
    inlet
    {
        type turbulentInlet;
        referenceField uniform (0.2132 0 0);
        fluctuationScale (0.02 0.01 0.01);
        value uniform (0.2132 0 0);
    }

    outlet
    {
        type turbulentInlet;
        referenceField uniform (0.2132 0 0);
        fluctuationScale (0.02 0.01 0.01);
        value uniform (0.2132 0 0);
        //type inletOutlet;
        //inletValue uniform (0.2132 0 0);
        //value uniform (0.2132 0 0);
    }

    wall
    {
        type noSlip;
    }

    SLA
    {
        type pressureInletOutletVelocity;
        value uniform (0 0 0);
    }
}

```

```
}
```

```
// ***** //
```

Laguna 2 y 3, datos del sistema (carpeta: system)

- Archivo ControlDict

```
/*-----*- C++ -*-----*\
|=====|
| \ / Field | OpenFOAM: The Open Source CFD Toolbox |
| \ / Operation | Version: v1812 |
| \ / And | Web: www.OpenFOAM.com |
| \ \ Manipulation |
\*-----*/
FoamFile
{
  version 2.0;
  format ascii;
  class dictionary;
  location "system";
  object controlDict;
}
// ***** //

application pisoFoam;

startFrom startTime;

startTime 0;

stopAt endTime;

endTime 21600;

deltaT 0.5;

writeControl timeStep;

writeInterval 2400;

purgeWrite 0;

writeFormat ascii;

writePrecision 6;

writeCompression off;

timeFormat general;

timePrecision 6;
```

```

runTimeModifiable true;

functions
{
  probes
  {
    type      probes;
    libs      ("libsampling.so");
    writeControl  timeStep;
    writeInterval  1;

    fields
    (
      p
    );

    probeLocations
    (
      (0.0254 0.0253 0)
      (0.0508 0.0253 0)
      (0.0762 0.0253 0)
      (0.1016 0.0253 0)
      (0.127 0.0253 0)
      (0.1524 0.0253 0)
      (0.1778 0.0253 0)
    );
  }
}

fieldAverage1
{
  type      fieldAverage;
  libs      ("libfieldFunctionObjects.so");
  writeControl  writeTime;

  fields
  (
    U
    {
      mean      on;
      prime2Mean on;
      base      time;
    }

    p
    {
      mean      on;
      prime2Mean on;
      base      time;
    }
  );
}

surfaceSampling
{

```

```

// Sample near-wall velocity

type surfaces;

// Where to load it from (if not already in solver)
libs      ("libsampling.so");
writeControl  writeTime;

interpolationScheme cellPoint;

surfaceFormat vtk;

// Fields to be sampled
fields
(
    U
);

surfaces
(
    nearWall
    {
        type      patchInternalField;
        patches    ( lowerWall );
        distance    1E-6;
        interpolate true;
        triangulate false;
    }
);

#includeFunc scalarTransport
}

// ***** //

```

- Archivo fvSchemes

```

/*----- C++ -----*\
|=====|
| \ / Field | OpenFOAM: The Open Source CFD Toolbox |
| \ / Operation | Version: v1812 |
| \ / And | Web: www.OpenFOAM.com |
| \ / Manipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object fvSchemes;
}
// ***** //

```



```

ddtSchemes
{
  default    backward;
}

gradSchemes
{
  default    Gauss linear;
}

divSchemes
{
  default    none;
  div(phi,U) Gauss LUST grad(U);
  div(phi,k) Gauss limitedLinear 1;
  div(phi,s) bounded Gauss limitedLinear 1;
  div((nuEff*dev2(T(grad(U)))) Gauss linear;
}

laplacianSchemes
{
  default    Gauss linear corrected;
}

interpolationSchemes
{
  default    linear;
}

snGradSchemes
{
  default    corrected;
}

// ***** //

```

- Archivo fvSolution

```

/*-----* C++ -*-----*\
|=====| |
|\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
|\ / O p e r a t i o n | Version: v1812 |
|\ / A n d | Web: www.OpenFOAM.com |
|\ \ M a n i p u l a t i o n | |
\*-----*/
FoamFile
{
  version 2.0;
  format ascii;
  class dictionary;
  location "system";
  object fvSolution;
}
// ***** //

solvers
{
  p
  {
    solver GAMG;
    tolerance 1e-06;
    relTol 0.1;
    smoother GaussSeidel;
  }

  pFinal
  {
    $p;
    smoother DICGaussSeidel;
    tolerance 1e-06;
    relTol 0;
  }

  "(U|k|B|nuTilda)s"
  {
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-05;
    relTol 0;
  }
}

PISO
{
  nCorrectors 2;
  nNonOrthogonalCorrectors 0;
}

// ***** //

```

Laguna 2 y 3, datos de constantes (carpeta: constant)

- Archivo transportProperties

```
/*----- C++ -----*\
|=====|
| \ / Field | OpenFOAM: The Open Source CFD Toolbox |
| \ / Operation | Version: v1812 |
| \ / And | Web: www.OpenFOAM.com |
| \ \ Manipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "constant";
    object transportProperties;
}
// ***** //

transportModel Newtonian;

nu 1e-06;

// ***** //
```

- Archivo turbulenceProperties

```
/*----- C++ -----*\
|=====|
| \ / Field | OpenFOAM: The Open Source CFD Toolbox |
| \ / Operation | Version: v1812 |
| \ / And | Web: www.OpenFOAM.com |
| \ \ Manipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "constant";
    object turbulenceProperties;
}
// ***** //

simulationType LES;

LES
```

```

{
  LESModel    dynamicKEqn;

  turbulence  on;

  printCoeffs on;

  delta      cubeRootVol;

  dynamicKEqnCoeffs
  {
    filter simple;
  }

  cubeRootVolCoeffs
  {
    deltaCoeff  1;
  }

  PrandtlCoeffs
  {
    delta      cubeRootVol;
    cubeRootVolCoeffs
    {
      deltaCoeff  1;
    }

    smoothCoeffs
    {
      delta      cubeRootVol;
      cubeRootVolCoeffs
      {
        deltaCoeff  1;
      }

      maxDeltaRatio 1.1;
    }

    Cdelta     0.158;
  }

  vanDriestCoeffs
  {
    delta      cubeRootVol;
    cubeRootVolCoeffs
    {
      deltaCoeff  1;
    }

    smoothCoeffs
    {
      delta      cubeRootVol;
      cubeRootVolCoeffs
      {
        deltaCoeff  1;
      }
    }
  }
}

```

```
    maxDeltaRatio 1.1;
}

Aplus      26;
Cdelta     0.158;
}

smoothCoeffs
{
    delta      cubeRootVol;
    cubeRootVolCoeffs
    {
        deltaCoeff  1;
    }

    maxDeltaRatio 1.1;
}
}

// ***** //
```

8.4 Anexo 4. Código de programación en Python para la solución de las ecuaciones de advección, difusión y reacción

```
#Ecuacion de adveccion, difusion y reaccion
#Elaborado por Benigno Duran Aguilar
#Fecha: 01/06/2019

#modulos

import os
import matplotlib
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt
from matplotlib.colors import BoundaryNorm
from matplotlib.ticker import MaxNLocator
import pandas as pd
import math
from numpy import genfromtxt

os.system('cls')
os.system('clear')

print ("Elija una opcion")
print ("1: Verificacion del modelo numerico con terminos de biodegradacion
aerobia")
print ("2: Ejemplo de aplicacion 2D, transporte de un contaminante conservativo
Solucion analitica")
print ("3: Ejemplo de aplicacion 2D, transporte de un contaminante conservativo
Solucion numerica")
print ("4: Ejemplo de aplicacion 2D, transporte de dos especies Solucion
numerica")
print ("5: Ejemplo de aplicacion 3D, transporte de dos especies Solucion numerica
Laguna 1, 2 y 3")
print ("6: Salir")

numero = int(input("Introduce un numero: "))

if numero == 1:

    #Datos del problema
    Vm = 1    #dias^-1 para i=1,2
    Kh = 0.1  #mg/l para i=1,2
    k12 = 2.0 #constante
    k21 = 0.5 #constante
```

```

#Especie estacionaria
X1 = 0.2 #mg/l en lugar de 0.2

#Tiempo de simulacion
t = 68 #days en el problema es 68 days

#Incremento espacial y temporal
X = 100 #Dominio espacial en metros en x
J = 100 #Contador en x

Deltax = X/J #Incremento espacial en x, en metros
Deltat = 0.1 #Incremento temporal

#Simulacion
#Condicion inicial y de frontera
Ze = np.zeros(1)
C1 = Ze+3 #mg/l oxigeno
C2 = Ze+10 #mg/l carbono
C1F = Ze+3 #mg/l oxigeno
C2F = Ze+0 #mg/l carbono

#Datos de Flujo
Dx = 0.2 #m^2/dia
Dy = 0.2 #m^2/dia
Dz = 0.2 #m^2/dia

#Matriz de velocidades
#Umax =
pd.read_csv("/Users/benignoduranaguilar/Documents/docker/Tesis/2D/Up149/Uxp
149.csv").values
#Umx = Uma.transpose() #agregar Ux[0,j+1] en la ecuacion
Umx = 1

#Numero de courant y Peclet
Cr = Umx*Deltat/Deltax #Numero de Courant
Pe = Umx*Deltax/Dx #Numero de Peclet
print("Cr max =", Cr)
print("Pe max =", Pe)

#Escalar 1 c1 Oxigeno
Co1 = np.zeros(J+1) + 3

#Escalar 2 c2 Sustrato
#Co2 = c
Co2 = np.zeros(J+1)
Co2[0] = C2

```

```

Cm1 = np.zeros(J-1)
Cm2 = np.zeros(J-1)
t_sim = t / Deltat

#Matriz para guardar la simulacion en cada delta t
Co1s = np.zeros((int(t_sim+1),J+1))
Co2s = np.zeros((int(t_sim+1),J+1))
Co1s[0] = Co1
Co2s[0] = Co2

xx = np.linspace(0, J-2, J-1)

k=0
while k < t_sim:
    k = k+1
    print("t simulacion (dias) = ",round(k*Deltat,4))
    for i in xx:
        j = int(i)
        if Co2[j+1] > Co1[j+1]:
            delta1=1
            delta2=0
        elif Co1[j+1] > Co2[j+1]:
            delta1=0
            delta2=1

        Ad1 = -Umx*((Co1[j+2]-Co1[j])/(2*Deltax))
        Di1 = Dx*((Co1[j+2]-2*Co1[j+1]+Co1[j])/(Deltax)**2)
        Re1 = -k12*((Vm*X1)/(Kh+Co2[j+1]))*delta2*Co2[j+1]-
        ((Vm*X1)/(Kh+Co1[j+1]))*delta1*Co1[j+1]
        Cm1[j] = (Ad1+Di1+Re1)*Deltat+Co1[j+1]

        Cont1a = np.append(C1,Cm1)
        Cont1 = np.append(Cont1a,C1F)

        Ad2 = -Umx*((Co2[j+2]-Co2[j])/(2*Deltax))
        Di2 = Dx*((Co2[j+2]-2*Co2[j+1]+Co2[j])/(Deltax)**2)
        Re2 = -k21*((Vm*X1)/(Kh+Co1[j+1]))*delta1*Co1[j+1]-
        ((Vm*X1)/(Kh+Co2[j+1]))*delta2*Co2[j+1]
        Cm2[j] = (Ad2+Di2+Re2)*Deltat+Co2[j+1]

        Cont2a = np.append(C2,Cm2)
        Cont2 = np.append(Cont2a,C2F)

    Co1 = Cont1
    Co2 = Cont2
    Co1s[k] = Co1
    Co2s[k] = Co2

```



```

# animacion de grafico
x = np.arange(0, X+1, 1) #matriz de dominio espacial
plt.clf()
plt.plot(x, Co1, marker='+', linestyle='-', color='b', lw=0.3)
plt.plot(x, Co2, marker='.', linestyle='-', color='r', lw=0.3)
plt.xlabel('Distancia (m)')
plt.ylabel('Concentraci3n (mg/L)')
plt.legend(('c1','c2'), loc='upper right')
plt.grid(True)
plt.draw()
plt.pause(0.00001)
plt.show()

```

elif numero == 2:

```

#Tiempo de simulacion
t = 5*(np.pi/10) #Una vuelta completa

#Incremento espacial y temporal
X = 1 #Dominio espacial en x
Y = 1 #Dominio espacial en y
J = 100 #Contador en x
N = 100 #Contador en y
Deltax = X/J #Incremento espacial en x
Deltay = Y/N #Incremento espacial en y
Deltat = np.pi/1000 #Incremento temporal

#Simulacion
ax = ay = np.linspace(-0.5, 0.5, J+1)
aX,aY = np.meshgrid(ax,ay)
xc = -0.25
yc = 0
Di = 0.0001 #Coeficiente de difusi3n
Desv = 0.047
for k in range(1,6):
    ti = k*(np.pi/10)

    #Escalar 1 c1
    Co1 = ((2*Desv**2)/(2*Desv**2+4*Di*ti))*np.exp(-
(((aX*math.cos(4*ti)+aY*math.sin(4*ti))-xc)**2+((-
aX*math.sin(4*ti)+aY*math.cos(4*ti))-yc)**2)/(2*Desv**2+4*Di*ti)))
    f = plt.figure(10)
    cs = plt.contour(aX,aY,Co1, 5, vmin=0, vmax=1)
    plt.axis('equal')
    plt.text(-0.30, -0.40, 't=(1/10)*\u03C0')
    plt.text(0.20, -0.30, 't=(2/10)*\u03C0')

```

```

plt.text(0.20, 0.30, 't=(3/10)*\u03C0')
plt.text(-0.15, 0.4, 't=(4/10)*\u03C0')
plt.text(-0.40, 0.15, 't=(5/10)*\u03C0')
plt.clabel(cs, inline = 10 , fontsize = 4)
plt.title('Solucion Analitica')
plt.draw()
plt.grid(True)
plt.pause(0.00001)

if k == 0:
    h0 = plt.figure(0)
    M0 = max(Co1[50,:])
    print('Valor max. (t = 0)=' ,M0)
    plt.clf()
    plt.plot(ax, Co1[50,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Solucion analitica t = 0')
    plt.grid(True)

elif k == 1:
    h1 = plt.figure(1)
    M1 = max(Co1[26,:])
    print('Valor max. (t = (1/10)*\u03C0)=' ,M1)
    plt.clf()
    plt.plot(ax, Co1[26,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Solucion analitica t = (1/10)* \u03C0')
    plt.grid(True)

elif k == 2:
    h2 = plt.figure(2)
    M2 = max(Co1[36,:])
    print('Valor max. (t = (2/10)*\u03C0)=' ,M2)
    plt.plot(ax, Co1[36,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Solucion analitica t = (2/10)* \u03C0')
    plt.grid(True)

elif k == 3:
    h3 = plt.figure(3)
    M3 = max(Co1[64,:])
    print('Valor max. (t = (3/10)*\u03C0)=' ,M3)
    plt.plot(ax, Co1[64,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')

```

```

plt.ylabel('c1')
plt.title('Solucion analitica t = (3/10)* \u03C0')
plt.grid(True)

elif k == 4:
    h4 = plt.figure(4)
    M4 = max(Co1[74,:])
    print('Valor max. (t = (4/10)*\u03C0)=',M4)
    plt.plot(ax, Co1[74:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Solucion analitica t = (4/10)* \u03C0')
    plt.grid(True)

elif k == 5:
    h5 = plt.figure(5)
    M5 = max(Co1[50,:])
    print('Valor max. (t = (5/10)*\u03C0)=',M5)
    plt.plot(ax, Co1[50:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Solucion analitica t = (5/10)* \u03C0')
    plt.grid(True)
plt.show()

```

```

elif numero == 3:

```

```

#Tiempo de simulacion
t = 5*(np.pi/10) #Una vuelta completa

#Incremento espacial y temporal
X = 1 #Dominio espacial en x
Y = 1 #Dominio espacial en y
J = 100 #Contador en x
N = 100 #Contador en y

Deltax = X/J #Incremento espacial en x, en metros
Deltay = Y/N #Incremento espacial en y, en metros
Deltat = np.pi/10000 #0.1 #Incremento temporal

#Simulacion
ax = ay = np.linspace(-0.5, 0.5, J+1)
aX,aY = np.meshgrid(ax,ay)
xc = -0.25
yc = 0
Desv = 0.047

```

```

#Escalar 1 c1
Co1 = np.exp(-(((aX-xc)**2+(aY-yc)**2)/(2*Desv**2)))
Umx = -4*aY
Umy = 4*aX

#Escalar 2 c2
c2 = np.zeros((N+1,N+1))
Co2 = c2

#Datos de Flujo
Dx = 0.0001 #m^2/dia
Dy = 0.0001 #m^2/dia

#Numero de courant y Peclet
Cr = (max(Umy[50,:])*Deltat/Deltax #Numero de Courant
Pe = (max(Umy[50,:])*Deltax/Dx #Numero de Peclet
print("Cr max =", Cr)
print("Pe max =", Pe)

Cm1 = Co1
Cm2 = Co2
t_sim = t / Deltat

m=0
while m < t_sim:
    m = m+1
    print("t simulacion = ",round(m*Deltat,4))
    for jj in range(1,N):
        j = int(jj)
        for ii in range(1,J):
            i = int(ii)
            if Co2[j,i] > Co1[j,i]:
                delta1 = 1
                delta2 = 0
            elif Co1[j,i] > Co2[j,i]:
                delta1 = 0
                delta2 = 1

            if Co1[j,i] < 0:
                Co1[j,i] = 0

            Ad1 = -Umx[j,i]*((Co1[j,i+1]-Co1[j,i-1])/(2*Deltax))-Umy[j,i]*((Co1[j+1,i]-
Co1[j-1,i])/(2*Deltay))
            Di1 = Dx*((Co1[j,i+1]-2*Co1[j,i]+Co1[j,i-1])/(Deltax)**2)+Dy*((Co1[j+1,i]-
2*Co1[j,i]+Co1[j-1,i])/(Deltay)**2)
            Re1 = 0
            Cm1[j,i] = (Ad1+Di1+Re1)*Deltat+Co1[j,i]

```

```

        Ad2 = -Umx[j,i]*((Co2[j,i+1]-Co2[j,i-1])/(2*Deltax))-Umy[j,i]*((Co2[j+1,i]-
Co2[j-1,i])/(2*Deltay))
        Di2 = Dx*((Co2[j,i+1]-2*Co2[j,i]+Co2[j,i-1])/(Deltax)**2)+Dy*((Co2[j+1,i]-
2*Co2[j,i]+Co2[j-1,i])/(Deltay)**2)
        Re2 = 0
        Cm2[j,i] = (Ad2+Di2+Re2)*Deltat+Co2[j,i]

```

```

Co1 = Cm1
Co2 = Cm2

```

```

#np.savetxt('result6.txt', Cont1a, fmt='%e')
# animacion de grafico
f = plt.figure(10)
plt.clf()
cs = plt.contour(aX,aY,abs(Co1), 6)
plt.axis('equal')
plt.clabel(cs, inline = 10 , fontsize = 4)
plt.title('Solucion modelo numerico')
plt.draw()
plt.grid(True)
plt.pause(0.00001)

if m == 0:
    h0 = plt.figure(0)
    M0 = max(Co1[50,:])
    print('\Valor max. (t = 0)=' ,M0)
    plt.plot(ax, Co1[50,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Modelo numerico t = 0')
    plt.grid(True)
    g = plt.figure(11)
    cs = plt.contour(aX,aY,abs(Co1), 6)
    plt.axis('equal')
    plt.clabel(cs, inline = 10 , fontsize = 4)
    plt.title('Solucion modelo numerico')
    plt.draw()
    plt.grid(True)

elif m == 1000:
    h1 = plt.figure(1)
    M1 = max(Co1[27,:])
    print('\Valor max. (t = (1/10)*\u03C0)=' ,M1)
    #plt.clf()
    plt.plot(ax, Co1[27,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')

```

```

plt.ylabel('c1')
plt.title('Modelo numerico t = (1/10)* \u03C0')
plt.grid(True)
g = plt.figure(11)
cs = plt.contour(aX,aY,abs(Co1), 6)
plt.axis('equal')
plt.text(-0.30, -0.40, 't=(1/10)*\u03C0')
plt.clabel(cs, inline = 10 , fontsize = 4)
plt.title('Solucion modelo numerico')
plt.draw()
plt.grid(True)

elif m == 2000:
    h2 = plt.figure(2)
    M2 = max(Co1[35,:])
    print('Valor max. (t = (2/10)*\u03C0)=',M2)
    plt.plot(ax, Co1[35,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Modelo numerico t = (2/10)* \u03C0')
    plt.grid(True)
    g = plt.figure(11)
    cs = plt.contour(aX,aY,abs(Co1), 6)
    plt.axis('equal')
    plt.text(0.20, -0.30, 't=(2/10)*\u03C0')
    plt.clabel(cs, inline = 10 , fontsize = 4)
    plt.draw()

elif m == 3000:
    h3 = plt.figure(3)
    M3 = max(Co1[64,:])
    print('Valor max. (t = (3/10)*\u03C0)=',M3)
    plt.plot(ax, Co1[64,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Modelo numerico t = (3/10)* \u03C0')
    plt.grid(True)
    f = plt.figure(11)
    cs = plt.contour(aX,aY,abs(Co1), 6)
    plt.axis('equal')
    plt.text(0.20, 0.30, 't=(3/10)*\u03C0')
    plt.clabel(cs, inline = 10 , fontsize = 4)
    plt.draw()

elif m == 4000:
    h4 = plt.figure(4)
    M4 = max(Co1[74,:])

```

```

print('Valor max. (t = (4/10)*\u03C0)=' ,M4)
plt.plot(ax, Co1[74,:], marker='+', linestyle='-', color='b', lw=0.3)
plt.xlabel('Distancia')
plt.ylabel('c1')
plt.title('Modelo numerico t = (4/10)* \u03C0')
plt.grid(True)
g = plt.figure(11)
cs = plt.contour(aX,aY,abs(Co1), 6)
plt.axis('equal')
plt.text(-0.15, 0.4, 't=(4/10)*\u03C0')
plt.clabel(cs, inline = 10 , fontsize = 4)
plt.draw()

elif m == 5000:
    h5 = plt.figure(5)
    M5 = max(Co1[52,:])
    print('Valor max. (t = (5/10)*\u03C0)=' ,M5)
    plt.plot(ax, Co1[52,:], marker='+', linestyle='-', color='b', lw=0.3)
    plt.xlabel('Distancia')
    plt.ylabel('c1')
    plt.title('Modelo numerico t = (5/10)* \u03C0')
    plt.grid(True)
    g = plt.figure(11)
    cs = plt.contour(aX,aY,abs(Co1), 6)
    plt.axis('equal')
    plt.text(-0.47, 0.14, 't=(5/10)*\u03C0')
    plt.clabel(cs, inline = 10 , fontsize = 4)
    plt.draw()
plt.show()

elif numero == 4:

#Datos del problema
Vm = 1    #dias^-1 para i=1,2
Kh = 0.1  #mg/l para i=1,2
k12 = 2.0 #constante
k21 = 0.5 #constante

#Especie estacionaria
X1 = 0.2  #mg/l

#Tiempo de simulacion
t = 150   #days en el problema es 68 days

#Incremento espacial y temporal
X = 100   #Dominio espacial en metros en x
Y = 100   #Dominio espacial en metros en y

```

```

J = 100 #Contador en x
N = 100 #Contador en y

Deltax = X/J #Incremento espacial en x, en metros
Deltay = Y/N #Incremento espacial en y, en metros
Deltat = 0.1 #Incremento temporal

#Escalar 2 c2 Carbono
ax = ay = np.linspace(0, X, J+1)
aX,aY = np.meshgrid(ax,ay)
xc = 25
yc = 50
Desv = 4.7

Co2 = np.exp(-(((aX-xc)**2+(aY-yc)**2)/(2*Desv**2)))*10

#Simulacion
#Condicion inicial y de frontera
c1 = np.zeros((J+1,J+1)) #matriz de ceros en dominio espacial

#Escalar 1 c1 Oxigeno
Co1 = c1
Co1[:,:] = 3
Co1[:,X] = 3
C1 = np.full((N-1,1), 3)
C1F = np.full((N-1,1), 3)
C1v = np.full((1,J+1), 3)

#Datos de Flujo
Dx = 0.2 #m^2/dia
Dy = 0.2 #m^2/dia
Dz = 0 #m^2/dia

#Matriz de velocidades
Umx =
pd.read_csv("/Users/benignoduranaguilar/Documents/docker/Tesis/Python3D/Uxr.
csv").values
Umy =
pd.read_csv("/Users/benignoduranaguilar/Documents/docker/Tesis/Python3D/Uyr.
csv").values

#Matriz base de simulaciÃ³n
Cm1 = Co1
Cm2 = Co2
t_sim = t / Deltat

m=0

```



```

while m < t_sim:
    m = m+1
    print("t simulacion (dias) = ",round(m*Deltat,4))
    for jj in range(1,N):
        j = int(jj)
        for ii in range(1,J):
            i = int(ii)
            if Co2[j,i] > Co1[j,i]:
                delta1 = 1
                delta2 = 0
            elif Co1[j,i] > Co2[j,i]:
                delta1 = 0
                delta2 = 1

            Ad1 = -Umx[j,i]*((Co1[j,i+1]-Co1[j,i-1])/(2*Deltax))-Umy[j,i]*((Co1[j+1,i]-
Co1[j-1,i])/(2*Deltay))
            Di1 = Dx*((Co1[j,i+1]-2*Co1[j,i]+Co1[j,i-1])/(Deltax)**2)+Dy*((Co1[j+1,i]-
2*Co1[j,i]+Co1[j-1,i])/(Deltay)**2)
            Re1 = -((Vm*X1)/(Kh+Co1[j,i]))*delta1*Co1[j,i]-
k12*((Vm*X1)/(Kh+Co2[j,i]))*delta2*Co2[j,i]
            Cm1[j,i] = (Ad1+Di1+Re1)*Deltat+Co1[j,i]

            Ad2 = -Umx[j,i]*((Co2[j,i+1]-Co2[j,i-1])/(2*Deltax))-Umy[j,i]*((Co2[j+1,i]-
Co2[j-1,i])/(2*Deltay))
            Di2 = Dx*((Co2[j,i+1]-2*Co2[j,i]+Co2[j,i-1])/(Deltax)**2)+Dy*((Co2[j+1,i]-
2*Co2[j,i]+Co2[j-1,i])/(Deltay)**2)
            Re2 = -((Vm*X1)/(Kh+Co2[j,i]))*delta2*Co2[j,i]-
k21*((Vm*X1)/(Kh+Co1[j,i]))*delta1*Co1[j,i]
            Cm2[j,i] = (Ad2+Di2+Re2)*Deltat+Co2[j,i]

            Co1 = Cm1
            Co2 = Cm2

# animacion de grafico
f = plt.figure(1)
plt.clf()
x = np.arange(0, X+1, 1);
y = np.arange(0, Y+1, 1);
XX,YY = np.meshgrid(x,y)
levels = MaxNLocator(nbins=10).tick_values(0, 3.1)
cmap = plt.get_cmap('ocean')
norm = BoundaryNorm(levels, ncolors=cmap.N, clip=True)
cs = plt.contourf(XX,YY,abs(Co1), levels=levels, cmap=cmap)
plt.colorbar(cs)
plt.title('c1 en mg/l')
plt.xlabel('x (m)')
plt.ylabel('y (m)')

```

```

plt.draw()
plt.pause(0.00001)

g = plt.figure(2)
plt.clf()
x = np.arange(0, X+1, 1);
y = np.arange(0, Y+1, 1);
XX,YY = np.meshgrid(x,y)
levels = MaxNLocator(nbins=10).tick_values(0, 10)
cmap = plt.get_cmap('terrain')
norm = BoundaryNorm(levels, ncolors=cmap.N, clip=True)
cs2 = plt.contourf(XX,YY,abs(Co2), levels=levels, cmap=cmap)
plt.colorbar(cs2)
plt.title('c2 en mg/l')
plt.xlabel('x (m)')
plt.ylabel('y (m)')
plt.draw()
plt.pause(0.00001)
Pos = int(round((np.argmax(Co2))/100)-1)
h = plt.figure(3)
plt.plot(x, Co1[Pos,:], marker='+', linestyle='-', color='b', lw=0.3)
plt.plot(x, Co2[Pos,:], marker='.', linestyle='-', color='r', lw=0.3)
plt.xlabel('Distancia (m)')
plt.ylabel('Concentraci3n (mg/L)')
plt.legend(('c1','c2'), loc='upper right')
plt.grid(True)
print('Co2 max =',max(Co2[Pos,:]))
plt.show()

```

elif numero == 5:

```

#Datos del problema
Vm = 1    #dias^-1 para i=1,2
Kh = 0.1  #mg/l para i=1,2
k12 = 2.0 #constante
k21 = 0.5 #constante

#Especie estacionaria
X1 = 0.2  #mg/l en lugar de 0.2

#Tiempo de simulacion
t = 68    #days en el problema es 68 days

#Incremento espacial y temporal (Cambiar datos para laguna 1, 2 y 3)
X = 274   #Dominio espacial en metros en x #1 en 3 y 4
Y = 73    #Dominio espacial en metros en y #1 en 3 y 4
Z = 3     #Dominio espacial en metros en z

```

```

J = 274 #Contador en x
N = 73 #Contador en y
Q = 3 #Contador en z

Deltax = X/J #Incremento espacial en x, en metros
Deltay = Y/N #Incremento espacial en y, en metros
Deltaz = Z/Q #Incremento espacial en z, en metros
Deltat = 0.1 #Incremento temporal, en dias para 1,2 y 3

#Simulacion
#Condicion inicial y de frontera
c1 = np.zeros((J+1,N+1,Q+1)) #matriz de ceros en dominio espacial

#Escalar 1 c1 Oxigeno
Co1 = c1
Co1[:, :, :] = 6

#Escalar 2 c2 Carbono
c2 = np.zeros((J+1,N+1,Q+1)) #matriz de ceros en dominio espacial
Co2 = c2
Co2[0, :, :] = 69

#Datos de Flujo
Dx = 0.2 #m^2/dia
Dy = 0.2 #m^2/dia
Dz = 0.2 #m^2/dia

#Matriz de velocidades
Umax = genfromtxt('L1AUx.txt', delimiter=' ')
Umx = Umax.transpose() #agregar Ux[j,i] en la ecuacion

#exit()
#Matriz base para simulacion
Cm1 = Co1
Cm2 = Co2

t_sim = t / Deltat

m=0
while m < t_sim:
    m = m+1
    print("t simulacion (dias) = ",round(m*Deltat,4))
    for kk in range(1,Q):
        k = int(kk)
        for jj in range(1,J):
            j = int(jj)
            for ii in range(1,N):

```

```

i = int(ii)

if k == 0:

    Umx = np.zeros((J+1,N+1))
    Umy = np.zeros((J+1,N+1))
    Umz = np.zeros((J+1,N+1))

elif k == 1:

    #Matriz de velocidades
    Umax = genfromtxt('L1AUx.txt', delimiter=' ')
    Umx = Umax.transpose() #agregar Ux[j,i] en la ecuacion
    Umay = genfromtxt('L1AUy.txt', delimiter=' ')
    Umy = Umay.transpose() #agregar Ux[j,i] en la ecuacion
    Umaz = genfromtxt('L1AUz.txt', delimiter=' ')
    Umz = Umaz.transpose() #agregar Ux[j,i] en la ecuacion

elif k == 2:

    #Matriz de velocidades
    Umax = genfromtxt('L1BUx.txt', delimiter=' ')
    Umx = Umax.transpose() #agregar Ux[j,i] en la ecuacion
    Umay = genfromtxt('L1BUy.txt', delimiter=' ')
    Umy = Umay.transpose() #agregar Ux[j,i] en la ecuacion
    Umaz = genfromtxt('L1BUz.txt', delimiter=' ')
    Umz = Umaz.transpose() #agregar Ux[j,i] en la ecuacion

elif k == 3:

    #Matriz de velocidades
    Umax = genfromtxt('L1CUx.txt', delimiter=' ')
    Umx = Umax.transpose() #agregar Ux[j,i] en la ecuacion
    Umay = genfromtxt('L1CUy.txt', delimiter=' ')
    Umy = Umay.transpose() #agregar Ux[j,i] en la ecuacion
    Umaz = genfromtxt('L1CUz.txt', delimiter=' ')
    Umz = Umaz.transpose() #agregar Ux[j,i] en la ecuacion

if Co2[j,i,k] > Co1[j,i,k]:
    delta1 = 1
    delta2 = 0
elif Co1[j,i,k] > Co2[j,i,k]:
    delta1 = 0
    delta2 = 1

Ad1 = -Umx[j,i]*((Co1[j,i+1,k]-Co1[j,i-1,k])/(2*Deltax))-
Umy[j,i]*((Co1[j+1,i,k]-Co1[j-1,i,k])/(2*Deltay))-Umz[j,i]*((Co1[j,i,k+1]-Co1[j,i,k-1])/(2*Deltaz))

```

```

Di1 = Dx*((Co1[j,i+1,k]-2*Co1[j,i,k]+Co1[j,i-
1,k])/(Deltax)**2)+Dy*((Co1[j+1,i,k]-2*Co1[j,i,k]+Co1[j-
1,i,k])/(Deltay)**2)+Dz*((Co1[j,i,k+1]-2*Co1[j,i,k]+Co1[j,i,k-1])/(Deltaz)**2)
Re1 = -((Vm*X1)/(Kh+Co1[j,i,k]))*delta1*Co1[j,i,k]-
k12*((Vm*X1)/(Kh+Co2[j,i,k]))*delta2*Co2[j,i,k]
Cm1[j,i,k] = (Ad1+Di1+Re1)*Deltat+Co1[j,i,k]

Ad2 = -Umx[j,i]*((Co2[j,i+1,k]-Co2[j,i-1,k])/(2*Deltax))-
Umy[j,i]*((Co2[j+1,i,k]-Co2[j-1,i,k])/(2*Deltay))-Umz[j,i]*((Co2[j,i,k+1]-Co2[j,i,k-
1])/(2*Deltaz))
Di2 = Dx*((Co2[j,i+1,k]-2*Co2[j,i,k]+Co2[j,i-
1,k])/(Deltax)**2)+Dy*((Co2[j+1,i,k]-2*Co2[j,i,k]+Co2[j-
1,i,k])/(Deltay)**2)+Dz*((Co2[j,i,k+1]-2*Co2[j,i,k]+Co2[j,i,k-1])/(Deltaz)**2)
Re2 = -((Vm*X1)/(Kh+Co2[j,i,k]))*delta2*Co2[j,i,k]-
k21*((Vm*X1)/(Kh+Co1[j,i,k]))*delta1*Co1[j,i,k]
Cm2[j,i,k] = (Ad2+Di2+Re2)*Deltat+Co2[j,i,k]

```

```

Co1[:, :, :] = 6
Co2 = Cm2

```

```

if k*Deltat == 10:
    np.savetxt('Co1_10dias.txt', Co1, fmt='%e')
    np.savetxt('Co2_10dias.txt', Co2, fmt='%e')
elif k*Deltat == 20:
    np.savetxt('Co1_20dias.txt', Co1, fmt='%e')
    np.savetxt('Co2_20dias.txt', Co2, fmt='%e')
elif k*Deltat == 30:
    np.savetxt('Co1_30dias.txt', Co1, fmt='%e')
    np.savetxt('Co2_30dias.txt', Co2, fmt='%e')
elif k*Deltat == 40:
    np.savetxt('Co1_40dias.txt', Co1, fmt='%e')
    np.savetxt('Co2_40dias.txt', Co2, fmt='%e')
elif k*Deltat == 50:
    np.savetxt('Co1_50dias.txt', Co1, fmt='%e')
    np.savetxt('Co2_50dias.txt', Co2, fmt='%e')
elif k*Deltat == 59:
    np.savetxt('Co1_59dias.txt', Co1, fmt='%e')
    np.savetxt('Co2_59dias.txt', Co2, fmt='%e')

```

```
plt.show()
```

```

elif numero == 6:
    print ("Salir")
    exit()

```