



Fundação Universidade Federal de Mato Grosso do Sul

Serviço Público Federal  
Ministério da Educação



Luiz Eduardo Domingos de Oliveira

## MASS EXCHANGE IN DEAD ZONES: A NUMERICAL APPROACH

Campo Grande, MS  
January 2021

Federal University of Mato Grosso do Sul  
College of Engineering, Architecture and Urbanism and Geography  
Graduation Programme in Environmental Technologies

Luiz Eduardo Domingos de Oliveira

## MASS EXCHANGE IN DEAD ZONES: A NUMERICAL APPROACH

A dissertation submitted in partial fulfillment of the requirements for the Master of Science degree in the Graduation Programme in Environmental Technologies in the Federal University of Mato Grosso do Sul, academic area: *Environmental Sanitation and Water Resources*

**Advisors:**

Prof. PhD. Johannes Gérson Janzen

Prof. PhD Carlo Gualtieri (University of Naples Federico II)

Approved in: February 22th, 2020

**Dissertation Committee:**

---

Prof. PhD. Paulo Tarso Sanches de Oliveira  
UFMS

---

PhD. Manoel Lucas Machado Xavier  
UFMS

Campo Grande, MS  
January 2021

# Abstract

The hydrodynamics of dead waters (DZ) were investigated for two different types of structures: lateral cavities and groyne fields. A literature review of the main methods of investigation of this kind of flow was conducted in which knowledge gaps were identified. The structure of this dissertation starts with a numerical model of groyne fields that identified different phases in the mass exchange between the DZ and the main channel. Following, a numerical model was developed to describe the hydrodynamics of a lateral cavity using a hybrid method to account for the turbulence fields (Detached Eddy Simulation) under a commercial package. This model was further developed in a Large Eddy Simulation (LES) under an open-source package to make the data accessible. Lastly, the main topic of this dissertation was described in which the investigation of a vegetated lateral cavity was investigated. In this paper, we found the presence of a secondary circulation that was not expected for this geometry in a non-vegetated scenario. Still, an analysis of the flow and its variation in different vegetation densities was conducted where we found a threshold that divides the way the flow occurs.

# Resumo

A hidrodinâmica de zonas mortas foi investigada em dois diferentes tipos de estruturas: cavidades laterais e campos de espigão. Uma revisão de literatura dos principais métodos de investigação deste tipo de escoamento foi conduzida na qual identificamos lacunas a serem preenchidas. A estrutura desta dissertação começa com um modelo numérico de campos de espigão que identificou diferentes fases na qual a troca de massa entre o canal inalterado e a zona morta ocorre. Em seguida, um modelo numérico foi desenvolvido para descrever a hidrodinâmica de uma cavidade lateral usando um método híbrido para calcular os campos turbulentos (*Detached Eddy Simulation*) sob um pacote comercial. Este modelo foi melhorado no seguinte capítulo em uma simulação que considera os campos instantâneos do escoamento (modelo de turbulência *Large Eddy Simulation*) na qual um pacote de código aberto foi utilizado para uma ampliação do acesso do modelo. Finalmente, o principal tópico da dissertação foi descrito e consiste na investigação de uma cavidade lateral vegetada. Neste artigo, descobrimos a presença de uma circulação secundária que não era esperada para essa geometria, caso não houvesse vegetação. Além disso, o artigo trata da análise do escoamento e sua variação em diferentes níveis de densidade de vegetação a qual nos levou a encontrar um valor limite que divide o escoamento.

# Dedication

This is for my beloved Thaís, my parents and JoTa.

# Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

---

Luiz Eduardo Domingos de Oliveira

# Acknowledgements

I want to thank my family for all of their support. My parents have always provided an example of how to cherish life, to finish difficult tasks, and pushed me to pursue excellence. I want to honour my brother, João, for giving me friendship and an example of living life to the fullest.

I would like to thank Johannes Janzen and Carlo Gualtieri for their direction, drive, and ambition.

I would like to thank Taís N. Yamasaki, Felipe Costa and Filipe Queiroz for boarding the projects and co-authoring many of the here published works.

Also, I would like to thank Taís for helping me to develop this career since my first steps at the laboratory. She has been a great work colleague and a wonderful friend through all these years.

I would like to thank my laboratory colleagues for guiding this journey and for fellowship through out these years. All those coffees were amazing and it was always rewarding and joyful to gather with you.

Thank you to my Lord and Savior Jesus Christ for providing the grace and mercy to sustain and direct me through this endeavour.

# Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001

This study was partly conducted in Lobo Carneiro cluster located in NACAD/- Coppe - Rio de Janeiro, Brazil

This project was funded by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Brasil -Institutional Internationalisation Programme (Print)

# Contents

<b>Table of contents</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background and Motivation . . . . .	1
1.2 Hydrodynamics and Mass Exchange . . . . .	2
1.3 Ecology and Vegetation . . . . .	5
1.4 Objective and Research Questions . . . . .	7
1.5 Dissertation Structure . . . . .	7
<b>2 Mass Exchange in Dead Water Zones: A Numerical Approach</b>	<b>9</b>
Abstract . . . . .	9
2.1 Introduction . . . . .	10
2.2 Methods . . . . .	11
2.3 Results and Discussion . . . . .	14
2.4 Conclusion . . . . .	19
Acknowledgements . . . . .	19
Funding . . . . .	19
References . . . . .	19
<b>3 Hydrodynamics of Vegetated Lateral Cavities</b>	<b>22</b>

Abstract . . . . .	22
3.1 Introduction . . . . .	23
3.2 Methods . . . . .	24
3.3 Results and Discussion . . . . .	25
3.4 Conclusion . . . . .	26
Funding . . . . .	26
References . . . . .	26
<b>4 Velocity Estimates in Vegetated Lateral Cavities</b>	<b>28</b>
Abstract . . . . .	28
4.1 Introduction . . . . .	29
4.2 Methods . . . . .	30
4.3 Results and Discussion . . . . .	31
4.4 Conclusion . . . . .	34
References . . . . .	34
<b>5 The Effects of Vegetation Density Upon Flow and Mass Transport in Lateral Cavities</b>	<b>37</b>
Abstract . . . . .	37
5.1 Introduction . . . . .	38
5.2 Numerical Model . . . . .	41
5.2.1 Model Equations . . . . .	41
5.2.2 Simulation Setup . . . . .	43
5.2.3 Numerical Programme . . . . .	44
5.2.4 LES Quality and Grid Independence . . . . .	45
5.2.5 Validation . . . . .	45

5.3	Flow Characteristics . . . . .	46
5.4	Hydrodynamics of the Mixing Layer . . . . .	51
5.4.1	Thickness of the Mixing Layer . . . . .	51
5.4.2	Vorticity . . . . .	52
5.4.3	Turbulent Kinetic Energy (TKE) . . . . .	54
5.5	Mass Exchange . . . . .	58
5.6	Discussion . . . . .	60
5.7	Conclusion . . . . .	61
	Funding . . . . .	62
	References . . . . .	62
<b>6</b>	<b>Conclusion and Recommendations</b>	<b>69</b>
	<b>References</b>	<b>71</b>
	<b>Appendices</b>	<b>80</b>
	<b>Appendix A Total Turbulent Kinetic Energy Function</b>	<b>80</b>
	<b>Appendix B Data Processing</b>	<b>84</b>
B.1	File Structure . . . . .	85
B.2	preProcessing.py . . . . .	86
B.3	dataAnalysis.py . . . . .	88
B.4	preProcessing Scripts . . . . .	91
B.4.1	importCSV.py . . . . .	91
B.4.2	dataProcess.py . . . . .	95
B.4.3	mass.py . . . . .	102

B.4.4	plot.py . . . . .	105
B.4.5	thickness.py . . . . .	117
B.5	dataAnalysis Scripts . . . . .	123
B.5.1	multipleSimulationImport.py . . . . .	123
B.5.2	multipleSimulationProcess.py . . . . .	126
B.5.3	multipleSimulationPlot.py . . . . .	127
<b>Appendix C</b>	<b>Grid Convergence Index (GCI) Python Script</b>	<b>148</b>
C.1	File Structure . . . . .	149
C.2	main.py . . . . .	149
C.3	import.py . . . . .	151
C.4	gci.py . . . . .	153
C.5	gciLES.py . . . . .	156
C.6	plot.py . . . . .	164
<b>Appendix D</b>	<b>OpenFOAM Configuration of The Effects of Vegetation Density Upon Flow and Mass Transport in Lateral Cavities model</b>	<b>166</b>
D.1	File Structure . . . . .	167
D.2	0.orig/nut . . . . .	168
D.3	0.orig/p . . . . .	169
D.4	0.orig/tracer . . . . .	171
D.5	0.orig/U . . . . .	172
D.6	constant/fvOptions . . . . .	173
D.7	constant/g . . . . .	175
D.8	constant/transportProperties . . . . .	175
D.9	constant/turbulenceProperties . . . . .	176

D.10 system/blockMeshDict . . . . .	177
D.11 system/controlDict . . . . .	183
D.12 system/decomposeParDict . . . . .	200
D.13 system/fvSchemes . . . . .	201
D.14 system/fvSolution . . . . .	202
D.15 system/setFieldsDict . . . . .	206
D.16 system/topoSetDict . . . . .	207
D.17 system/totalTKE . . . . .	211
D.18 allClear . . . . .	214
D.19 mesh . . . . .	214
D.20 ramCache . . . . .	216
D.21 reconstructParParallel . . . . .	216

# List of Figures

1.1	Groyne Fields $53^{\circ}23'54.32''$ N, $10^{\circ}11'51.08''$ E, elevation 1.90 km. Terrain layer, viewed 29 August 2019. < <a href="http://www.google.com/earth/index.html">http://www.google.com/earth/index.html</a> >	2
1.2	Schema on the flow patterns of emergent lateral cavities (JACKSON; HAGERTY; APTE, 2013)	4
1.3	Time averaged quantities at $z/h = 0.95$ : a) streamwise velocity and b) TKE (MCCOY; CONSTANTINESCU; WEBER, 2008)	4
1.4	The variation of mean residence time ( $T_{DWZ}$ ) with the increase of vegetation density ( $a$ ) (XIANG; YANG; HUAI et al., 2019).	6
2.1	Upper view of the computational domain, from the free surface, and its boundary conditions.	12
2.2	Computational mesh: a) mesh in the free-surface plane; b) curvilinear grid around groyne tip; c) mesh in a vertical plane near the middle of the groyne field.	13
2.3	Mean velocity contour.	14
2.4	Mean streamwise velocity distributions. a) $x/L = 0.25$ b) $x/L = 0.50$ and c) $x/L = 0.75$ . The dashed line represents the groyne head position ( $y/h = 10.87$ )	15
2.5	Tracer mass fraction in the free-surface plane in time 229s.	16
2.6	Volumetric averaged mass concentration inside groyne field.	17
2.7	Volumetric averaged mass concentration inside groyne field fitted with two curves.	17

3.1	Computational domain. The flow direction is indicated by the grey arrow ('Entrada'). The dimensions are in metres and the coordinate origin ( $x, y, z = 0$ ) at the lower right portion of the channel. . . . .	24
3.2	Velocity distributions along the plane $z = 0.60H$ . . . . .	25
4.1	Numerical domain. The coordinate origins ( $x, y, z = 0$ ) is at the lower left corner of the picture. The inlet surface is at $x = 0\text{m}$ , outlet at $x = 0.75\text{m}$ and the cavity is between $0.25 < x(\text{m}) < 0.50$ , connected to the channel. . . . .	30
4.2	Mean velocity contour in the XY plane, in $z = 0.6H$ . . . . .	32
4.3	Mean velocity contour in the XY plane, in $z = 0.6H$ with additional streamlines associated to the flow. . . . .	33
4.4	Mean velocity vectors in the XY plane, in $z = 0.6H$ . . . . .	33
4.5	Comparison of the ensembled averaged mean velocity $u$ in the XY plane, in $z = 0.6H$ . . . . .	34
5.1	Computational domain with coordinates and dimensions. . . . .	43
5.2	Grid and Numerical Errors of the ensemble-averaged streamwise velocity in the cavity at $z = 0.6H$ , where $U$ is the bulk velocity in the main channel, $y_0 = 0.30\text{m}$ represents the beginning of the cavity and $H$ is the height of the flow. . . . .	46
5.3	Mean 2D streamlines of different vegetation densities at the horizontal plane $z/H = 0.6$ inside the cavity volume: a) Case 0, b) Case 1, c) Case 2, d) Case 3, e) Case 4, f) Case 5, g) Case 6, h) Case 7, i) Case 8, j) Case 9 and k) Case 10. . . . .	47
5.4	The variation of the streamwise velocity at the horizontal plane $z/H = 0.6$ inside the cavity volume. . . . .	48
5.5	The variation of the transversal velocity in the interface between the cavity and the main channel along the z-axis: a) Cases from 0 to 5; b) Cases from 5 to 10. Positive values of $v/U$ indicate the flow entering the cavity volume. . . . .	49
5.6	The variation of the transversal velocity in the interface between the cavity and the main channel along the x-axis. Positive values of $v/U$ indicate the flow entering the cavity volume. . . . .	50

5.7 Evolution of the mixing layer thickness averaged at the z-axis: a) inner mixing layer; b) outer mixing layer and c) total mixing layer. . . . .	52
5.8 Time averaged vorticity at $z/H = 0.6$ : a) Case 0, b) Case 1, c) Case 2, d) Case 3, e) Case 4 and f) Case 5. . . . .	53
5.9 Time averaged vorticity at $z/H = 0.6$ : a) Case 6, b) Case 7, c) Case 8, d) Case 9 and e) Case 10. . . . .	54
5.10 Time-averaged total kinetic energy (TKE) at $z/h = 0.6$ : a) Case 0, b) Case 1, c) Case 2, d) Case 3, e) Case 4 and f) Case 5. . . . .	56
5.11 Time-averaged total kinetic energy (TKE) at $z/h = 0.6$ : a) Case 6, b) Case 7, c) Case 8, d) Case 9 and e) Case 10. . . . .	57
5.12 Volumetric-averaged tracer concentration decay inside the lateral cavity over time: a) Case 1 b) Case 8. . . . .	58
5.13 The variation of the mass exchange coefficient and the mean retention time with the increase of vegetation density. . . . .	59

# List of Tables

2.1	Comparison of mean residence time inside groyne field and exchange coefficient in between experimental and numerical studies. . . . .	18
5.1	Vegetation levels and the calculated Darcy-Forchheimer coefficients, where $a$ (%) is the vegetation density, $d$ (1/m <sup>2</sup> ) is the viscosity drag coefficient, $f$ (1/m) is the inertial coefficient and $dh$ (m) is the hydraulic diameter. . .	45

# Chapter 1

## Introduction

In this chapter, the topic of this dissertation is first positioned in the research area and its objectives are developed. Following, a more specific definition of the topic will be given. Subsequently, the research objective and hypothesis will be presented.

### 1.1 Background and Motivation

Accidental pollutant spills in rivers can influence the water quality and the hydrodynamics of the flow over large portions of the channel. Therefore, the calculation of mass transport on rivers is an important aspect to determine the extension of the damage. The parameters that dictate the solute transport in streams and rivers are strongly related to geometric and hydrodynamic characteristics of the river (e.g., velocity distribution, channel width, flow depth, vortex shedding). Over large distances, the mass transport is mainly restricted to the length and thus it takes a 1-dimensional approach (WEITBRECHT, 2004). Although, the complexity of this solute transport must take into account different aspects of the river over its length, as rivers do not have a constant geometry over all its length. The accountability of the cross-section of the river leads to a 2-dimensional model. Despite the model now offer a variability range, there are aspects in natural and regulated rivers that introduce the third dimension due to rapid changes in the depth direction. Still, regions such as dead zones (DZ), that are regions separated from the main channel and have a net flux close to zero in the main stream direction which configures these structures as transient storage volumes. The formation of DZ can occur from any structure that creates an irregularity within the water body morphology, examples of structures are: groyne fields, lateral cavities, vanes, harbours and sidearms. The shared characteristic of these zones is its closeness, except for a single interface, the volume is completely dissociated from the main channel. This implies that the study of

this interface is essential to understand all the exchange processes between the DZ and the main channel.

Normally placed in shallow waters, the transport inside the DZ is regarded as a two-dimensional motion, except for the interface between the unaltered channel (main channel) and the DZ where complex three-dimensional motion occurs (XIANG; YANG; WU et al., 2020). The typical path in the interface follows the order where the fluid penetrates the DZ near the bottom of the channel and exits primarily in the top layer of the flow, also it enters approximately via the downstream portion and exits in the upstream of the DZ (WEITBRECHT, 2004; XIANG; YANG; WU et al., 2020). This structure can be considered as a transient storage volume.

The transient storage of mass inside the DZ has been known to provide refuge to aquatic communities as they seek shelter in slower-moving flows in the surface stream or the hyporheic zone (JACKSON; HAGGERTY; APTE, 2013). According to the same author, the benefits of this storage extends to water quality improvement as the solutes residence times increases further increasing the interaction of nutrient-rich surface waters with biogeochemically-reactive sediments. For instance, Schwartz e Kozerski (2003) detected in their sample larger amounts of element contents with sedimentary origin than from geogenic sources, the increase in mass settled to the riverbed. These settled matter can favour vegetation growth (Figure 1.1). The drag created by the presence of vegetation changes the flow and consequently the mass exchange rates, which increases the uncertainty of volumes captured within the seasons.



Figure 1.1: Groyne Fields 53°23'54.32" N, 10°11'51.08" E, elevation 1.90 km. Terrain layer, viewed 29 August 2019. <<http://www.google.com/earth/index.html>>

## 1.2 Hydrodynamics and Mass Exchange

The DZ is created by transversal structures placed on the riverbank, these structures diverge the flow creating a rotational field. The importance of DZ is due (1) the

enhancement in biodiversity (RIBI et al., 2014; HARVEY, 2016), (2) the function as a macro-roughness at the river banks, mitigating erosion (JUEZ, C. et al., 2018) and (3) act as a transient storage zone (JACKSON; HAGGERTY; APTE, 2013; DROST et al., 2014; JACKSON; APTE et al., 2015). The principal characteristic of the flow, in an emergent scenario, is the presence of gyres. These vortexes origin from the dissipation of moment that occurs in the interface layer between the DZ and the main channel. The shearing and flow separation at the leading edge form a mixing layer that extends until the downstream portion of the lateral cavity (UIJTTEWAAL, 2005; JACKSON; HAGGERTY; APTE, 2013). The shape and quantity of circulations inside the cavity are determined by a geometric aspect between the width (normal to the flow,  $W$ ) and length (parallel to the flow,  $L$ ) of the cavity. The aspect ratio  $W/L$  divides the flow in three configurations: (a)  $W/L < 0.5$  results in multiple circulations parallel to the main stream; (b)  $0.5 < W/L < 1.5$  results in a single circulation; and (c)  $W/L > 1.5$  results in multiple gyres transversal to the main stream (WEITBRECHT; JIRKA, 2001; JACKSON; HAGGERTY; APTE, 2013; SUKHODOLOV, 2014)(Figure 1.2).

The number of circulations in the system impacts on the mass exchange between the DZ and the main channel. As the mass decay inside the DZ follows a quick exponential decay in the early stages the rates get slower as the primary gyre transfers its mass out, in multiple gyres systems (JACKSON; HAGGERTY; APTE; COLEMAN et al., 2012; OLIVEIRA; JANZEN, 2020). After the main gyre transfers its mass, a slower exchange takes place between the second circulation into the primary one, since the velocity magnitudes in the secondary gyre are slower than the primary one. Henceforth, the mean residence time inside the DZ depends on the primary gyre residence time (early decay) and the secondary gyre volume (late decay) (JACKSON; HAGGERTY; APTE, 2013; OLIVEIRA; JANZEN, 2020).

From all the different structures that can create a DZ this study will focus on lateral cavities and groynes. A lateral cavity is a volume, normally, adjacent to the riverbank as an external structure (Figure 1.2). Groynes consist of a series of lateral cavities, normally, inside the channel course (Figure 1.1). The characteristics of the flow in both structures are similar. Although an important difference is in the stabilisation of the mixing layer, this region grows until the fourth-sixth rank until it reaches a developed state for groynes (Figure 1.3), in other words, once it stabilises the width of the interface the flow becomes *permanent* (WEITBRECHT, 2004; MCCOY; CONSTANTINESCU; WEBER, 2008; XIANG; YANG; WU et al., 2020). This behaviour is a key aspect for modellers as this is a way to save computational resources and still maintain the comprehensiveness of the model.

The mass exchange between DZs and the main channel was vastly studied in field,

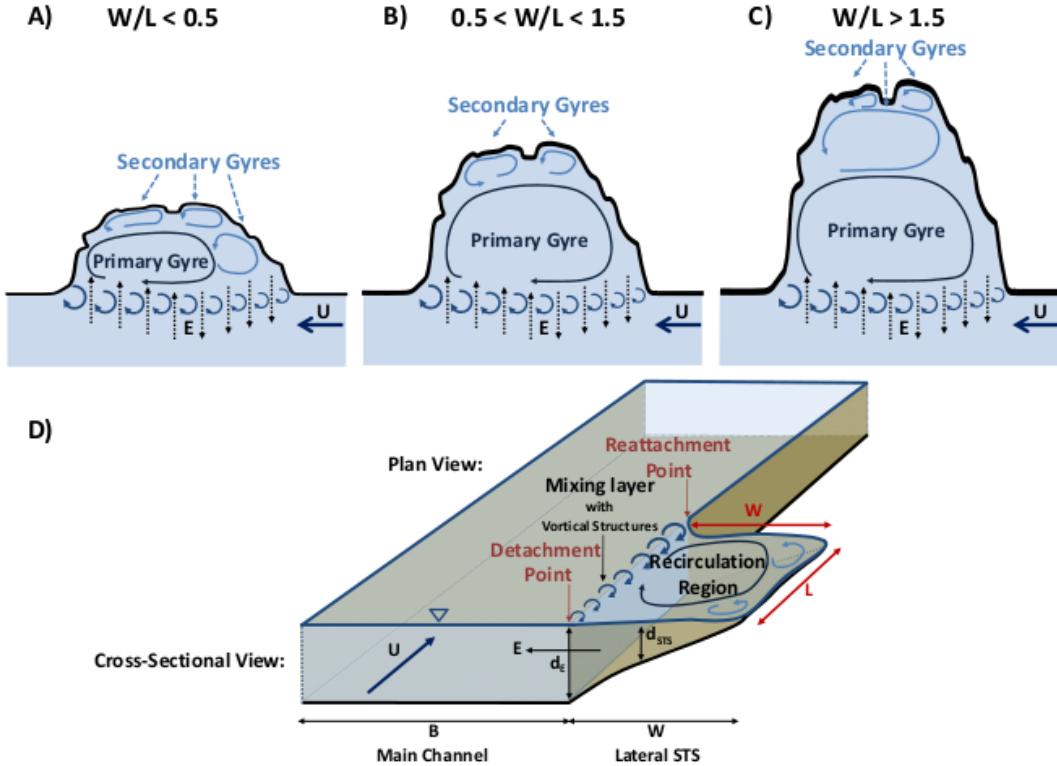


Figure 1.2: Schema on the flow patterns of emergent lateral cavities (JACKSON; HAGGERTY; APTE, 2013)

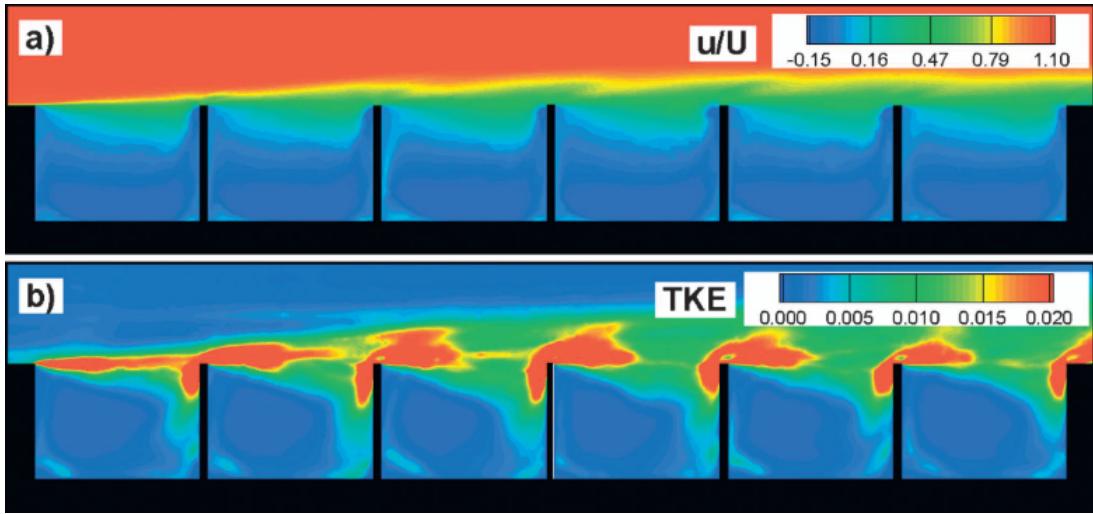


Figure 1.3: Time averaged quantities at  $z/h = 0.95$ : a) streamwise velocity and b) TKE (MCCOY; CONSTANTINESCU; WEBER, 2008)

experimental and numerical studies. Two different methods to estimate the velocity in which this exchange occurs are predominant: interface velocity measurements and tracer experiments.

As the effects of the exchange occur in a confined volume, Weitbrecht e Jirka (2001) proposed a model to account the exchanges in the interface between the zones. This method only requires geometrical parameters and the mean transversal velocity in the

interface surface. Despite this planar method give a good approximation on the exchange velocity the effect of mass diffusivity and depth variation is neglected, this further implies that systems slower circulations will have a larger impact of the estimated  $k$ , as the interface between the main channel and the cavity may remain with faster velocities. One can assume that this methodology works for conventional DZ, although it must be taken carefully for vegetated systems as the mixing layer alone may influence the result of  $k$ .

Another approach on the mass exchange is through tracer experiments, that can be divided into washout and pulse procedures, that consists in the ejection of mass from the interior of the DZ and a pulse at the inlet of the channel, respectively. Tracer methods treat the mass exchange tri-dimensionally as all the flow variables are considered. This approach gives a better understanding of the exchange in all conditions as it provides more information, for instance, the tracer methodology allows one to study the behaviour of mass in local regions of the volume or as a global volume. Furthermore, the coherent structures of the interface play a significant role in the transport of the tracer, given that the turbulence motion is transient, this method can capture the mass exchange rates over time and provide a better insight of the effect of those flow structures.

The advantage of the tracer method is the data richness that it provides, especially in numerical experiments. Some additional studies can be done to analyse other phenomena associated to mass, for instance, one can use a decay to estimate the amount of mass that is treated by plants or a settling velocity to preview sedimentation in the DZ. The only side effect of this method is the increased complexity to perform these experiments, be the difficulty in controlling the volume of water in the field or the calibration of the turbulent Schmidt number ( $S_{ct}$ ) in numerical studies.

### 1.3 Ecology and Vegetation

The presence of vegetation in river can also influence the hydrodynamics of the DZ and ,thus, the dispersion of solutes. Since the vegetation cover in rivers is dynamic, changing with seasonality and global climatic change, the dispersion of solutes in rivers is also dynamic. (SUKHODOLOVA et al., 2006), for example, studied the influence of the seasonality upon the longitudinal dispersion in a lowland river with vegetation. They observed that when vegetation is absent, the dead zones are represented predominantly by recirculation zones formed by flow separation on bank irregularities; in vegetative period, the dead zones are formed by blocking effect of vegetation occupying part of the river cross-section. These dead zones cause an increase of longitudinal dispersion, which means stronger lengthening of a solute cloud in the mainstream direction (WEITBRECHT,

2004).

The influence of vegetation in the mass exchange in lateral cavities was first studied in Xiang, Yang, Huai et al. (2019), that will be discussed in this paragraph. In this paper, a single lateral cavity was studied with a varied vegetation density. The vegetation was represented as solid cylinders inside the cavity volume. The cavity was emergent with a single circulation, due to its  $W/L = 0.6$ . The vegetation density ( $a$ ) ranged from 0 to 6.27% and as it increased more drag was introduced into the flow resulting in a slower circulation inside the volume. The turbulent kinetic energy in the DZ gradually decreased due to the blockage that impeded high energy vortexes from entering the volume. The effect on mass exchange occurred in two phases: first, there was a decay in the mean residence time due to the plant induced Karman vortex street and the plant blockage since the mixing rate from the vortex is greater than the blockage; in a second phase  $a > 3.96\%$  the blockage was higher than the mixing what increases of mean residence time (Figure 1.4).

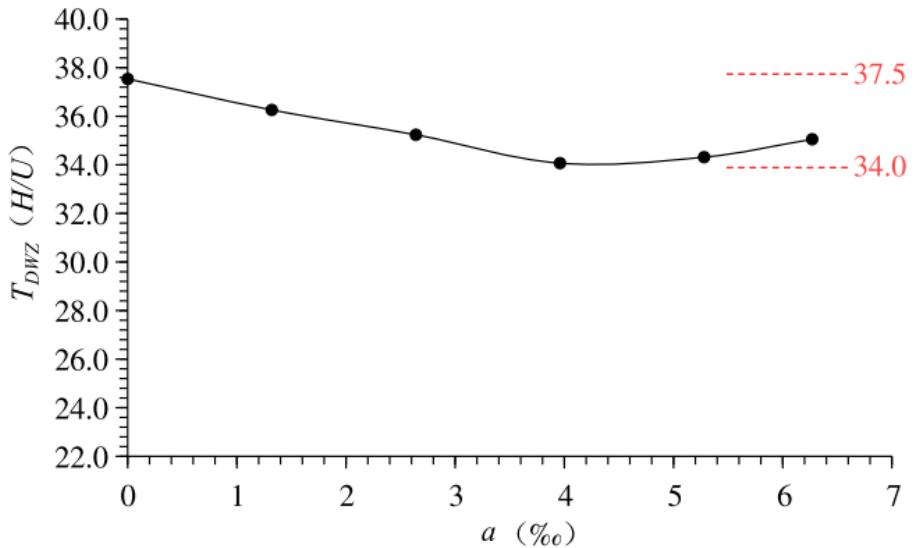


Figure 1.4: The variation of mean residence time ( $T_{DWZ}$ ) with the increase of vegetation density ( $a$ ) (XIANG; YANG; HUAI et al., 2019).

Although researchers have been studying the effect of groynes and vegetation on dispersion of solutes in rivers over the last decades (e.g. Sukhodolov, Sukhodolova e Krick (2017), Xiang, Yang, Huai et al. (2019) e Xiang, Yang, Wu et al. (2020)), there are still issues to that were not examined. For instance, in a vegetated groyne the vegetation levels were up to 15.7% (SUKHODOLOV; SUKHODOLOVA; KRICK, 2017), a larger vegetation density range may contain other phenomena that could not appear in the previous studies. This hypothesis indicates that studying the variation of density until the vegetation resistance blocks the flow would cover all the possible ranges and thereby

phenomena associated with this flow. Second, up to now the effects of the vegetation on the instantaneous fields were not investigated on lateral cavities. Third, the mass transfer at the main channel/dead zone was mainly studied using only the velocity at the interface, this leads to the opportunity of further describe the behaviour of mass inside the dead zone volume and how it affects the total exchange.

Furthermore, the study with vegetated cavities still could not identify a threshold for vegetation to be considered "dense" or "sparse" in cavities, and its understanding will allow researchers to identify flow modifications in the cavity (e.g., the suppression of recirculation gyres, the complete suppression of flow, the exchange coefficient asymptote, etc.). For emergent vegetation patches in an open channel, Chen et al. (2012) characterised them as being "dense" or "sparse" according to flow blockage thresholds, in which the flow properties near the patch (e.g., flow adjustment length and the velocity exiting the patch) were distinct above and below the threshold. A similar approach can be done for vegetated cavities.

## 1.4 Objective and Research Questions

In order to address this issues, the objective of this study is:

*To describe the hydrodynamics and the mass exchange between vegetated dead waters and the undisturbed section of the flow for different vegetation densities.*

The main hypothesis is that there is a threshold between dense and sparse vegetation in dead water. We also hypothesised that there is a threshold where the flow ceases inside the lateral cavity.

Specific methodological objectives:

- The choice of the modelling technique;
- The choice of the modelling package (open and commercial software);
- The method to represent the vegetation drag.

## 1.5 Dissertation Structure

The dissertation was divided into six chapters. Following the Introduction, the first paper is presented where the methodology of the study of groyne fields is firstly introduced. Thirdly, the presented paper discussed the first approach to the study of

lateral cavities. Fourthly, further development of the numerical model is presented, this implementation focused on the accessibility of the model by making use of open-source tools. Fifth, the main topic of this dissertation is introduced in the paper that describes the influence of the vegetation density in lateral cavities. Finally, the conclusive remarks about the flow in DZ and its mass exchange were presented.

# Chapter 2

## Mass Exchange in Dead Water Zones: A Numerical Approach

In this chapter, the first topic of the dissertation is presented as a published conference paper. The objective of this paper was to develop a simple numerical method capable of estimating the flow and the mass exchange between consecutive groyne fields and the main channel.

The original paper was published in the book 'Water, Energy and Food Nexus in the Context of Strategies for Climate Change Mitigation', under Springer publishing and can be found in <https://www.springer.com/gp/book/9783030572341>.

### Authors

- Luiz Eduardo Domingos de Oliveira <sup>1</sup>
- Johannes Gérsen Janzen <sup>1</sup>

### Abstract

Dead water zones (DWZs) in natural open channels, formed by consecutive groynes, are regions separated from the main channel, characterized by recirculating flows. These regions present smaller velocities compared to the main channel, increasing the deposition of sediment and the temporary storage of polluted materials. Exchange processes between

---

<sup>1</sup>Federal University of Mato Grosso do Sul

DWZs and the main channel influence the transport of pollutants in channels. This study adopts the k-omega shear stress transport (SST) turbulence model to examine the mass exchange between the main channel and the DWZ created by an infinite series of groynes. The computational results were compared to data collected in literature. A good agreement was achieved in mass exchange coefficient, with a relative error of approximately 2%.

**Keywords:** Dead water zones (DWZ); Groyne Fields; Mass Exchange; Open Channel; Computational Fluid Dynamics (CFD).

## 2.1 Introduction

In fluvial engineering, channels are generally shaped by complicated boundaries that can be composed by dead water zones (DWZ), which can be formed by consecutive groynes (XIANG; YANG; HUAI et al., 2019). Groynes are transversal dykes placed in sequence along riverbanks keeping the flow away from the banks. The effects of this structure in rivers are an increase in mean velocity and water depth in the main channel, improved navigability; increased efficiency of sediment transport; protection against flooding and the mitigation of bank erosion (MCCOY; CONSTANTINESCU; WEBER, 2008). Its placement also provides lateral heterogeneity that can favour the presence of aquatic organisms, improving the biodiversity of river ecosystems (MCCOY; CONSTANTINESCU; WEBER, 2008; SZLAUER-ŁUKASZEWSKA, 2015; BUCZYŃSKI et al., 2017; MIGNOT et al., 2017; BUCZYŃSKA et al., 2018; XIANG; YANG; HUAI et al., 2019).

Since the magnitude of mean flow velocities inside the DWZ is approximately 25% of the flow velocities in the main channel, not only the deposition of sediment is enhanced, but also nutrients and contaminants which are readily attached to fine particles (SUKHODOLOV, 2014). For instance, the attachment of contaminants to particles was observed in the Middle Elbe River, in Germany, leading to a low standard classification from an ecological view (SCHWARTZ; KOZERSKI, 2003). The authors found, in the groyne fields, the deposition of fresh organic mud with high nutrient and pollution content (e.g. nitrogen). The deposition of pollution content attached to sediments creates a problem for river management (UIJTEWAAL, 2005), especially in flood seasons, when the groyne field becomes submersed, being a source of contaminants to the main channel.

Therefore, in order to estimate the transport of pollutants in a channel, it is important to be able to understand and predict the exchange processes between the main

channel and the DWZ formed between groynes (WEITBRECHT; JIRKA, 2001). These exchange processes were studied in detail in a series of laboratorial experiments carried out by Weitbrecht (2004). Hinterberger, Fröhlich e Rodi (2007) used large eddy simulation (LES) to model Weitbrecht' experimental results. Although being a very precise model, LES is also more time consuming when compared to simpler models. Therefore, this study aims to investigate the mass exchange between the main channel and the groyne field using a simpler two-equations turbulence model, k-omega SST. The computational results are compared to Weibrech results and a good agreement was obtained.

## 2.2 Methods

The geometry was chosen to match the groynes from the second series of experiments described in Weitbrecht (2004). The flow depth ( $h$ ) was kept constant at 0.046 m and the experimental channel width ( $B$ ) at 1.80m. The emergent groynes were 0.50m long ( $W$ ) and spaced 1.25m apart ( $L$ ), producing an aspect ratio of  $W/L = 0.40$ . The groyne heads were in a semi-circle format with diameter of 0.05m. The Reynolds number was 7360, and thereby turbulent.

The flow past the most downstream-located groyne in the series had a periodic behaviour (HINTERBERGER; FRÖHLICH; RODI, 2007). Consequently, only one complete groyne field and two halves (located upstream and downstream from the complete one) was computed and a translational periodic boundary condition was imposed (Figure 2.1). The mean streamwise velocity in the computational domain was approximately  $U = 11\text{cm/s}$ , which corresponds to a mass flux of  $6.56 \text{ kg/m}^2$  of water in the periodic zones.

As the effects of the obstacles in the main channel extends up to one obstacle length in the transversal direction (y-axis) (BREVIS; GARCÍA-VILLALBA; NIÑO, 2014) the domain was two-thirds of the experimental flume width (B), reducing the computational effort. A free-slip symmetry boundary condition was imposed on the surface (Figure 2.1). This boundary condition was also used on the free surface plane as it is an acceptable simplification for flows with Froude numbers smaller than 0.5 (our Froude number was 0.24) (ALFRINK; RIJN, 1983). All walls, bed, lower side wall and groyne walls were considered hydraulically smooth.

The domain was calculated in a three-dimensional grid (Figure 2.2 a). The spatial discretization had a higher refinement in regions close to walls and at high velocity gradients regions. The meshing of the groyne's heads considered its curvature and the

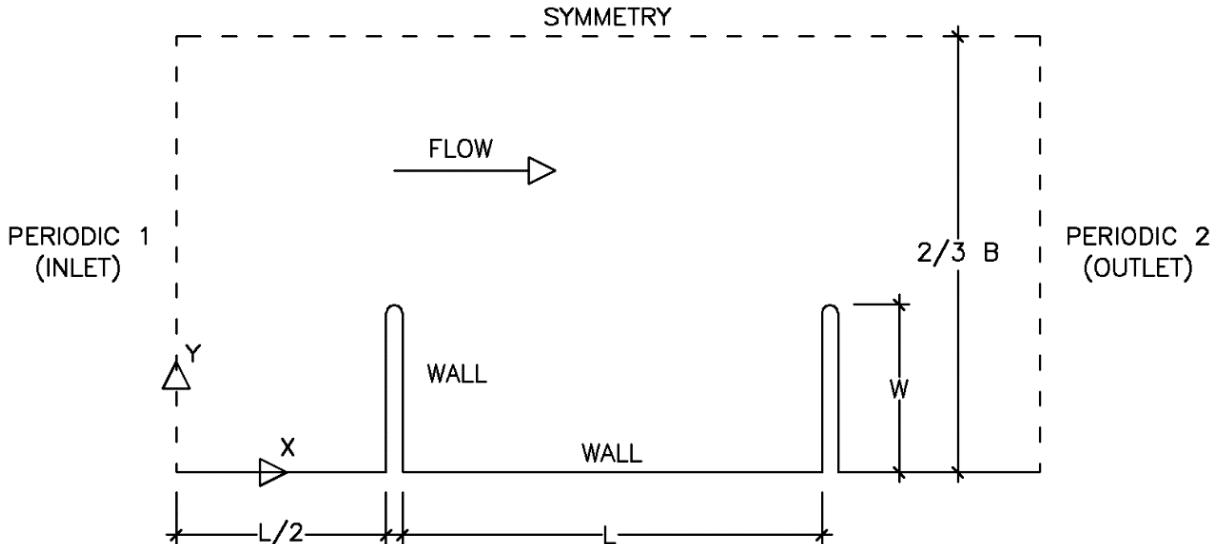


Figure 2.1: Upper view of the computational domain, from the free surface, and its boundary conditions.

proximity to the wall. This region used an O-grid with increasing element size (Figure 2.2 b). The mesh had 20 divisions in the z-axis, increasing gradually from the bottom of the channel to its free surface (Figure 2.2 c). In the y-axis, the groyne field had 70 divisions that gradually increased in size as it gets closer to the middle of the field. The strip that contains the groyne's heads had finer elements due to the momentum transfer in the shear layer. The total grid presented approximately one million elements.

The commercial software called Ansys® FLUENT (version 14) was used to solve the grid, using the finite volume method to discretize the governing mass and momentum equations. The turbulent model chosen is based at Reynolds-averaged Navier-Stokes equations (RANS) approach, that consists of time averaged equations for fluid flow. The turbulent calculations were solved using the k-omega SST model proposed by Menter et al. (2005), due to its capability of solving fluid flow in low Reynolds numbers. The pressure-velocity coupling method was SIMPLE and the gradient spatial discretization was Least Squares Cell Based. The momentum was discretized in a third order MUSCL scheme. The turbulent kinetic energy ( $tke$ ) and specific dissipation rate ( $\omega$ ) were discretized in a second order upwind scheme.

In addition to the velocity field, tracer concentration fields were also calculated by solving the following transport equation

$$\frac{\partial}{\partial t}(\rho Y_i) + \nabla(\rho \vec{v} Y_i) = \nabla(\rho D_{i,m} + \frac{\mu_t}{Sc_t}) \nabla Y_i \quad (2.1)$$

$$Sc_t = \frac{\mu_t}{\rho D_t} \quad (2.2)$$

where  $\rho$  is the fluid mass density,  $Y_i$  is the local mass fraction of each species,  $D_{i,m}$  is the mass diffusion coefficient for species in the mixture,  $\vec{v}$  is the velocity vector,  $Sc_t$  is the

turbulent Schmidt number (Equation 2.2),  $\mu_t$  turbulent viscosity and  $D_t$  the turbulent diffusivity. In other terms, the transport equation means that the rate of change and the net rate of flow (convection) equals the rate of change due to diffusion.

Equation (2.1) does not consider any chemical reactions or addition of phases during the solution and was discretized in a second order upwind scheme. The tracer was conservative, pursuing the same properties than water.

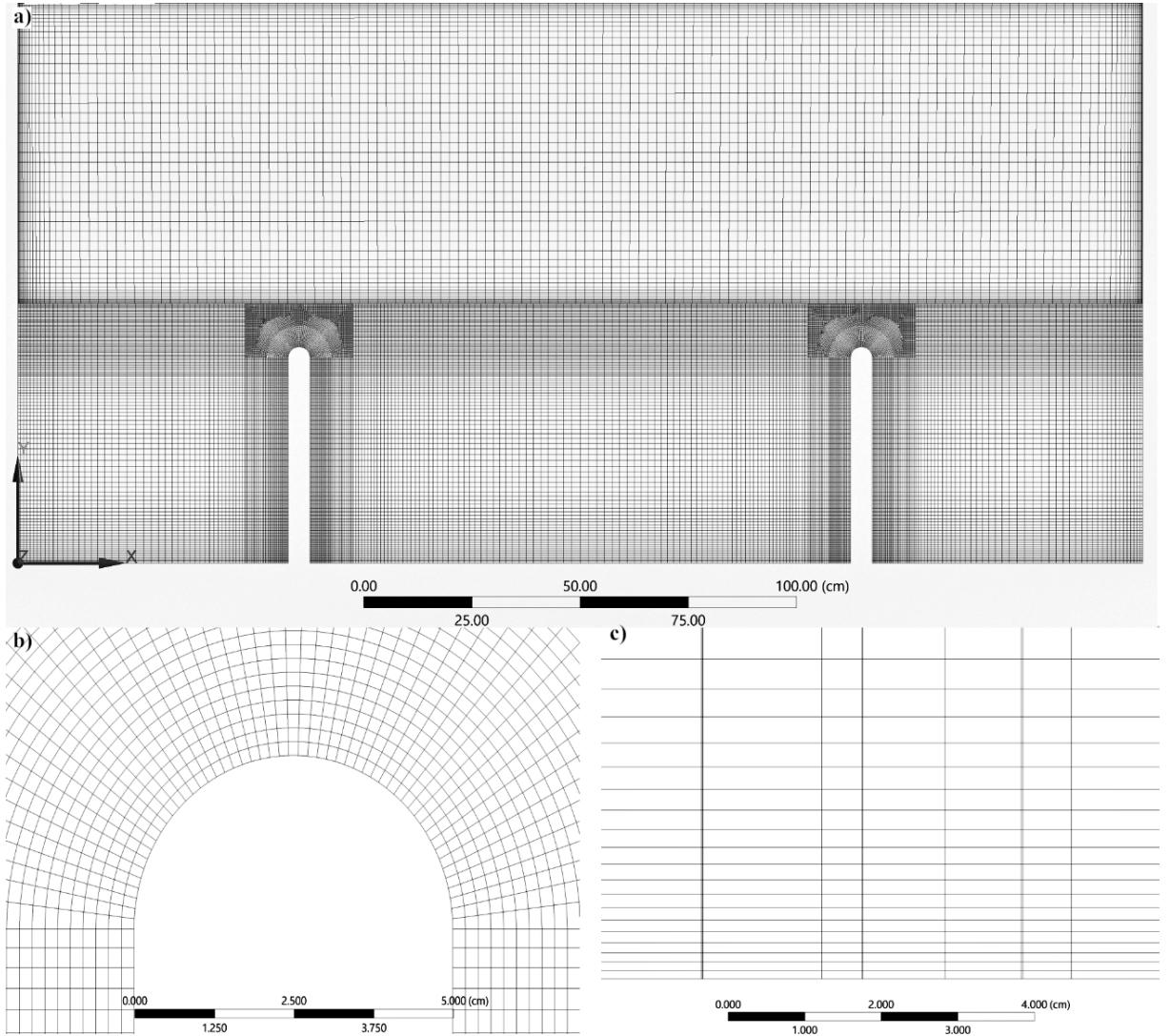


Figure 2.2: Computational mesh: a) mesh in the free-surface plane; b) curvilinear grid around groyne tip; c) mesh in a vertical plane near the middle of the groyne field.

The time step in the simulation was  $0.024h/U$ . The simulation was run for nearly  $180h/U$  until the fully developed state was achieved. Once the flow reached the fully developed state, the tracer mass fraction was set to 1 within the groyne field and 0 in the other parts of the channel. Then, statistics of the mean flow and tracer transport were calculated using the instantaneous flow fields and mean tracer concentration inside the groyne field over the next  $548h/U$ .

## 2.3 Results and Discussion

Two gyres could be observed in the groyne field. A large primary gyre (right vortex in the central groyne field) and a small secondary gyre in the upstream groyne (Figure 2.3). The formation of this system occurred by the momentum transferred by the main channel through a mixing layer. As the main flow went downstream, the shear in between zones excited an anticlockwise gyre (primary gyre) that further excited a smaller clockwise circulation (secondary gyre) that had no contact with the main channel. The secondary gyre was smaller in size (approximately 21% of the groyne field area) and velocity magnitudes, when compared to the mean circulation (Figure 2.3).

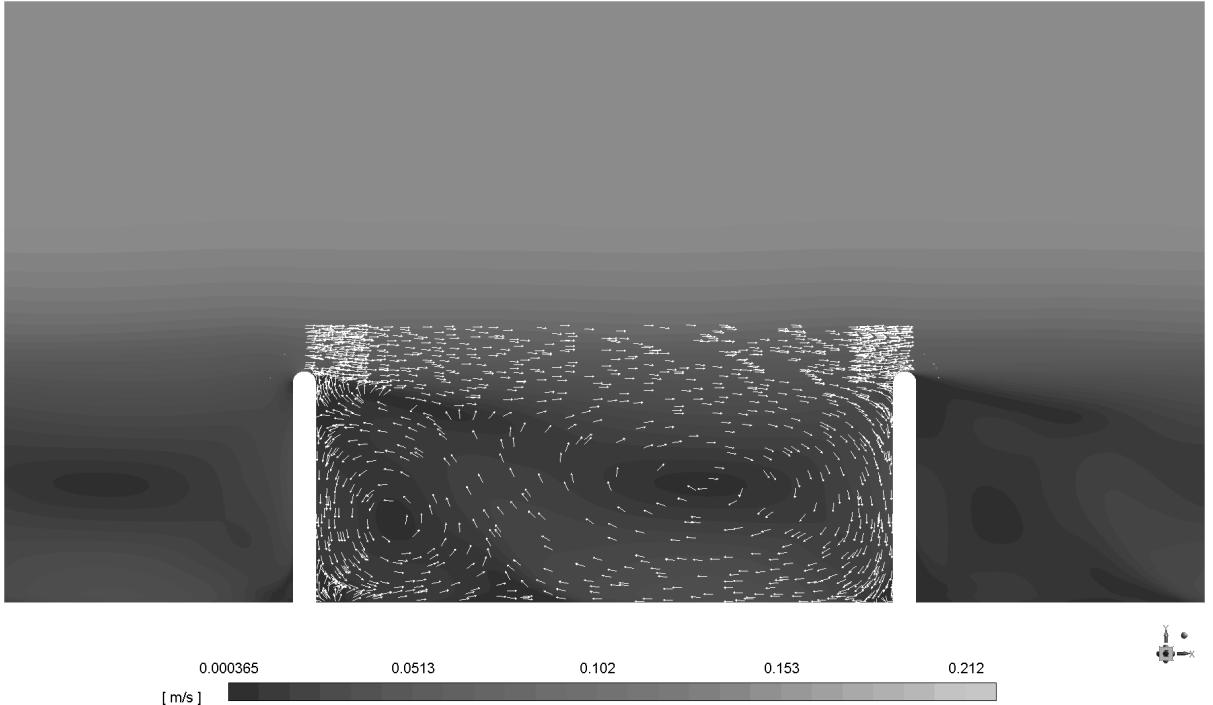


Figure 2.3: Mean velocity contour.

Figure 2.4 shows the mean streamwise velocity distributions for  $x/L = 0.25, 0.50$  and  $0.75$  ( $x$  has origin in the right face of the first groyne and points to the right). Overall, the model had a good accordance in the main channel and in the central part of the groyne field. The computational model was able to capture the circulation pattern inside the groyne field. However, near the groyne heads (interface between the main channel and the groyne fields) the concordance was not so good. This is due to the high dissipation of momentum that occurred in the mixing layer. Despite the fine resolution of the grid, the model could not describe the flow inside this region. For the same reason the secondary gyre did not have contact with the mixing layer, since this vortex was formed by the dissipation of momentum from the primary gyre. The mean error was approximately 102%, 21 % and 47 % for Figure 2.4 a), b) and c), respectively. However, the flow was in the same order of magnitude than the experimental, which indicates that (Figure 2.3)

represents qualitatively, at least, the flow within the region.

The ejection of tracer from a groyne field to the mixing layer (region between the DWZ and the main channel) occurs in the upstream portion of the field (up to 40%), while the following 60% is a region where mass can re-enter the system (WEITBRECHT, 2004). The tracer concentration stayed higher in the secondary gyre, while the primary gyre oscillated due to the injection of tracer from the mixing layer and its natural ejection (Figure 2.5). This movement was captured by the model and can be seen completely in <https://youtu.be/9b-4JZJdeA0>.

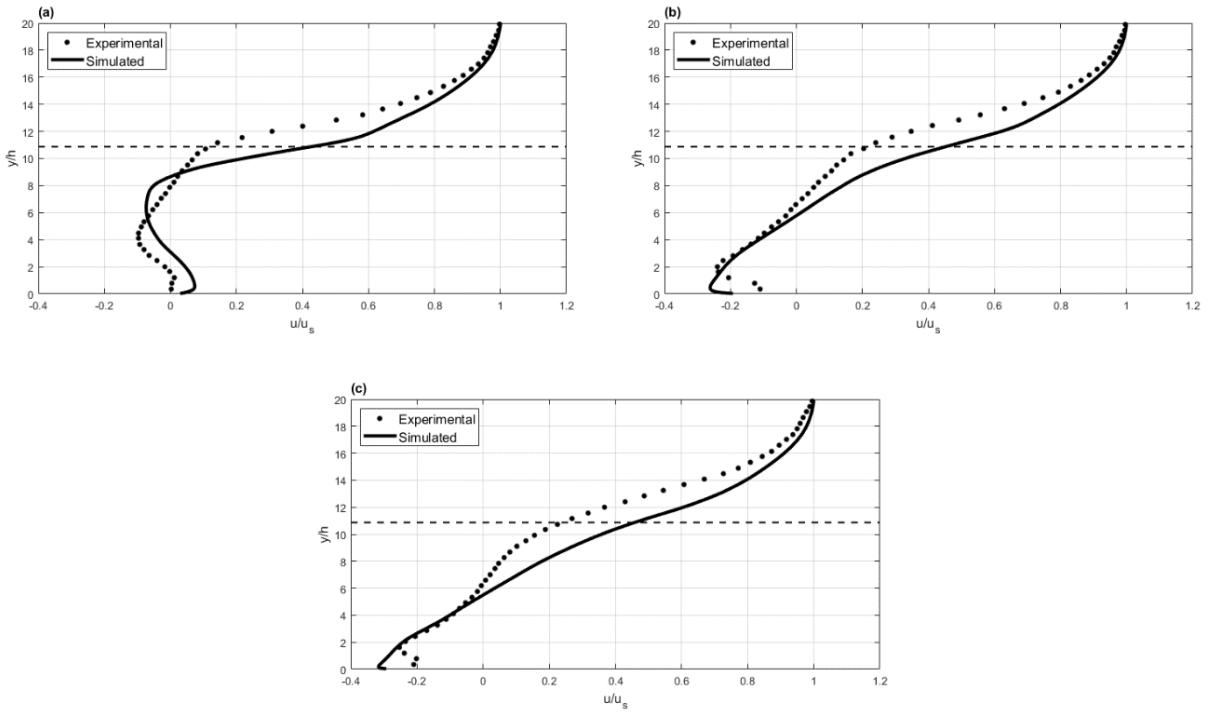


Figure 2.4: Mean streamwise velocity distributions. a)  $x/L = 0.25$  b)  $x/L = 0.50$  and c)  $x/L = 0.75$ . The dashed line represents the groyne head position ( $y/h = 10.87$ ).

The tracer concentration inside the field was fitted in a first order decay model (Equation 2.3) following the same procedure from the experimental study (Figure 2.6).

$$C = C_0 e^{\frac{-t}{MRT}} \quad (2.3)$$

Where MRT is the mean retention time. Based on the MRT, the mass coefficient  $k$  (Equation 2.4) was calculated in order to estimate the intensity of mass exchange (WEITBRECHT; JIRKA, 2001).

$$k = \frac{W}{MRTU} \quad (2.4)$$

The fitted curve presented an  $MRT = 117.7\text{s}$  that related to an exchange coefficient of  $k = 0.026$ . The relative error between the mean value of Weitbrecht' experiments and our model was 1.99% for the exchange coefficient and 1.69 % for MRT (Table 2.1).

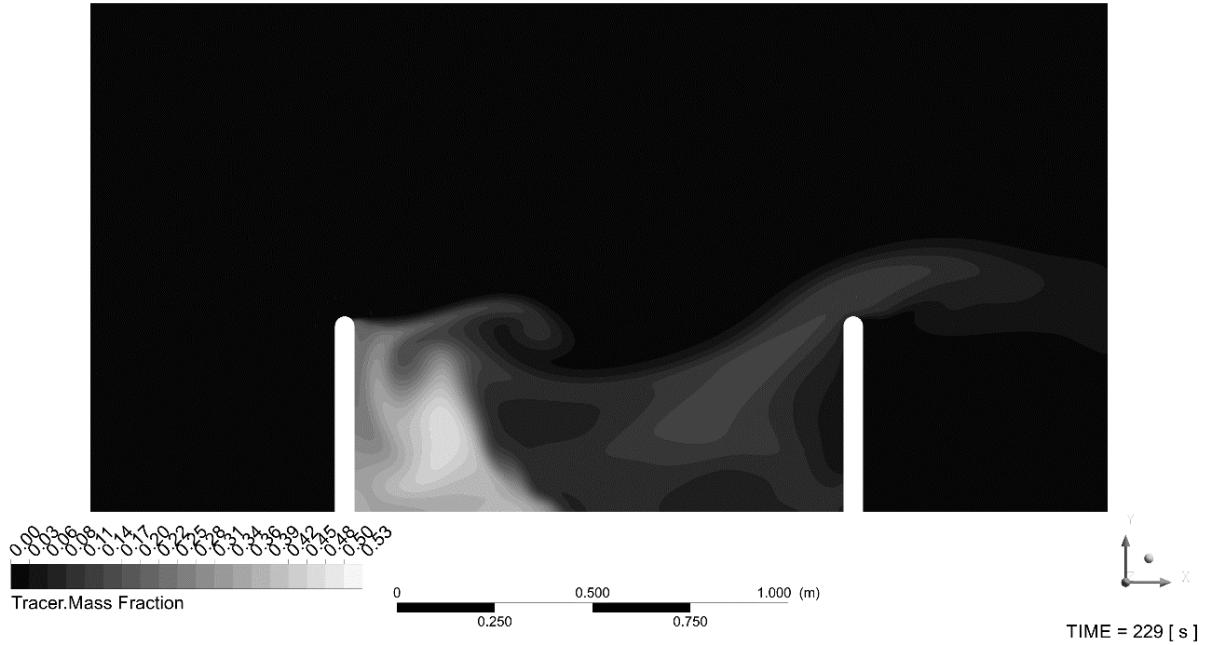


Figure 2.5: Tracer mass fraction in the free-surface plane in time 229s.

Although we could observe a good fitting between our computational model and the experimental results, it can be observed that the system presented two slopes, with a breakpoint near  $C/C_0 = 0.2$  (Figure 2.6). The first slope was influenced by the tracer concentration present in the primary gyre, that oscillates between ejecting mass and re-absorbing via the shear layer. The second one ejects mass slower, as the concentration in the field was mainly disposed in the secondary gyre. Figure 2.7 shows the tracer concentration fitted in two curves, the first curve presented an  $MRT = 113.27s$  and a  $k = 0.0274$  while the second  $MRT = 121.43s$  and  $k = 0.0256$ . The summary of the model and comparisons with previous studies can be seen in Table 2.1.

Our results are consistent with field observations. Sukhodolov (2014), for example, observed that the mass concentrated in the secondary gyre, since it presented the slowest velocities in the groyne field.

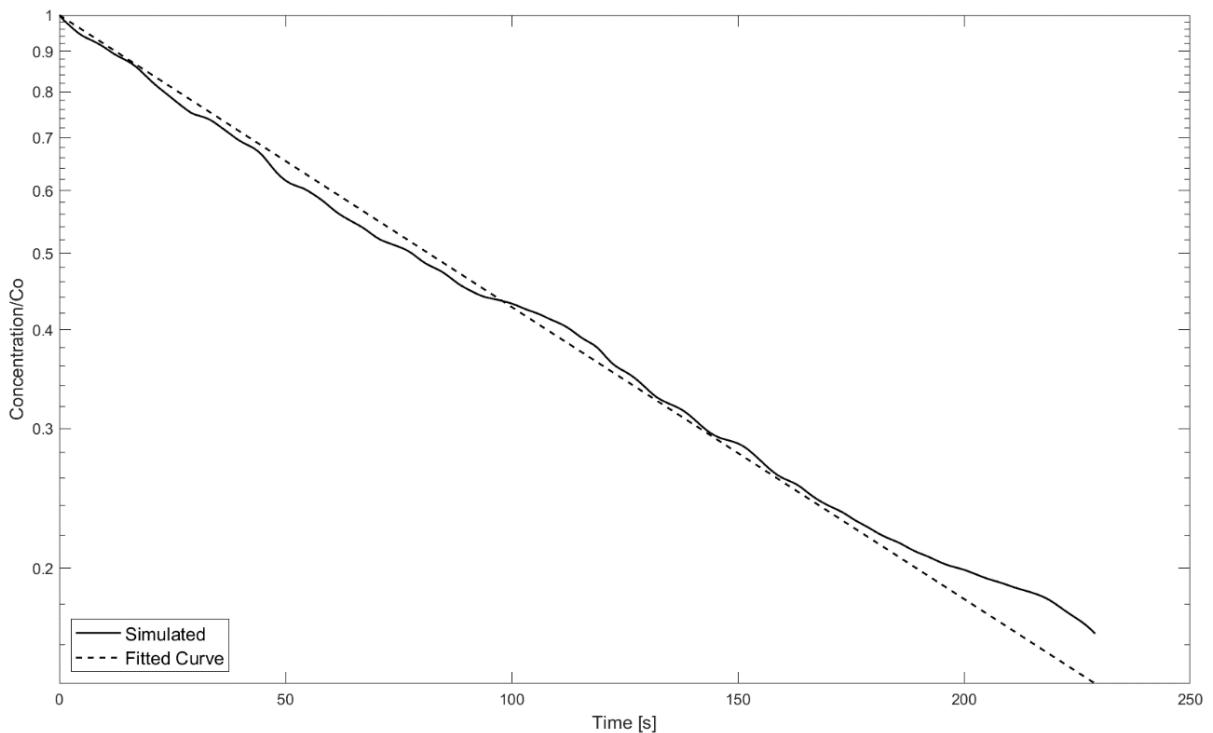


Figure 2.6: Volumetric averaged mass concentration inside groyne field.

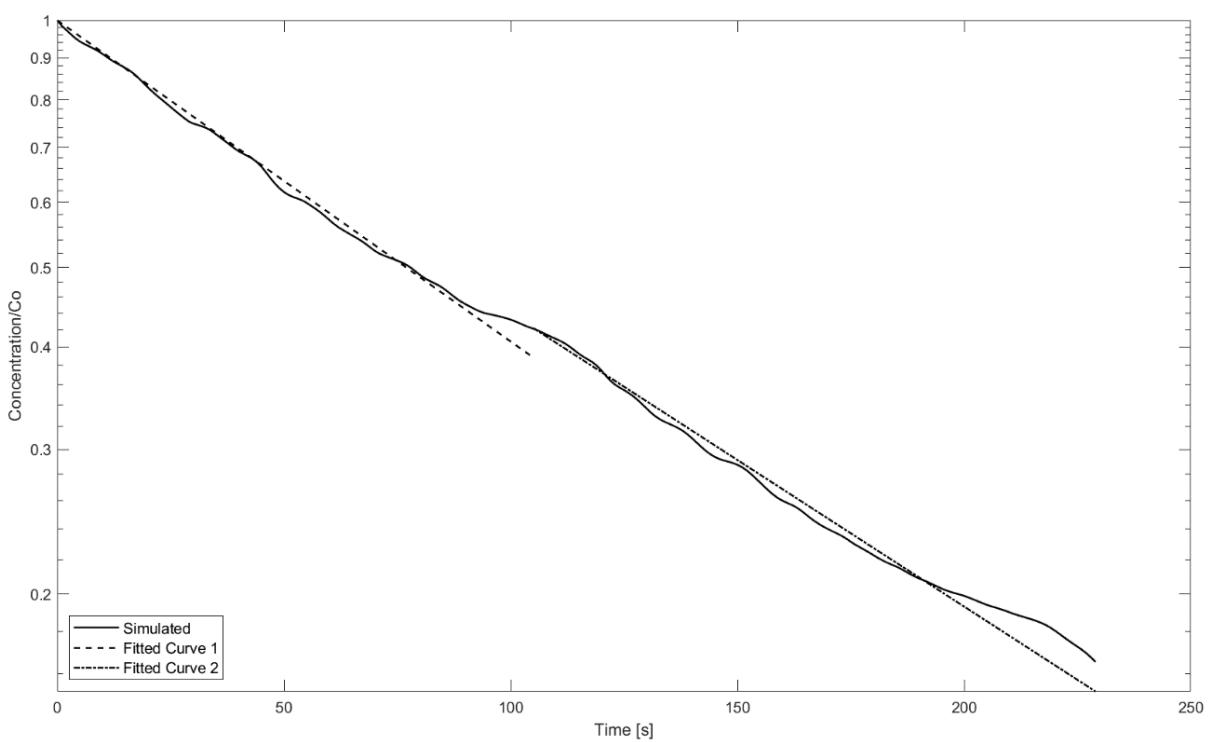


Figure 2.7: Volumetric averaged mass concentration inside groyne field fitted with two curves.

Table 2.1: Comparison of mean residence time inside groyne field and exchange coefficient in between experimental and numerical studies.

Experiment/Model	MRT [s]	k
Experiment 1	97	0.029
Experiment 2	114	0.028
Experiment 3	125	0.022
Mean value of experiments	118	0.027
3D LES \cite{Hinterberger2007}	137	0.023
2D LES \cite{Hinterberger2007}	75	0.042
3D k-omega SST (global fitted curve)	117.7	0.026
3D k-omega SST (first slope)	113.3	0.0274
3D k-omega SST (second slope)	121.62	0.0256

## 2.4 Conclusion

A 3D k-omega SST simulation was presented for a periodic shallow water flow in a groyne field. Our model was able to reproduce a similar structure and magnitude flow compared to experimental data. Furthermore, our model could predict the mass exchange coefficient between the main channel and the DWZ and the mean retention time of the DWZ, being in good concordance with experimental results. In agreement to experimental and field observations, the decay of mass inside the field is described in two phases, first when the primary gyre dominates the ejection and second when the mass is concentrated in the second gyre prolonging the MRT. Hence, a simpler model than LES can predict the main parameters related to the mass exchange process in groyne structures.

## Acknowledgements

The authors are grateful to members of SCF laboratory for providing the necessary hardware. Specially, D.F. Silva, M.L.M. Xavier, P.H.S de Lima, T.C.R Ventura and T.N. Yamasaki for research assistance and hardware set up.

## Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) -Finance Code 001.

## References

- ALFRINK, Ben J.; RIJN, Leo C. van. Two-Equation Turbulence Model for Flow in Trenches. **Journal of Hydraulic Engineering**, v. 109, n. 7, p. 941–958, 1983. DOI: 10.1061/(asce)0733-9429(1983)109:7(941). Disponível em: <<http://ascelibrary.org/doi/10.1061/%5C%28ASCE%5C%290733-9429%5C%281983%5C%29109%5C%3A7%5C%28941%5C%29>>. Citado nas pp. 11, 43.
- BREVIS, W.; GARCÍA-VILLALBA, M.; NIÑO, Y. Experimental and large eddy simulation study of the flow developed by a sequence of lateral obstacles. **Environmental Fluid Mechanics**, v. 14, n. 4, p. 873–893, 2014. ISSN 15677419. DOI: 10.1007/s10652-013-9328-x. Citado nas pp. 11, 43.

BUCZYŃSKA, Edyta et al. Human impact on large rivers: the influence of groynes of the River Oder on larval assemblages of caddisflies (Trichoptera). **Hydrobiologia**, v. 819, n. 1, p. 177–195, 2018. ISSN 15735117. DOI: 10.1007/s10750-018-3636-6. Citado na p. 10.

BUCZYŃSKI, P. et al. Groynes: A factor modifying the occurrence of dragonfly larvae (Odonata) on a large lowland river. **Marine and Freshwater Research**, v. 68, n. 9, p. 1653–1663, 2017. ISSN 13231650. DOI: 10.1071/mf16217. Citado na p. 10.

HINTERBERGER, C.; FRÖHLICH, J.; RODI, W. Three-Dimensional and Depth-Averaged Large-Eddy Simulations of Some Shallow Water Flows. **Journal of Hydraulic Engineering**, v. 133, n. 8, p. 857–872, ago. 2007. ISSN 0733-9429. DOI: 10.1061/(asce)0733-9429(2007)133:8(857). Disponível em: <<http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D290733-9429%7B%5C%7D282007%7B%5C%7D29133%7B%5C%7D3A8%7B%5C%7D28857%7B%5C%7D29>>. Citado na p. 11.

MCCOY, Andrew; CONSTANTINESCU, George; WEBER, Larry J. Numerical Investigation of Flow Hydrodynamics in a Channel with a Series of Groynes. **Journal of Hydraulic Engineering**, v. 134, n. 2, p. 157–172, 2008. ISSN 0733-9429. DOI: 10.1061/(asce)0733-9429(2008)134:2(157). Disponível em: <<http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D290733-9429%7B%5C%7D282008%7B%5C%7D29134%7B%5C%7D3A2%7B%5C%7D28157%7B%5C%7D29>>. Citado nas pp. 3, 4, 10.

MENTER, F. R. et al. Transition Modelling for General Purpose CFD Codes. **Engineering Turbulence Modelling and Experiments 6**, August, p. 31–48, 2005. ISSN 1386-6184. DOI: 10.1016/b978-008044544-1/50003-0. Citado na p. 12.

MIGNOT, Emmanuel et al. Measurement of mass exchange processes and coefficients in a simplified open-channel lateral cavity connected to a main stream. **Environmental Fluid Mechanics**, Springer Netherlands, v. 17, n. 3, p. 429–448, 2017. ISSN 15731510. DOI: 10.1007/s10652-016-9495-7. Citado nas pp. 10, 38, 40, 51.

SCHWARTZ, René; KOZERSKI, Hans-Peter. Entry and Deposits of Suspended Particulate Matter in Groyne Fields of the Middle Elbe and its Ecological Relevance. **Acta hydrochimica et hydrobiologica**, v. 31, 4-5, p. 391–399, 2003. DOI: 10.1002/aheh.200300496. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/aheh.200300496>. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/aheh.200300496>>. Citado nas pp. 2, 10.

- SUKHODOLOV, Alexander N. Hydrodynamics of groyne fields in a straight river reach: insight from field experiments. **Journal of Hydraulic Research**, Taylor & Francis, v. 52, n. 1, p. 105–120, 2014. DOI: 10.1080/00221686.2014.880859. eprint: <https://doi.org/10.1080/00221686.2014.880859>. Disponível em: <<https://doi.org/10.1080/00221686.2014.880859>>. Citado nas pp. 3, 10, 16, 25, 44.
- SZLAUER-ŁUKASZEWSKA, Agnieszka. Substrate type as a factor affecting the ostracod assemblages in groyne fields of the Oder River (Poland). **North-Western Journal of Zoology**, v. 11, n. 2, p. 274–287, 2015. ISSN 18435629. Citado na p. 10.
- UIJTTEWAAL, Wim S. Effects of Groyne Layout on the Flow in Groyne Fields: Laboratory Experiments. **Journal of Hydraulic Engineering**, v. 131, n. 9, p. 782–791, 2005. DOI: 10.1061/(asce)0733-9429(2005)131:9(782). eprint: <https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9429%282005%29131%3A9%28782%29>. Disponível em: <<https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9429%282005%29131%3A9%28782%29>>. Citado nas pp. 3, 10, 32.
- WEITBRECHT, V.; JIRKA, G. Flow Patterns In Dead Zones of Rivers and their Effect On Exchange Processes. In: citado nas pp. 3, 4, 11, 15, 40.
- WEITBRECHT, Volker. **Influence of dead-water zones on the dispersive mass-transport in rivers**. 2004. f. 130. 130 f. Tese (Doutorado) – Karlsruhe Institute of Technology, Karlsruhe. ISBN 3-937300-07-4. DOI: 10.5445/ksp/1242004. Disponível em: <<https://publikationen.bibliothek.kit.edu/1242004>>. Citado nas pp. 1–3, 5, 11, 15, 40, 48, 58, 60.
- XIANG, Ke; YANG, Zhonghua; HUAI, Wenxin et al. Large eddy simulation of turbulent flow structure in a rectangular embayment zone with different population densities of vegetation. **Environmental Science and Pollution Research**, Environmental Science e Pollution Research, v. 26, n. 14, p. 14583–14597, mai. 2019. ISSN 1614-7499. DOI: 10.1007/s11356-019-04709-x. Disponível em: <<https://doi.org/10.1007/s11356-019-04709-x>>. Citado nas pp. 6, 10, 23–25, 29–32, 39–43, 45, 55, 58.

# Chapter 3

## Hydrodynamics of Vegetated Lateral Cavities

In this chapter, the second topic of the dissertation is presented as a published conference paper. This paper was originally written in Portuguese and was translated for this dissertation. The objective of this paper was to develop a simple numerical method capable of estimating the flow and the mass exchange between a lateral cavity and the main channel.

The original paper was published in the '17º Congresso Nacional do Meio Ambiente', on September 24th 2020, Poços de Caldas, Brazil.

## Authors

- Luiz Eduardo Domingos de Oliveira <sup>1</sup>
- Taís Natsumi Yamasaki <sup>1</sup>
- Felipe Rezende da Costa <sup>1</sup>
- Johannes Gérson Janzen <sup>1</sup>
- Carlo Gualtieri <sup>2</sup>

---

<sup>1</sup>Federal University of Mato Grosso do Sul

<sup>2</sup>University of Naples Federico II

# Abstract

In rivers and channels, dead zones are regions of low velocity with the presence of recirculation motion, that have important ecological functions (eg. sediment retention) and also can be formed from human-made structures (eg. transversal dikes). The presence of vegetation in dead zones is a new topic of this discussion opening its way in the vegetated flow area of study, as the vegetation has the potential of changing the flow and altering mass exchange processes with the main channel. This study aimed to develop a numerical model of a single vegetated lateral cavity using Computational Fluid Dynamics (CFD). The vegetation drag was represented by a porous media which coefficients were calculated from experimental data. The results of the model shown that the cavity had a single vortex system in its interior and the flow velocity varied from  $-0.11$  to  $0.24\text{cm/s}$ . The simulation adapted well to the experimental data, which proved that the porous media is a suitable method of representing the vegetation drag in CFD.

**Keywords:** Lateral Cavities; Vegetation; Computational Fluid Dynamics (CFD).

## 3.1 Introduction

Rivers are formed by complex morphological boundaries, that create a variety of regions of high or low flows. One of these regions is named dead zone, in which slow velocities occur when compared to the main channel. The dead zones can occur naturally through lateral cavities (JACKSON; HAGGERTY; APTE, 2013) or in man-made structures, groyne fields (SUKHODOLOV; SUKHODOLOVA; KRICK, 2017) and transversal dikes (PANDEY; AHMAD; SHARMA, 2018). From the environmental point of view, dead zones function a 'foam' that absorbs part of the energy from the flow, which causes it to favour the retention of sediments, the protection of the margins and creates a habitat for biota that depends on slow waters (WEITBRECHT; SOCOLOFSKY; JIRKA, 2008).

In the field of vegetated flows, in which the main objective is to study the hydrodynamics between the flow and the vegetation, the research of vegetated dead zones is still recent. The presence of vegetation in the dead zone offers an additional drag to the flow, further impacting the velocity patterns in the region. Henceforth, the mass exchange processes between the main channel and the dead zone are also altered (XIANG; YANG; HUAI et al., 2019). This enlightens the importance of understanding the relationships, so it could be better used aiming to benefit the surrounding ecosystem. This study aims to simulate through Computational Fluid Dynamics (CFD) a vegetated lateral cavity using

a porous media to represent the vegetation.

### 3.2 Methods

The chosen geometry consisted of a part of a channel and a lateral cavity (Figure 3.1) based in (XIANG; YANG; HUAI et al., 2019). The depth of the flow and the channel ( $H$ ) was defined in  $0.10\text{ m}$  and the channel width ( $B$ ) in  $0.30\text{ m}$ . The cavity was  $L = 0.25\text{ m}$  long and  $W = 0.15\text{ m}$  wide. The mean velocity was kept constant as  $U = 0.101\text{ m/s}$ , which corresponded to a Reynolds number of  $Re = 9000$  (turbulent flow). The water was kept at a constant temperature of  $T = 293\text{ K}$ .

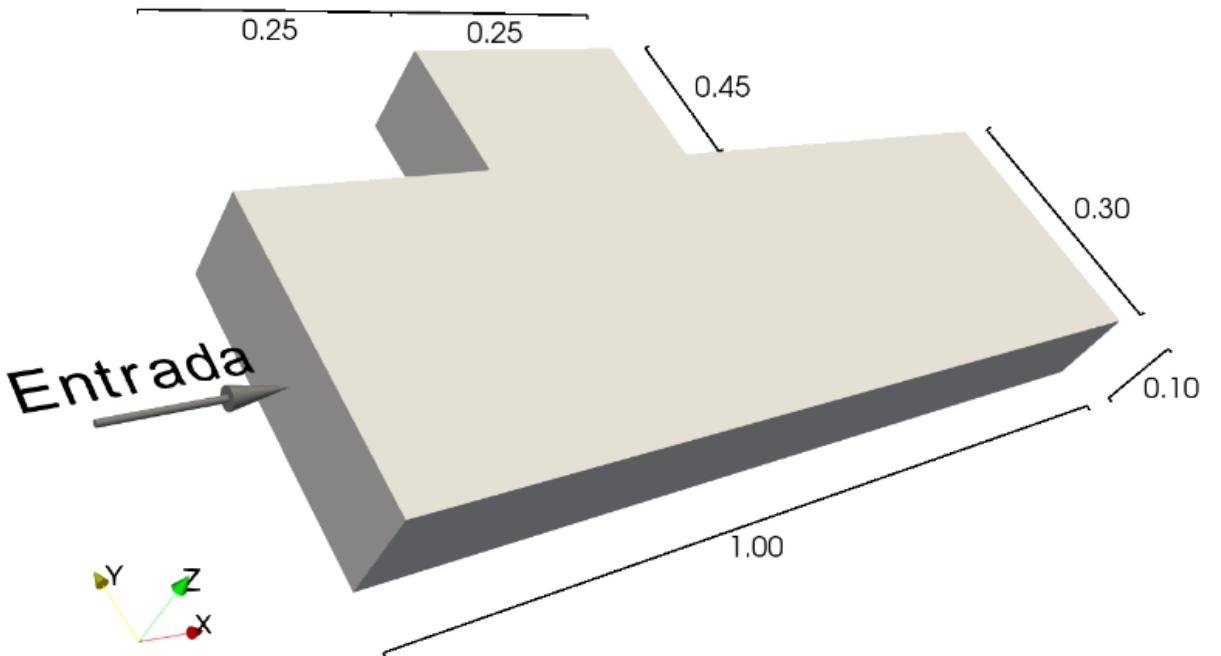


Figure 3.1: Computational domain. The flow direction is indicated by the grey arrow ('Entrada'). The dimensions are in metres and the coordinate origin ( $x, y, z = 0$ ) at the lower right portion of the channel.

The used boundary conditions were a longitudinal plane that cuts the domain at  $y = 0$  and at the top of the domain ( $z = 0.10\text{ m}$ ) that were defined as slip surfaces. The planes that cut the left portion of the channel and the cavity walls were considered hydraulic smooth walls of zero velocity. The channel entrance ( $x = 0\text{ m}$ ) imported a velocity profile previously simulated in a periodic channel. The outlet surface ( $x = 1.00\text{ m}$ ) was calculated with a zero gradient function.

The vegetation was represented with a porous media that filled all the lateral cavity. This is a simple way to represent the vegetation drag, and yet being an effective method of capturing the hydrodynamic effects (YAMASAKI et al., 2019). The adopted

porous media was calculated by the Darcy-Forchheimer model (DF), that divides the drag into viscous and inertial resistances. The coefficients were calculated using the Ergun formulation and Sonnenwald, Guymer e Stovin (2017), the vegetation parameters used to calculate the coefficients for the DF were taken from the second case of Xiang, Yang, Huai et al. (2019). The details of the used methods can be found in the user's guide of *Fluent*<sup>®</sup>.

The numerical model was simulated under the commercial software *Fluent*<sup>®</sup> (version 14), using the method of finite volumes to discretise the governing equations of mass conservation and momentum. The turbulence model applied was the Detached Eddy Simulation, with the contour model using the k-omega Shear Stress Transport. The simulation ran under a transient configuration for 350 seconds, that was enough time to stabilise the flow.

### 3.3 Results and Discussion

As expected, the flow inside the cavity became slower when compared to the main channel, the x component of the velocity varied between  $0.11$  and  $0.25U$  (Figure 3.2a). The high-velocity gradient in the cavity entrance ( $y = 0.30\text{ m}$ ) formed a shear layer that originated the vortexes. These vortexes were carried inside the cavity where a single circulation system, concentrated to the right portion, occurred as the streamlines in Figure 3.2a indicates. The adjusted drag coefficients were:  $83.37\text{ m}^2$  for the viscous resistance and  $3.79\text{ m}^{-1}$  for the inertial resistance.

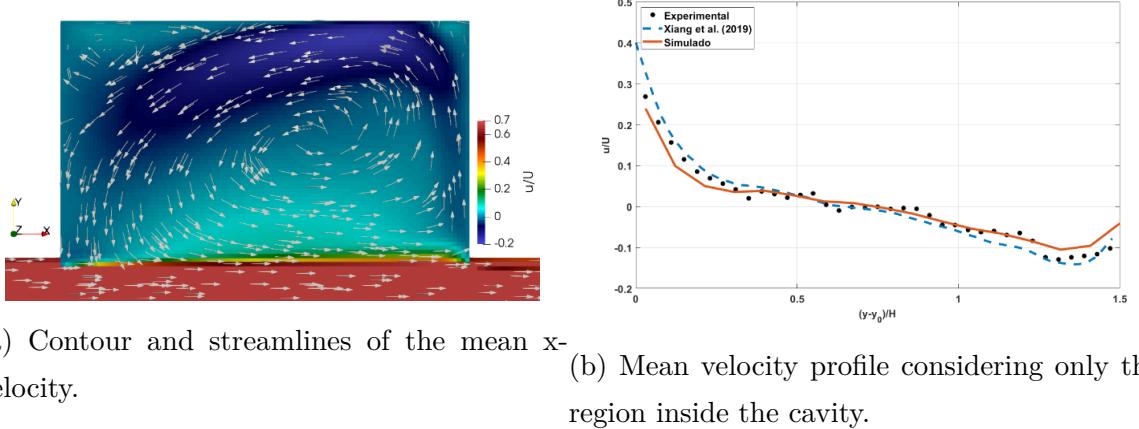


Figure 3.2: Velocity distributions along the plane  $z = 0.60H$ .

The y-velocity data was extracted along the plane  $z = 0.6H$  and condensed in a line using an ensemble averaging procedure along the y-axis similar to the procedure adopted in Sukhodolov (2014). The velocity profile is shown and compared with numerical

and experimental data extracted from the literature in the Figure 3.2b. At the entrance of the cavity ( $(y - y_0)/H = 0$ ), the velocity  $u$  was  $0.25U$  and it kept getting slower as it moved towards the interior of the cavity. In  $(y - y_0)/H = 1.4$ , the flow got a negative value of  $u = -0.1U$ , indicating the presence of vortexes. The presented model, in orange, was well adjusted to the experimental data (black dots). This means that the porous media coefficients were well calculated and the model was capable of reproducing the flow in accordance to the laboratory experiments.

### 3.4 Conclusion

The porous media model was capable of representing the vegetation in the numerical simulation. The cavity presented a single circulation system with a slower velocity than the main channel. The velocity profile obtained from the simulation was well adjusted to the experimental data, which further demonstrates that the model was capable of capturing the effects of vegetation inside the cavity.

### Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) -Finance Code 001.

### References

- JACKSON, T. R.; HAGGERTY, R.; APTE, S. V. A fluid-mechanics based classification scheme for surface transient storage in riverine environments: Quantitatively separating surface from hyporheic transient storage. **Hydrology and Earth System Sciences**, v. 17, n. 7, p. 2747–2779, 2013. ISSN 10275606. DOI: 10.5194/hess-17-2747-2013. Citado nas pp. 2–4, 23, 39.
- PANDEY, Manish; AHMAD, Z.; SHARMA, P. K. **Scour around impermeable spur dikes: a review**. [S.l.: s.n.], 2018. DOI: 10.1080/09715010.2017.1342571. Citado na p. 23.
- SONNENWALD, Fred; GUYMER, Ian; STOVIN, Virginia. Computational fluid dynamics modelling of residence times in vegetated stormwater ponds. **Proceedings of the Institution of Civil Engineers - Water Management**, v. 171, n. 2, p. 76–86, 2017. ISSN 1741-7589. DOI: 10.1680/jwama.16.00117. Citado na p. 25.

- SUKHODOLOV, Alexander N. Hydrodynamics of groyne fields in a straight river reach: insight from field experiments. **Journal of Hydraulic Research**, Taylor & Francis, v. 52, n. 1, p. 105–120, 2014. DOI: 10.1080/00221686.2014.880859. eprint: <https://doi.org/10.1080/00221686.2014.880859>. Disponível em: <<https://doi.org/10.1080/00221686.2014.880859>>. Citado nas pp. 3, 10, 16, 25, 44.
- SUKHODOLOV, Alexander N.; SUKHODOLOVA, Tatiana A.; KRICK, Julian. Effects of vegetation on turbulent flow structure in groyne fields. **Journal of Hydraulic Research**, v. 55, n. 1, p. 1–15, 2017. ISSN 00221686. DOI: 10.1080/00221686.2016.1211183. Citado nas pp. 6, 23, 39–41, 60.
- WEITBRECHT, Volker; SOCOLOFSKY, Scott a.; JIRKA, Gerhard H. Experiments on Mass Exchange between Groin Fields and Main Stream in Rivers. **Journal of Hydraulic Engineering**, v. 134, n. 2, p. 173–183, 2008. ISSN 0733-9429. DOI: 10.1061/(asce)0733-9429(2008)134:2(173). Citado na p. 23.
- XIANG, Ke; YANG, Zhonghua; HUAI, Wenxin et al. Large eddy simulation of turbulent flow structure in a rectangular embayment zone with different population densities of vegetation. **Environmental Science and Pollution Research**, Environmental Science e Pollution Research, v. 26, n. 14, p. 14583–14597, mai. 2019. ISSN 1614-7499. DOI: 10.1007/s11356-019-04709-x. Disponível em: <<https://doi.org/10.1007/s11356-019-04709-x>>. Citado nas pp. 6, 10, 23–25, 29–32, 39–43, 45, 55, 58.
- YAMASAKI, Taís N. et al. From patch to channel scale: The evolution of emergent vegetation in a channel. **Advances in Water Resources**, 2019. ISSN 03091708. DOI: 10.1016/j.advwatres.2019.05.009. Citado na p. 24.

# Chapter 4

## Velocity Estimates in Vegetated Lateral Cavities

In this chapter, the second topic of the dissertation is further developed in a published conference paper. The objective of this paper was to develop a simple numerical method capable of estimating the flow and the mass exchange between a lateral cavity and the main channel. Differently of the previous chapter, this paper introduces an open source approach to the problem, making the model further accessible to the general public. Also, the numerical model was further developed to account the mass transfer between the regions.

The original paper was published in the 'XIII Encontro Nacional de Águas Urbanas', on October 2020, Porto Alegre, Brazil.

## Authors

- Luiz Eduardo Domingos de Oliveira <sup>1</sup>
- Taís Natsumi Yamasaki <sup>1</sup>
- Johannes Géron Janzen <sup>1</sup>
- Carlo Gualtieri <sup>2</sup>

---

<sup>1</sup>Federal University of Mato Grosso do Sul

<sup>2</sup>University of Naples Federico II

# Abstract

Lateral cavities are a type of transient storage zones that occur in riverine systems. They play an important role in mass transport processes, especially due to a higher residence time. In this study, a numerical simulation of flow past a lateral cavity with vegetation was performed to assess the impact of the vegetation on the cavity hydrodynamics. The vegetation drag was introduced in a simplified method, as it was modelled as an anisotropic porous medium. The model could reproduce the experimental results at a reduced computational cost and can be considered a study platform for future studies.

**Keywords:** Lateral Cavities; Vegetation; Computational Fluid Dynamics (CFD).

## 4.1 Introduction

In rivers, lateral cavities are regions laterally attached to the channel, where the dynamics of the flow are characterised by slow velocities, increased mass residence time and the presence of re-circulations (CHANG; CONSTANTINESCU; PARK, 2006). The exchange processes between the unaltered channel (main channel) and the cavity occur solely by an interface in which the mass and momentum relationships occur. Since there is an increased residence time within the cavity volume due to the lower velocity magnitudes, this region favours sedimentation processes and vegetation growth. The presence of vegetation in the cavity alters the hydrodynamics and the interface exchanges (XIANG; YANG; HUAI et al., 2019).

The importance of lateral cavities in ecosystems is significative. For instance, the reduced velocities and the recirculation promote higher rates of sediments deposition and organic matter (JUEZ, C. et al., 2018) and also creates a favourable lentic environment to fish populations (LANDWÜST, 2006). Furthermore, lateral cavities promote the temporary storage of nutrients and contaminants, eg. heavy metals (ARGERICH et al., 2011; XIANG; YANG; HUAI et al., 2019), what make this structure a viable place for absorption and treatment of these substances.

Since the vegetation occurs naturally in lateral cavities and that its presence alters the dynamics of the flow. In this study, we aimed to simulate numerically the flow in a single rectangular lateral cavity with the presence of emergent vegetation, to comprehend the effects of vegetation inside the lateral cavity.

## 4.2 Methods

The modelled geometry consists in a channel reach with a lateral cavity 4.1. The main channel had a length of  $L_{ch} = 1.25\text{m}$  (x-axis), a width of  $W_{ch} = 0.30\text{m}$  (y-axis) and depth of  $H_{ch} = 0.10\text{m}$  (z-axis). The lateral cavity had a length of  $L_{cv} = 0.25\text{m}$ , width of  $W_{cv} = 0.15\text{m}$  and depth of  $H_{cv} = 0.10\text{m}$ . These dimensions were based on the laboratory experiments of Xiang, Yang, Huai et al. (2019). The mean velocity at the main channel was  $U = 0.101\text{m/s}$ , which corresponds to a Reynolds number of 9000 (turbulent flow).

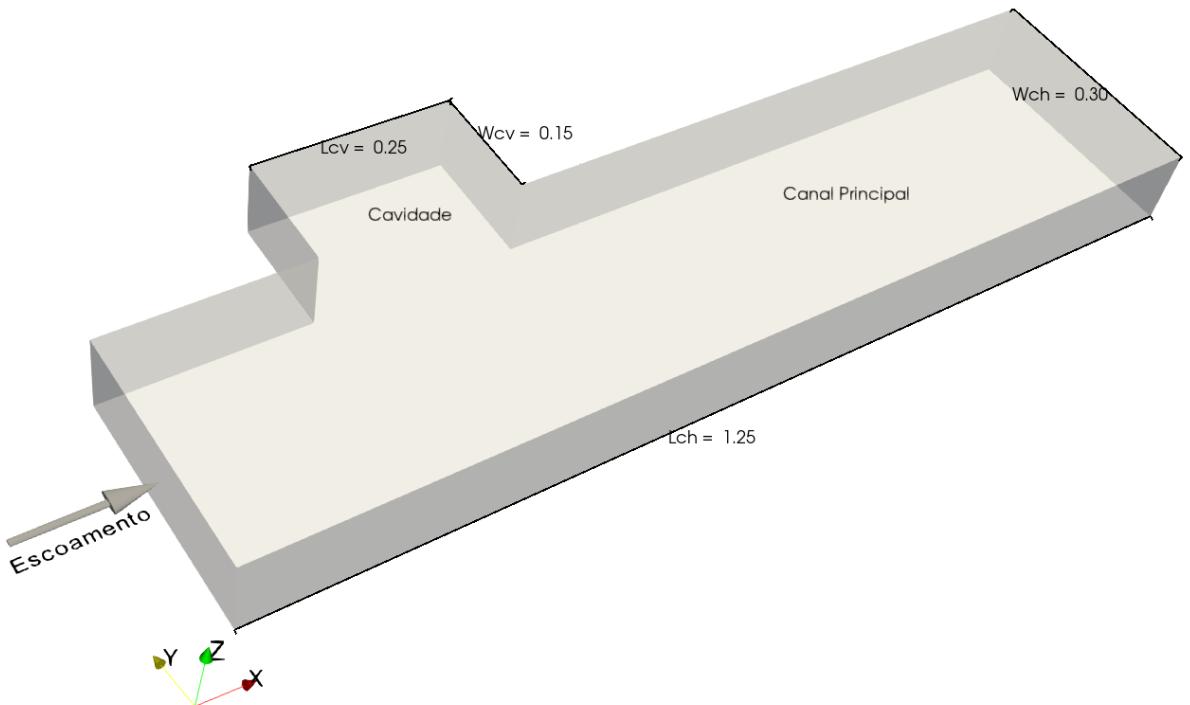


Figure 4.1: Numerical domain. The coordinate origins ( $x, y, z = 0$ ) is at the lower left corner of the picture. The inlet surface is at  $x = 0\text{m}$ , outlet at  $x = 0.75\text{m}$  and the cavity is between  $0.25 < x(\text{m}) < 0.50$ , connected to the channel.

The computational domain was calculated with the finite volume method and thus requires the discretisation of the geometry into a mesh. The geometry was divided into four blocks: cavity, upstream channel, downstream channel and middle channel. The mesh was made exclusively of orthogonal hexahedrons. The block within the cavity was divided into 80 divisions in both x and y directions, the elements close to the wall were refined to increase the accuracy of the model, the total growth rate was kept at a constant of 2. The entire domain was divided 40 times in the z-axis with a total growth rate, from the bottom, of 41. The mesh totalised in 1,408,000 elements.

At the free surface ( $z = 0.10\text{m}$ ) and at the cut surface ( $y = 0\text{m}$ ) the slip wall boundary condition was applied. The inlet surface ( $x = 0\text{m}$ ) was modelled through a pre-developed profile of velocities and reynolds stresses that were previously calculated

in a periodic flow separated from this main simulation. This data was mapped to feed the synthetic vortex boundary condition applied (*turbulentDFSEMInlet*). The outlet ( $x = 1.25\text{m}$ ) was treated as a zero gradient and all the other surfaces of the domain were treated as no-slip smooth walls. The wall function *nutUSpalldingWallFunction* was implemented in all walls to compute the variation of turbulence viscosity in the domain. Lastly, the model large eddy simulation (LES) was implemented, with a sub-grid filter wall-adapting local eddy-viscosity (WALE) to account the effects of turbulence in the channel.

The emergent vegetation inside the cavity was based in the second case of (XIANG; YANG; HUAI et al., 2019) study. The model used to describe the resistance caused by the vegetation was through a porous media calculated using the Darcy-Forchheimer equation, in which the inertial ( $f$ ) and viscous ( $d$ ) drag coefficients were calculated using the Ergun formulation in the x and y directions. The anisotropy caused by vegetation was considered in the z direction, where the drag coefficients were calculated using the method of (OLDHAM; STURMAN, 2001).

The open-source package OpenFOAM (version 1912) was used to calculate the computational model. The chosen calculation module of the pressure-velocity coupling was the PIMPLE, which uses both the transient formulation of the PISO with the permanent of SIMPLE. The numerical schemes chosen were of second-order to provide the necessary precision of LES. The time-steps were defined in an variable way assuring that the maximum Courant number was 0.90.

### 4.3 Results and Discussion

The mean velocities (time averaged) calculated from the model, had lower magnitudes than the main channel (4.2). A single circulation system was observed in the lateral cavity (4.3), as it was expected for aspect ratios between  $0.5 < W/L < 1.5$  (UI-JTTEWAAL; LEHMANN; MAZIJK, 2001). The origin of this circulation occurs in the momentum transfer from the main channel to the lateral cavity, as the flow occurs to the right, the circulation was in an anti-clockwise direction. At the upper left corner of the cavity an even higher reduction was observed, this occurs because of the path that the jet passes that starts at the inferior right region and follows it way deducting energy to the vegetation drag.

In energy exchange terms, the model was able to capture the interface between a cavity and the main channel (4.2), this could be visualised through the steady velocity

gradient that forms from the lower left corner of the cavity. From this point, the vortexes are dissipated and may enter the cavity or be ejected out to the main channel (4.4), similar to the process of multiple cavities inside the main channel (groynes) (UIJTEWAAL, 2005). Still in this figure, we could observe the difference in magnitude of the vectors, what reinforces the idea that the vegetated lateral cavities favour mass deposition due to its low velocities.

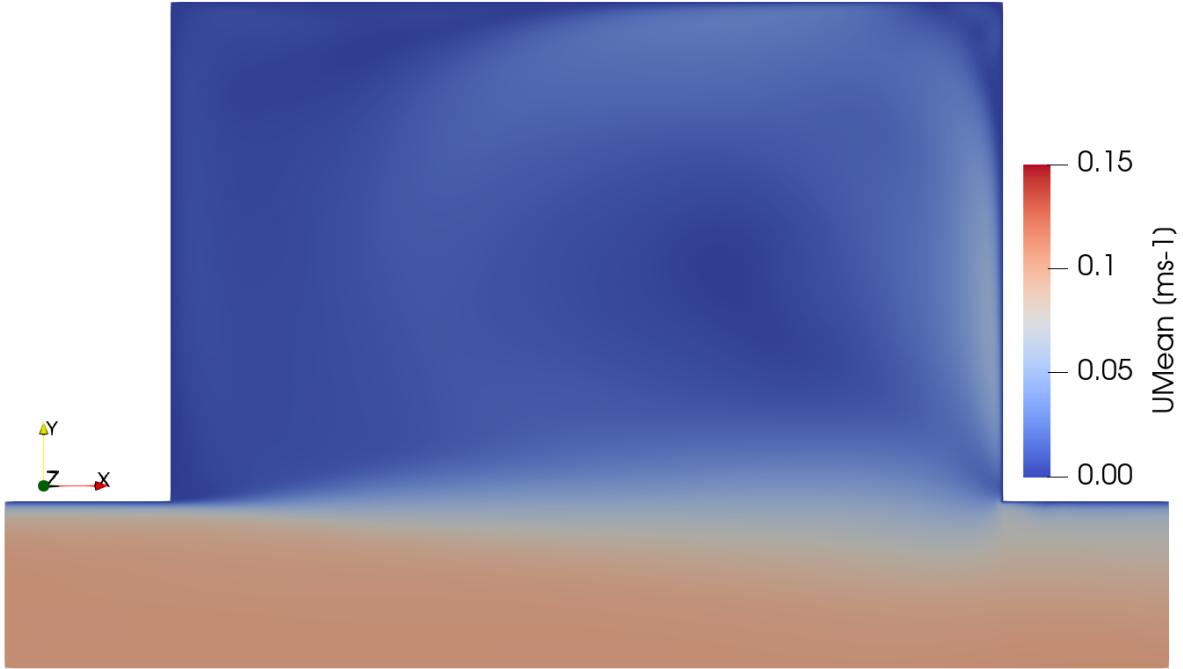


Figure 4.2: Mean velocity contour in the XY plane, in  $z = 0.6H$

The proposed model presented similar results when compared to numerical and experimental data. The ensemble average procedure was implemented to condense the values from the  $0.6H$  plane to a single line capable to describe the internal behaviour of the cavity (4.5), where  $y_0$  represents the beginning of the cavity. Notice that the model well predicts the flow except for the region close to  $(y - y_0)/H = 1.5$ , this occurs due to the size of the computational cells in the region, a further refinement in this region could decrease the difference to experimental values. Although, it is important to highlight that the model obtained a high precision taking in account the much lower number of elements in the grid ((XIANG; YANG; HUAI et al., 2019) model:  $1.5 \times 10^7$  elements; presented model:  $1.4 \times 10^6$  elements), what represents a faster execution and a less intensive computational usage. The anti-clockwise circulation tendency is confirmed by the velocity profile in the farthest region from the main channel that presented negative velocities and the close to the interface ( $0 < (y - y_0)/H < 0.6$ ), positive velocities. The circulation centre, region in which the velocity is zero were dislocated when compared to a lateral cavity without vegetation such as found in Gualtieri, López-Jiménez e Mora-Rodríguez (2010), in which the centre occurs at the cavity centroid. Although, in the vegetated case there

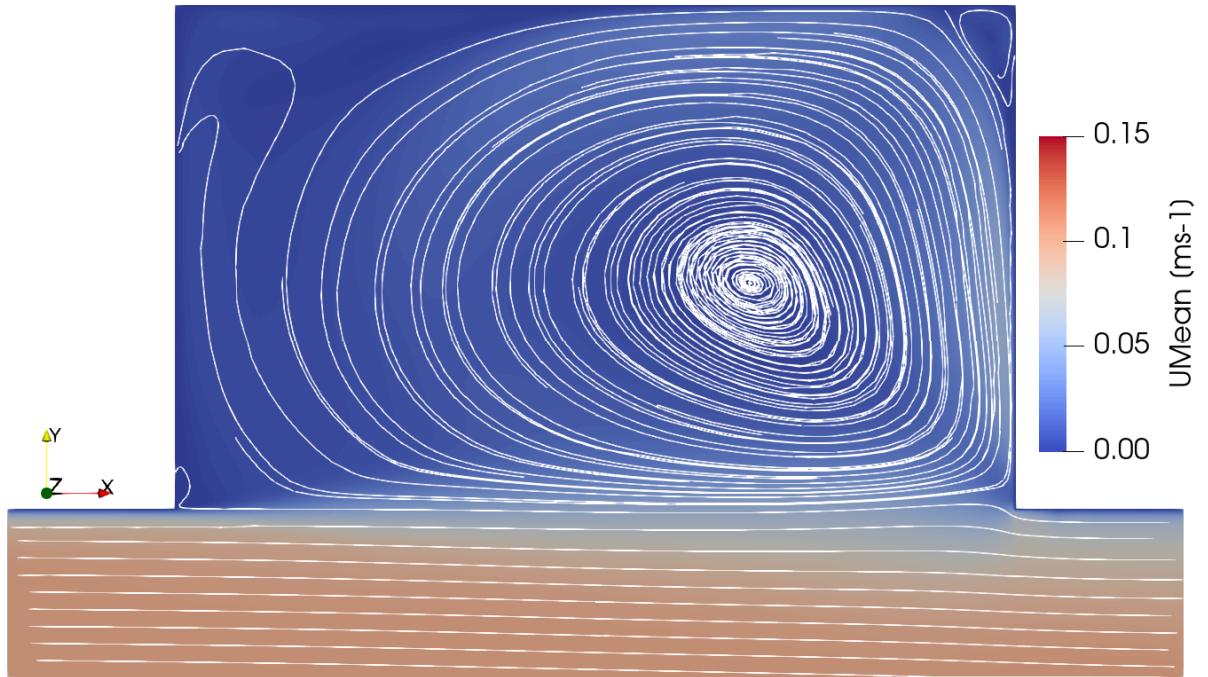


Figure 4.3: Mean velocity contour in the XY plane, in  $z = 0.6H$  with additional streamlines associated to the flow.

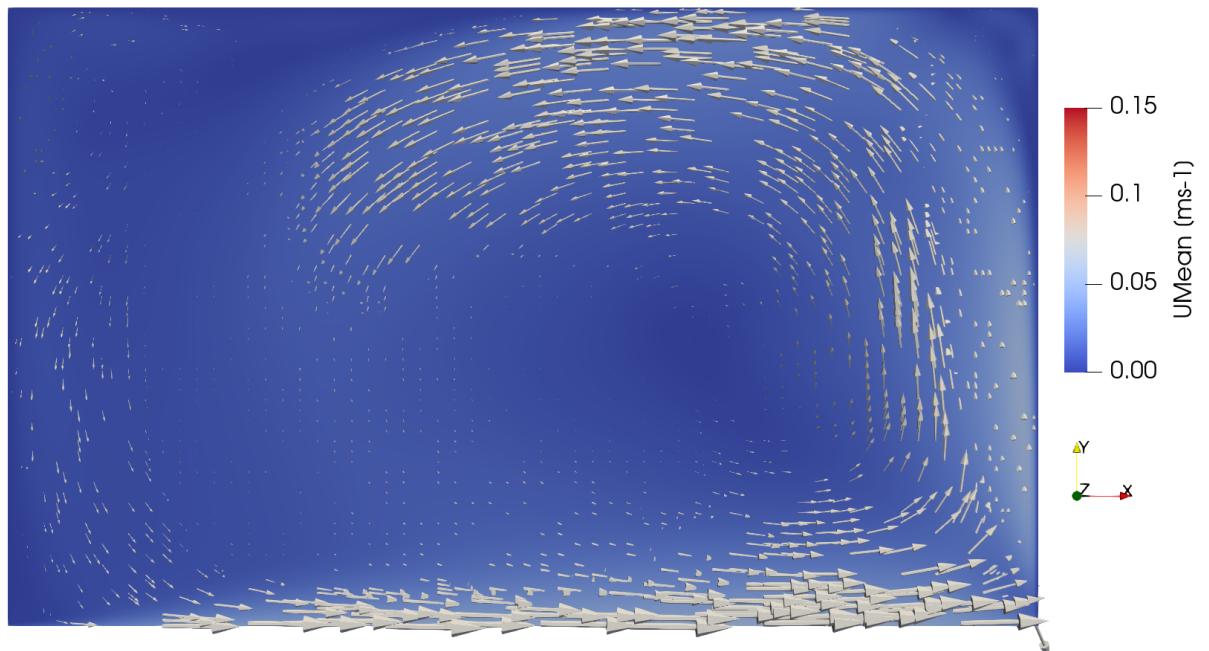


Figure 4.4: Mean velocity vectors in the XY plane, in  $z = 0.6H$

was a displacement to the right, that accords to the higher velocity magnitudes inside the cavity. Albeit there was a displacement to the right in the x-axis, there was none in the y-axis, that can be verified with the contours from 4.3 and with position of the velocities close to zero ( $0.6 < (y - y_0)/H < 0.82$ ).

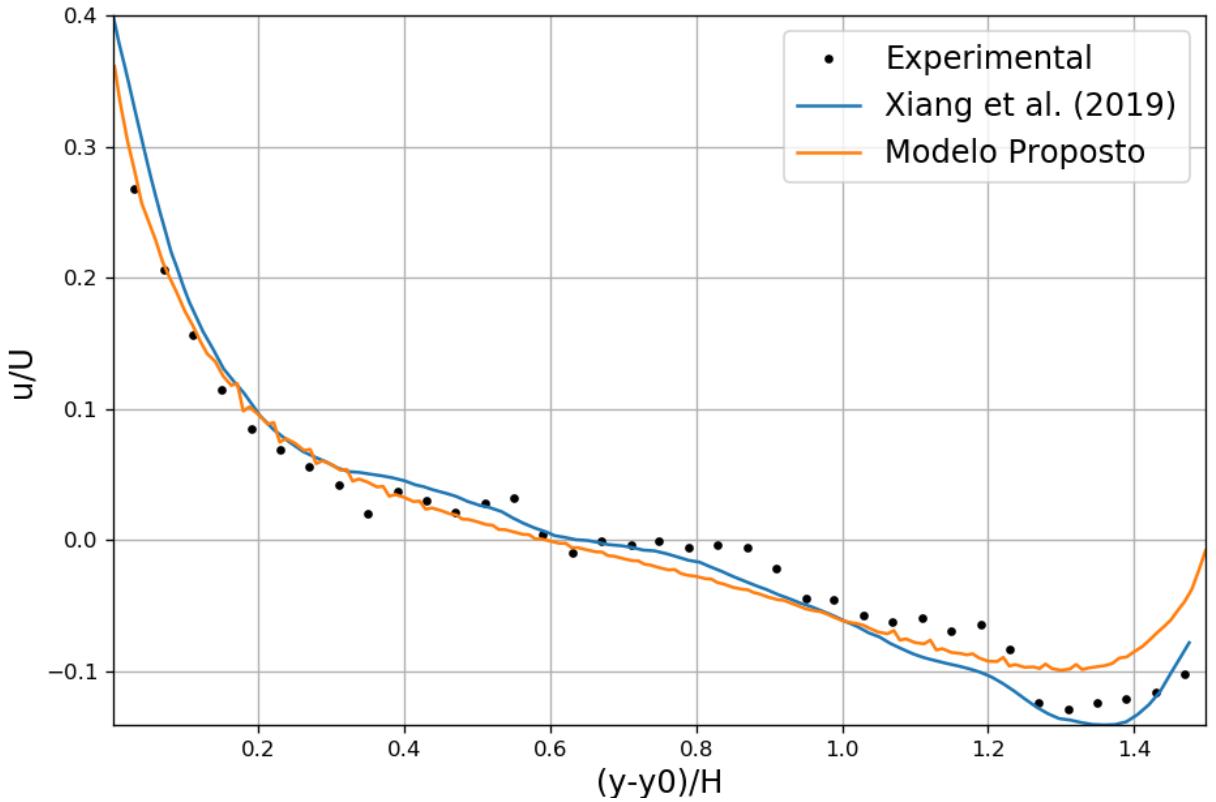


Figure 4.5: Comparison of the ensembled averaged mean velocity  $u$  in the XY plane, in  $z = 0.6H$

## 4.4 Conclusion

The numerical model of a vegetated lateral cavity presented a good accuracy when compared to experimental data from literature. The method of anisotropic porous media can be considered an effective approach to reproduce the qualitative and quantitative aspects of the model at a lower computational cost when compared to conventional techniques. This validated mode, can represent a new way to study cavities and be the basis of more detailed investigations.

## Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) -Finance Code 001.

## References

- ARGERICH, Alba et al. Influence of transient storage on stream nutrient uptake based on substrata manipulation. **Aquatic Sciences**, v. 73, n. 3, p. 365–376, 2011. ISSN 1420-9055. DOI: 10.1007/s00027-011-0184-9. Disponível em: <<https://doi.org/10.1007/s00027-011-0184-9>>. Citado na p. 29.
- CHANG, Kyoungsik; CONSTANTINESCU, George; PARK, Seung-O. Analysis of the flow and mass transfer processes for the incompressible flow past an open cavity with a laminar and a fully turbulent incoming boundary layer. **Journal of Fluid Mechanics**, Cambridge University Press, v. 561, p. 113, ago. 2006. ISSN 0022-1120. DOI: 10.1017/s0022112006000735. Disponível em: <<http://www.journals.cambridge.org/abstract%7B%5C%5F%7DS0022112006000735>>. Citado na p. 29.
- GUALTIERI, Carlo; LÓPEZ-JIMÉNEZ, P.; MORA-RODRÍGUEZ, Jesús. Modelling turbulence and solute transport in a square dead zone. In: citado na p. 32.
- JUEZ, C. et al. Morphological resilience to flow fluctuations of fine sediment deposits in bank lateral cavities. **Advances in Water Resources**, 2018. ISSN 03091708. DOI: 10.1016/j.advwatres.2018.03.004. Citado nas pp. 3, 29.
- LANDWÜST, Christian. Expansion of Proterorhinus marmoratus (Teleostei , Gobiidae) into the River Moselle (Germany). **Folia Zoologica**, v. 55, n. 1, p. 107–111, 2006. Citado na p. 29.
- OLDHAM, C. E.; STURMAN, J. J. The effect of emergent vegetation on convective flushing in shallow wetlands: Scaling and experiments. **Limnology and Oceanography**, v. 46, n. 6, p. 1486–1493, 2001. DOI: 10.4319/lo.2001.46.6.1486. eprint: <https://aslopubs.onlinelibrary.wiley.com/doi/pdf/10.4319/lo.2001.46.6.1486>. Disponível em: <<https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lo.2001.46.6.1486>>. Citado nas pp. 31, 42.
- UIJTTEWAAL, By W S J; LEHMANN, D; MAZIJK, a Van. Exchange Processes Between a River and Model Experiments Its Groyne Fields : Model Experiments. **Journal of Hydraulic Engineering**, v. 127, November, p. 928–936, 2001. Citado nas pp. 31, 40, 43.
- UIJTTEWAAL, Wim S. Effects of Groyne Layout on the Flow in Groyne Fields: Laboratory Experiments. **Journal of Hydraulic Engineering**, v. 131, n. 9, p. 782–791, 2005. DOI: 10.1061/(asce)0733-9429(2005)131:9(782). eprint: <https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9429%282005%29131%3A9%28782%29>. Disponível em: <<https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9429%282005%29131%3A9%28782%29>>. Citado nas pp. 3, 10, 32.

XIANG, Ke; YANG, Zhonghua; HUAI, Wenxin et al. Large eddy simulation of turbulent flow structure in a rectangular embayment zone with different population densities of vegetation. **Environmental Science and Pollution Research**, Environmental Science e Pollution Research, v. 26, n. 14, p. 14583–14597, mai. 2019. ISSN 1614-7499. DOI: 10.1007/s11356-019-04709-x. Disponível em: <<https://doi.org/10.1007/s11356-019-04709-x>>. Citado nas pp. 6, 10, 23–25, 29–32, 39–43, 45, 55, 58.

# Chapter 5

## The Effects of Vegetation Density Upon Flow and Mass Transport in Lateral Cavities

In this chapter, the main topic of this dissertation is developed. The effects of vegetation on the hydrodynamics and the mass exchange between the main channel/dead zone are investigated. The objective of this paper was to describe and possibly find a threshold on the behaviour of the dead zone given a certain density level.

### Authors

- Luiz Eduardo Domingos de Oliveira <sup>1</sup>
- Taís Natsumi Yamasaki <sup>1</sup>
- Filipe de Lima Queiroz <sup>1</sup>
- Felipe Rezende da Costa <sup>1</sup>
- Johannes Gérson Janzen <sup>1</sup>
- Carlo Gualtieri <sup>2</sup>

---

<sup>1</sup>Federal University of Mato Grosso do Sul

<sup>2</sup>University of Naples Federico II

# Abstract

Lateral cavities are regions attached to rivers affect the flow by creating a dead water zone. These regions reduce the flow velocity increasing the deposition of sediment and the temporary storage of polluted materials, which favours the growth of aquatic vegetation. The effect of this vegetation growth was studied using different vegetation densities in a Large Eddy Simulation (LES). The vegetation drag was represented by a porous media calculated with the Darcy-Forchheimer model. This numerical model showed that the hydrodynamics of the flow can present different patterns and phases for a vegetation density  $0 < a(\%) < 10.656$ . Furthermore, the occurrence of a secondary circulation was found where it normally would not occur for a non-vegetated scenario.

**Keywords:** Lateral Cavity; Aquatic Vegetation; Mass Exchange; Computational Fluid Dynamics (CFD).

## 5.1 Introduction

Lateral cavities are an important component of riverine (HARVEY; GOOSEFF, 2015) and estuarine (WARD; MICHAEL KEMP; BOYNTON, 1984) systems, because they (1) function as a macro-roughness at the river banks (JUEZ, Carmelo et al., 2017), (2) drive mass exchange processes with the open channel (OURO; JUEZ; FRANCA, 2020; MIGNOT et al., 2017; JACKSON; APTE et al., 2015), (3) act as transient storage zones (JACKSON; APTE et al., 2015; DROST et al., 2014; JACKSON; HAGGERTY; APTE; O'CONNOR, 2013), and (4) enhance biodiversity in the system (HARVEY, 2016; RIBI et al., 2014; WATTS; JOHNSON, 2004). These functions are linked to morphology-induced flow heterogeneity (JACKSON; HAGGERTY; APTE; O'CONNOR, 2013; SAN-JOU; AKIMOTO; OKAMOTO, 2012; MEILE; BOILLAT; SCHLEISS, 2011). Drawing on studies that demonstrated the relevance of transient storage zones on nutrient retention and cycling (ENSIGN; DOYLE, 2005; MULHOLLAND et al., 1994), lateral cavities can play a role in these processes because of increased timescales of solutes, especially due to the formation of recirculation gyres (JACKSON; HAGGERTY; APTE; COLEMAN et al., 2012; GOOSEFF et al., 2005). In face of flushing events, mobilized sediment can be carried out of the cavities, which may pose a risk of releasing pollutants in the stream (FORREST et al., 2007).

In aquatic systems, the retention of fine sediments and nutrients constitutes a favourable substrate for vegetation establishment and growth (NEPF, 2012; VANDEN-BRUWAENE et al., 2011; ASAEDA et al., 2009; COTTON et al., 2006; BARKO; GUN-

NISON; CARPENTER, 1991), which occurs in lateral cavities and embayments (JONES, 2020; ELY; EVANS, 2010; OLESEN, 1996; WARD; MICHAEL KEMP; BOYNTON, 1984). Except to the case of invasive species (MACEINA; SLIPKE; GRIZZLE, 1999), vegetation serves to refuge and sustain fish communities (KRAUS; JONES, 2012; AREND; BAIN, 2008), trap suspended material (WARD; MICHAEL KEMP; BOYNTON, 1984) and protect from bank erosion (DURÓ et al., 2020), these two last features being associated with the ability of vegetation to dissipate flow energy. Consequently, vegetation canopies increase the retention time and are considered by some authors as transient storage zones by themselves (KURZ et al., 2017).

The hydrodynamics of vegetated cavities are mainly dependent on the incoming flow properties, cavity geometry and vegetation characteristics (XIANG; YANG; WU et al., 2020; XIANG; YANG; HUAI et al., 2019; LU; DAI, 2016; SUKHODOLOV; SUKHODOLOVA; KRICK, 2017). Xiang, Yang, Huai et al. (2019) showed that the degree of vegetation effects on the initial bare-bed cavity depends on the vegetation density. The authors tested five vegetation densities in a rectangular cavity, using Computational Fluid Dynamics (CFD). The immediate effect of increasing the density was a reduction in velocity magnitude and turbulence inside the cavity, which was caused by the flow resistance exerted by the vegetation. The interface connecting the cavity to the channel presented a mixing layer with higher turbulence and vorticity than the rest of the domain, as a consequence of von Karman vortex streets generated by the vegetation, combined with shedding vortices created at the entrance corner of the cavity. Further, secondary recirculation gyres in the cavity were suppressed by denser vegetation values.

Field-scale experiments performed by Sukhodolov, Sukhodolova e Krick (2017) at a vegetated groyne (a type of cavity that is built inside the open channel, according to Jackson, Haggerty e Apte (2013)), indicated that denser vegetation diffused more momentum from the jet coming at the groyne entrance, which modified the circulation patterns in the groyne. The experiments showed that vegetation imposed a single circulation in the groyne, similar to Xiang, Yang, Huai et al. (2019), but that vegetation had little effect on the mixing layer formed at the groyne-channel interface. Another difference between the two studies was that Sukhodolov, Sukhodolova e Krick (2017) found that the emergent vegetation induced uniform flow patterns along with the depth, whereas Xiang, Yang, Huai et al. (2019) indicated that the flow pattern specifically at the cavity interface changes with depth in the presence of vegetation. Moreover, Xiang, Yang, Wu et al. (2020) showed that vegetation blocked the development of the mixing layer spreading inside the groyne, which affects the exchange between the open channel and the cavity (denser vegetation blocks more flow) and increases the mean retention time of the flow in the cavity, for denser vegetation (XIANG; YANG; HUAI et al., 2019).

The studies with vegetated cavities, as described above, varied the vegetation density between 0 and 0.627% (XIANG; YANG; HUAI et al., 2019), 0 and 0.969% (XIANG; YANG; WU et al., 2020), and 1.57% (SUKHODOLOV; SUKHODOLOVA; KRICK, 2017). Experimentally, Xiang, Yang, Wu et al. (2020) mentioned the difficulty to test denser vegetation arrays in cavities because the array would block the laser light and, thus, compromise flow measurements. The authors expanded the density values using numerical simulations. However, a reference threshold for vegetation to be considered “dense” or “sparse” in cavities has not been defined to date, and it points to the need of understanding which density thresholds will cause key flow modifications in the cavity (e.g., the suppression of recirculation gyres, the complete suppression of flow, the exchange coefficient asymptote, etc.). For emergent vegetation patches in an open channel, Chen et al. (2012) characterized them as being “dense” or “sparse” according to flow blockage thresholds, in which the flow properties near the patch (e.g., flow adjustment length and the velocity exiting the patch) were distinct above and below the threshold. A similar approach can be done for vegetated cavities. Furthermore, in previous field and laboratory experiments (MIGNOT et al., 2017; CONSTANTINESCU; SUKHODOLOV; MCCOY, 2009; WEITBRECHT, 2004; WEITBRECHT; JIRKA, 2001; UIJTTEWAAL; LEHMANN; MAZIJK, 2001), the mass exchange between the main channel and a dead water zone, lateral cavity or groyne, was studied with the ejection of tracer fields. This method of analysing the transport of the passive scalar provides a different perspective of the physics of this exchange and should be further explored (XIANG; YANG; HUAI et al., 2019). Hence, a dynamic model that considers passive scalar motion can be an effective way to help river managers to predict pollutant transport in accidental spills.

The objective of the present study was to expand the vegetation density range and identify the thresholds that can differentiate dense to sparse vegetation in a lateral cavity. The study was performed with CFD simulations.

This paper is divided into five main sections. Following the Introduction, the details of the numerical model were described, along with the grid independence test and solution quality. Third, the hydrodynamic characteristics of the flow were presented. Fourth, the impact on the mixing layer is presented and discussed. Fifth, the impact of the vegetation in the mass exchange is discussed. Finally, the conclusive remarks about the influence of vegetation in a single circulation lateral cavity were presented.

## 5.2 Numerical Model

### 5.2.1 Model Equations

The simulations were performed with the Large Eddy Simulation (LES) model, which uses the spatial filtering of the incompressible Navier-Stokes equations to solve the fluid motion and turbulence. For an incompressible fluid, the equations of mass and momentum conservation are depicted as follow, respectively:

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \quad \{i = 1, 2, 3\} \quad (5.1)$$

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}(\bar{u}_i \bar{u}_j) = -\frac{1}{\rho} \frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j} [\mu(2\bar{S}_{ij}) - \tau_{ij}] + \bar{S}_{M,i} \quad (5.2)$$

in which the overbar indicates resolved quantities;  $\bar{u}_i$  (m/s) is the velocity component in the  $i$  direction ( $i = 1, 2, 3$  correspond to x, y, z-axis, respectively),  $\rho$  (kg/m<sup>3</sup>) is the fluid density,  $\bar{p}$  (N/m<sup>2</sup>) is the dynamic pressure,  $\mu$  (m<sup>2</sup>/s) is the kinematic viscosity,  $\bar{S}_{ij}$  (1/s) is the strain-rate tensor,  $\tau_{ij}$  (m<sup>2</sup>/s<sup>2</sup>) is the subgrid-scale stress, and  $\bar{S}_{M,i}$  is the sink term related to vegetation drag (m/s<sup>2</sup>).  $\bar{S}_{ij}$  and  $\tau_{ij}$  are given by:

$$\bar{S}_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \quad (5.3)$$

$$\tau_{ij} = \bar{u}_i \bar{u}_j - \bar{u}_j \bar{u}_i \quad (5.4)$$

Specifically,  $\tau_{ij}$  represents the effect of unresolved small-scale motion on the resolved flow, and is based on the eddy-viscosity assumption:

$$\tau_{ij} - \frac{1}{3} \tau_{kk} \delta_{ij} = \nu_t (2\bar{S})ij \quad (5.5)$$

where  $\nu_t$  (m<sup>2</sup>/s) is the eddy viscosity. In this study, the Wall-Adapting Local Eddy-viscosity (WALE) model, proposed by Nicoud e Ducros (1999), was chosen as the subgrid-scale model to calculate  $\nu_t$ .

Even for CFD, adding more rigid cylinders in the cavity in order to increase the density (XIANG; YANG; HUAI et al., 2019; XIANG; YANG; WU et al., 2020) results in a heavier mesh that requires greater computational processing to run the model. The vegetation is considered uniform as flows like in lateral cavities are subject to riparian plants and vegetation cover that can develop almost uniformly of the area (SUKHODOLOV; SUKHODOLOVA; KRICK, 2017). For these reasons, the present study proposed to use the Darcy-Forchheimer porous media approach to represent the vegetation, adjusting the resistance in the horizontal and vertical directions. The vegetation inside the cavity was

represented by a porous media, in which the momentum loss caused by the vegetation drag was computed through the Darcy-Forchheimer (DF) model (Equation 5.6).

$$S_{M,i} = \left( -\mu d + \frac{\rho |u_{jj}|}{2} f \right) u_i \quad (5.6)$$

in which  $d$  (1/m<sup>2</sup>) is the viscosity drag coefficient and  $f$  (1/m) is the inertial coefficient. First, the porous model was configured and validated with laboratory experiments performed by Xiang, Yang, Huai et al. (2019), who created a surrogate for rigid vegetation by displaying different arrays of copper wires for different vegetation density values in the cavity. Then, to expand the density range, simulations with higher density values were performed, assuming the same stem diameter of Xiang, Yang, Huai et al. (2019). The wire diameter was  $d_w = 0.15\text{cm}$ . The vegetation density,  $a$ , was calculated as follows:

$$a = \frac{nS_V}{S_{cav}} \quad (5.7)$$

in which  $n$  is the number of vegetation stems,  $S_V$  (m<sup>2</sup>) is the horizontal cross-section area of the stems, and  $S_{cav}$  (m<sup>2</sup>) is the cavity area. The coefficients  $d$  and  $f$  were calculated using the Ergun equation:

$$d = \frac{150}{D_p^2} \frac{(1-\epsilon)^2}{\epsilon^3} \quad (5.8)$$

$$f = \frac{3.5}{D_p} \frac{(1-\epsilon)}{\epsilon^3} \quad (5.9)$$

in which  $D_p$  (cm) is the mean particle diameter, and  $\epsilon (= 1-a)$  is the void fraction. In the horizontal direction (flow perpendicular to the stems, which corresponds to the  $x$ - and  $y$ -axis),  $D_p$  was assumed as the wire diameter ( $D_p = d_w$ ). To account for non-isotropic resistance, the approach of Oldham e Sturman (2001) was used to calculate  $d$  and  $f$  in the  $z$ -axis, where the flow is parallel to the stems. In this case,  $D_p$  was calculated as the hydraulic diameter  $d_h$  (cm):

$$d_h = d \left( \frac{4 \left( \frac{s}{d} \right)^2}{\pi} - 1 \right) \quad (5.10)$$

In which  $s/d$  is the spacing: diameter ratio between the wires.

### 5.2.2 Simulation Setup

The numerical model was developed based on the physical experiments of Xiang, Yang, Huai et al. (2019). The 3D geometry consisted of a single lateral cavity that was adjacent to a rectangular open channel Figure (5.1). The lateral cavity was  $W = 0.15$  m wide and  $L = 0.25$  m long, resulting in the aspect ratio  $W/L = 0.60$ , which falls in the range of  $0.5 \leq W/L \leq 0.15$  and thus corresponds to a one-gyre circulation system to be formed inside the cavity Uijttewaal, Lehmann e Mazijk (2001). The depth of the channel and cavity was  $H = 0.10$  m. The flow in the main channel was turbulent ( $Re = 9000$ ) and subcritical ( $Fr = 0.102$ ), with bulk velocity  $U = 0.101$  m/s at the channel inlet ( $x = 0$  m). The temperature was constant at  $T = 293K$ .

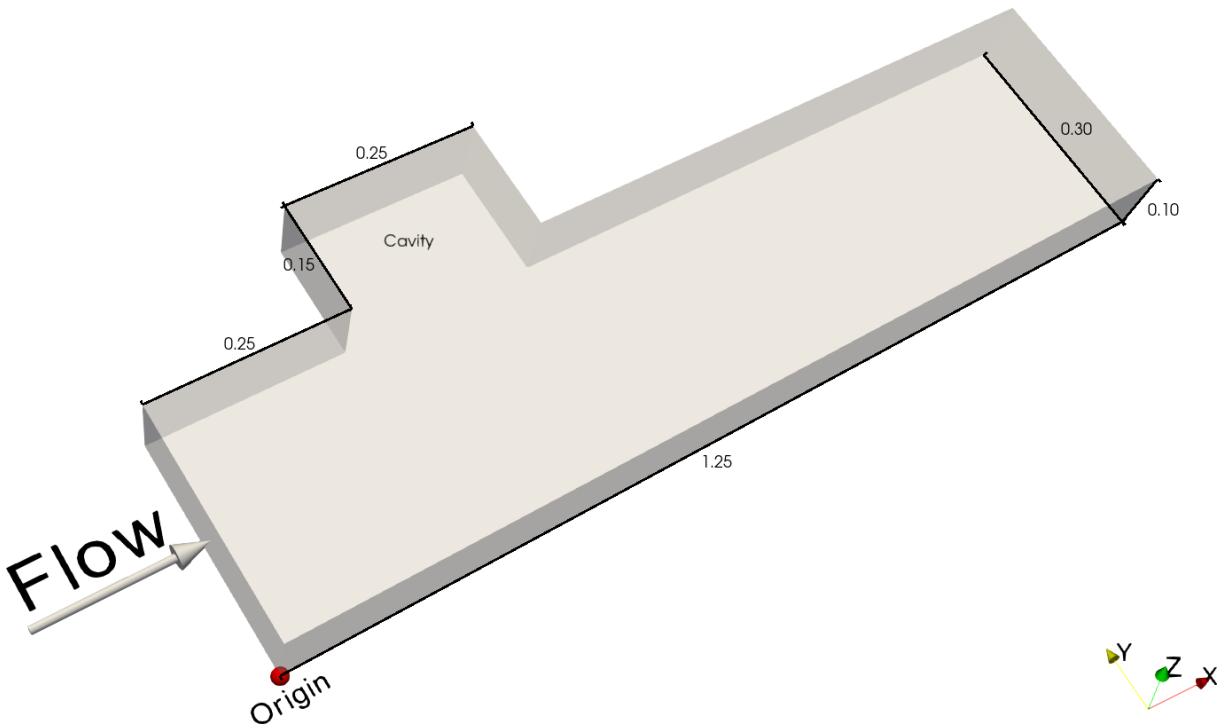


Figure 5.1: Computational domain with coordinates and dimensions.

The boundary conditions set to the model were the following. The rigid-lid approximation was applied at the free surface of the domain ( $z = 0.10$  m), which is valid for flows with  $Fr < 0.5$  (ALFRINK; RIJN, 1983). The longitudinal  $XZ$  plane, located at  $y = 0$  m, where the main channel was restricted in the domain, was defined as a free-slip surface. Knowing that flow effects caused by obstacles to the main channel do not exceed one obstacle length (BREVIS; GARCÍA-VILLALBA; NIÑO, 2014), and knowing that the cavity had  $L = 0.15$  m, we defined the width of the main channel to be  $B = 0.30$  m, which was sufficient to capture any flow effect in the main channel. The inlet portion of the domain ( $x = 0$  m) received precalculated velocity fields that were fully developed in a periodic channel, under the same flow conditions and the main channel geometry. The implementation of this boundary condition applied the turbulence Divergence-Free

Synthetic Eddy Method (DF-SEM) to synthesise eddies based on the turbulence developed of the imported flow (POLETTI; CRAFT; REVELL, 2013). A convective outflow boundary condition was adopted at the outlet ( $x = 1.25$  m), in which the zero-gradient condition allows the flow to exit the domain without having any backflow. The bottom of the domain ( $z = 0$  m) and the walls of the main channel ( $y = 0.30$  m) and the cavity were considered as no-slip surfaces.

The mass exchange between the main channel and the vegetated cavity was simulated with tracer fields, in which the washout procedure was implemented. After all the solution transients were eliminated, the lateral cavity was filled with an inert tracer. The flow was calculated until either all tracer left the cavity, or a time of 200 s passed. The associated turbulent Schmidt number was  $S_{ct} = 0.9$ , as used in other similar flows (GUALTIERI; ANGELOUDIS et al., 2017). In this period, the average flow was, also, calculated and condensed into an ensemble averaging (SUKHODOLOV, 2014). The computational time increment was held variable, with a Courant number kept under 0.90 and a maximum time step of 0.05 s.

The simulations were performed with the open-source package OpenFOAM (version 1912). To discretize the governing equations and numerical schemes, the module pimpleFoam, which employs the finite volume method (FVM), was used. For the pressure-velocity coupling, the PIMPLE method scheme was adopted. To solve the convection-diffusion equations, the implicit second-order backward time-stepping scheme and additional second-order schemes were used. The residual tolerance was set to  $1 \times 10^{-4}$  and the number of our loops was set to 3, the same count was set for the pressure correction loops.

### 5.2.3 Numerical Programme

The study of the vegetated cavity was proposed by varying the vegetation density values using different DF coefficients to emulate the increasing drag, which is summarised in Table 5.1. The density was varied between  $a = 0$  (no vegetation) and  $a = 10.656\%$ , distributed in ten scenarios for simulation. The vegetation density found in natural conditions varies from  $0.001 < a < 0.45$ , and the effects of the turbulence dissipation remains predominant until  $a < 0.1$  (NEPF, 2012), given that these values were based on a free open channel, we chose a smaller value of  $a$  that could comprehend all the turbulence dissipation spectrum as this is a key component of the hydrodynamics of dead waters. It was assumed that the vegetation was uniformly distributed in the cavity and that it spanned the cavity depth, similarly to emergent vegetation.

Case	a (%)	Horizontal direction (x and y-axis)		Vertical direction (z-axis)		
		$d$ (1/m <sup>2</sup> )	$f$ (1/m)	$dh$ (m)	$d$ (1/m <sup>2</sup> )	$f$ (1/m)
0	0	0.00	0.00	0.00	0.00	0.00
1	0.1332	116.53	3.09	0.7624	0.000451	0.00608
2	0.1665	182.25	3.87	0.8265	0.0006	0.00702
3	0.3330	753.83	7.89	0.3902	0.0111	0.0303
4	0.6660	3002.72	15.82	0.1846	0.198	0.19
5	1.3320	12344.01	32.40	0.0836	3.98	0.58
6	2.6640	51244.51	67.36	0.0360	88.96	2.81
7	3.9960	120314.00	105.38	0.0210	613.12	7.52
8	5.3280	223190.20	146.57	0.0139	2602.53	15.83
9	7.9920	546724.99	239.43	0.0072	23702.83	49.85
10	10.6560	1061150.94	348.58	0.0041	140829.09	126.99

Table 5.1: Vegetation levels and the calculated Darcy-Forchheimer coefficients, where  $a$  (%) is the vegetation density,  $d$  (1/m<sup>2</sup>) is the viscosity drag coefficient,  $f$  (1/m) is the inertial coefficient and  $dh$  (m) is the hydraulic diameter.

### 5.2.4 LES Quality and Grid Independence

The quality of the numerical solution was evaluated using a procedure based on three different grids (DUTTA; XING, 2018). The refinement rate between the grids was 1.80, although with the same configurations (numerical model and boundary conditions). The numerical and modelling errors were estimated and compared to the experimental data from Xiang, Yang, Huai et al. (2019). Figure 5.2 shows the ensemble-averaged streamwise velocity with the total error (numerical and modelled) expressed by error bars. Overall, the numerical solution presented low error magnitudes, with a mean total error of -0.0024 m/s. The errors could be mitigated by a further refinement, although the errors were small enough to continue the experiments.

### 5.2.5 Validation

Figure 5.2 compares the results of the time-averaged streamwise velocity  $u$  at  $z/H = 0.6$  obtained from the second case ( $a = 0.1332\%$ ) of Xiang, Yang, Huai et al. (2019), using both experimental and his numerical model. The numerical results, from the present paper, showed good consistency with the experimental data. A difference between the wall resolved LES (WRLES) and the wall modelled LES (WMLES) is highlighted in

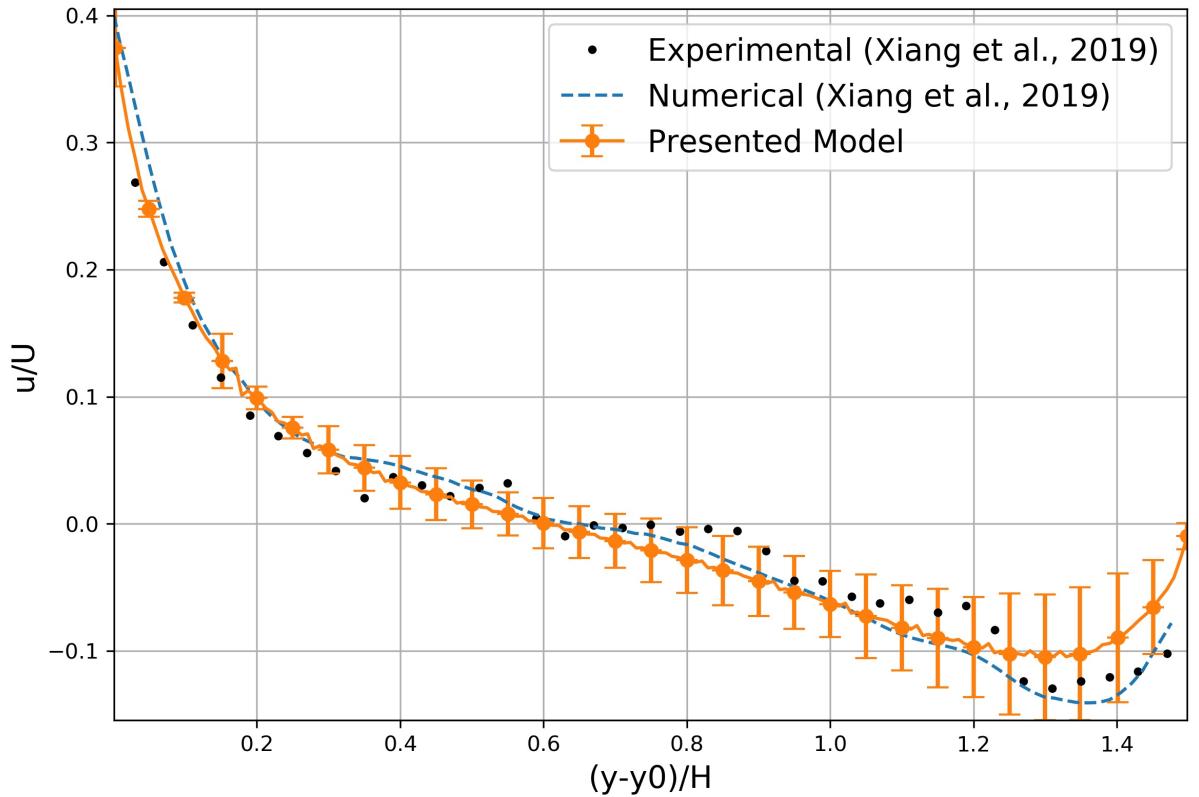


Figure 5.2: Grid and Numerical Errors of the ensemble-averaged streamwise velocity in the cavity at  $z = 0.6H$ , where  $U$  is the bulk velocity in the main channel,  $y_0 = 0.30\text{m}$  represents the beginning of the cavity and  $H$  is the height of the flow.

the region  $(y - y_0)/H > 1.20$ , where the continuous line deviated from the dashed result. Although, in all other regions the results followed closely both experimental and the WRLES.

### 5.3 Flow Characteristics

Figure 5.3 show the mean 2D streamlines for all the cases at  $z/H = 0.6$  inside the cavity volume as the principal phenomena occurs in this region. Under the cases 0 to 5 a main anti-clockwise motion takes place (Figure 5.3 a-f). The increase in vegetation density translates the centre of the gyre towards the main channel and downstream in the x-direction as the blockage effects increase and the flow loses energy faster. For  $a = 1.3320\%$  (case 5) the main circulation starts to lose its shape and this process continues up to  $a = 3.9960\%$  when the flow stabilised (Figure 5.3 f and Figure 5.3 g-h). The case 8 showed the formation of a secondary gyre system at the right portion of the cavity,  $0.45 < x/L < 1$  and  $0 < y/W < 1$  (Figure 5.3 i). This behaviour was shifted to the left as the vegetation increased to  $a = 7.9920\%$  (Case 9),  $0.30 < x/L < 1$  and  $0 < y/W < 1$

(Figure 5.3 i). The presence of secondary circulations normally occurs at different aspect ratios:  $W/L < 0.5$  and  $W/L > 1.5$  (SUKHODOLOV; UIJTEWAAL; ENGELHARDT, 2002), this circulation naturally does not have any contact with the main channel as they are derived from the primary circulation. Figure 5.3 i-k show the primary circulation at the bottom left of the cavity and the secondary gyre occupying approximately 50% of the area in  $a = 5.3280\%$ , the area comprehending the secondary gyre further increased with the vegetation drag increase.

Figure 5.4 show the flow at the horizontal plane  $XY$  at  $z/H = 0.6$  along the  $y$ -axis,  $(y - y_0)/H$  being  $y_0 = 0.30\text{m}$  the beginning of the cavity, where the velocity decreases as the vegetation density increases. Another important aspect of this figure is the positioning of the circulation centre that slowly shifts towards the region close to the interface ( $(y - y_0)/H = 0$ ) that is associated with the flow resistance imposed by the vegetation.

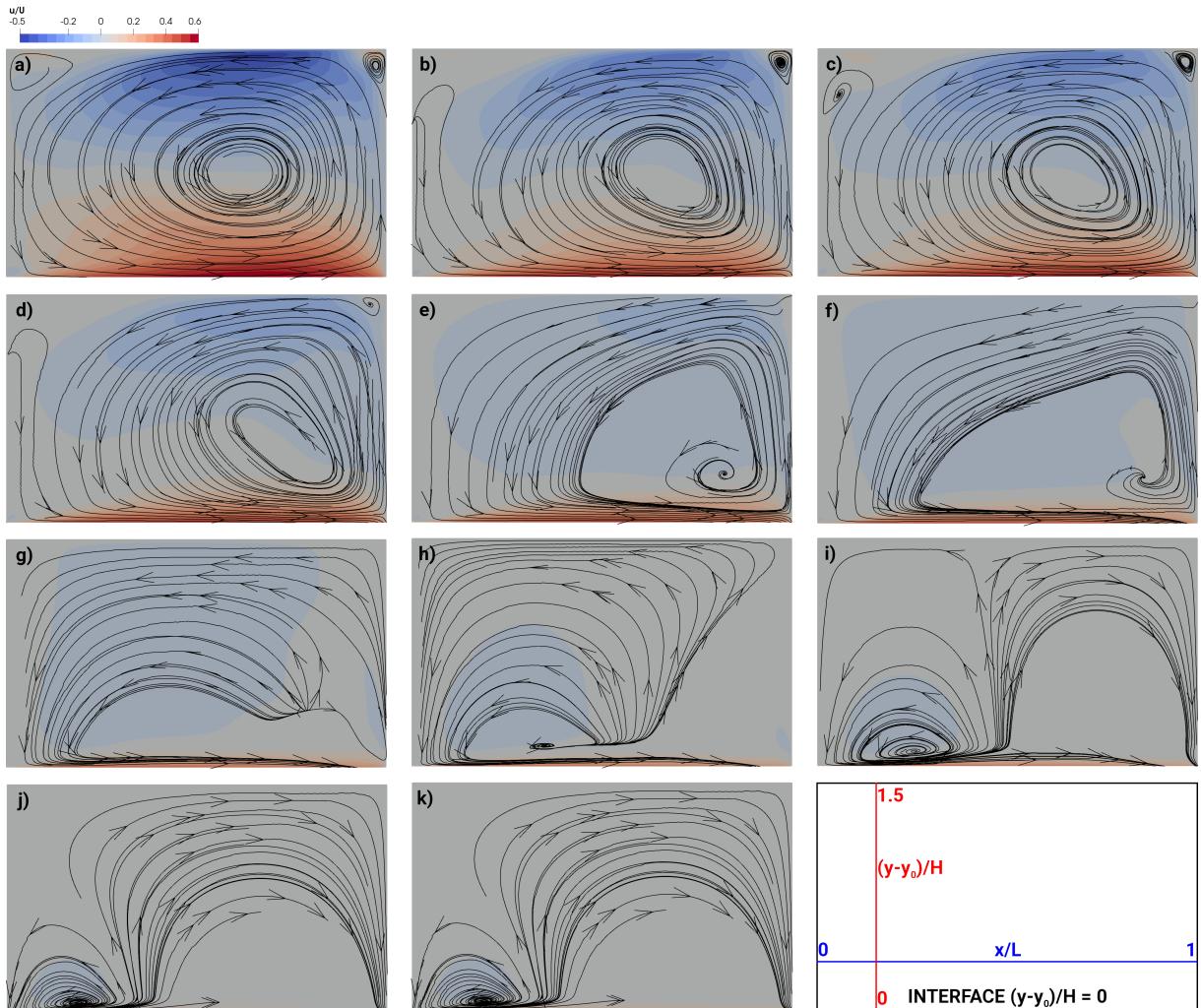


Figure 5.3: Mean 2D streamlines of different vegetation densities at the horizontal plane  $z/H = 0.6$  inside the cavity volume: a) Case 0, b) Case 1, c) Case 2, d) Case 3, e) Case 4, f) Case 5, g) Case 6, h) Case 7, i) Case 8, j) Case 9 and k) Case 10.

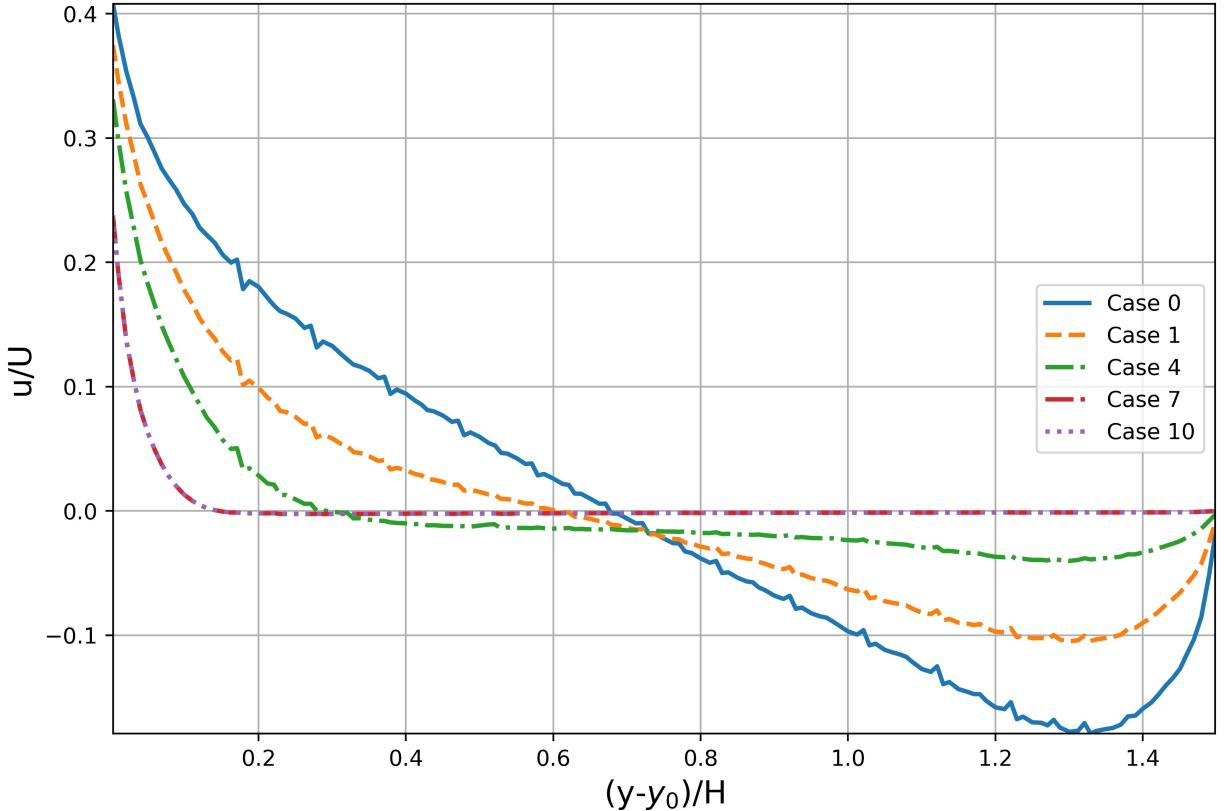


Figure 5.4: The variation of the streamwise velocity at the horizontal plane  $z/H = 0.6$  inside the cavity volume.

The flow through the interface was initially directed toward the cavity ( $z/H < 0.1$ ); positive velocity; then it was outwards ( $0.1 < z/H < 0.9$ ); negative velocity; and lastly entering the domain ( $z/H > 0.9$ ) (Figure 5.5). Through the variation in density, this behaviour did not change as the location of the phases did not change through all the cases, as seen in Figure 5.5, although the peak velocities at each phase gradually decreased as the vegetation density increased, which is attributable to the energy dissipation caused by vegetation. As the velocity values decreased the second phase ( $0.1 < z/H < 0.9$ ) tended to flat as the vegetation was tending to a solid block behaviour similar to the behaviour of vegetation in (CHEN et al., 2012). Furthermore, the initial peak in velocity disappeared for  $a > 5.32\%$  (Case 8) and was substituted by the increase of the third phase.

Figure 5.6 shows the behaviour of the interface along the  $x$ -axis. Analogous to the  $z$ -axis, the increase in vegetation density altered the velocity zones. When the vegetation was not present, Case 0, the profile initially was set to the main channel up to 50 % of the interface length similar to the behaviour of the series of groynes in Weitbrecht (2004). Although, with the increase of vegetation this first negative zone became positive and the only region where water exited the DZ volume was tending to  $(x - x_0)/L > 0.8$ , due to the shock of the vortices to the downstream wall of the cavity.

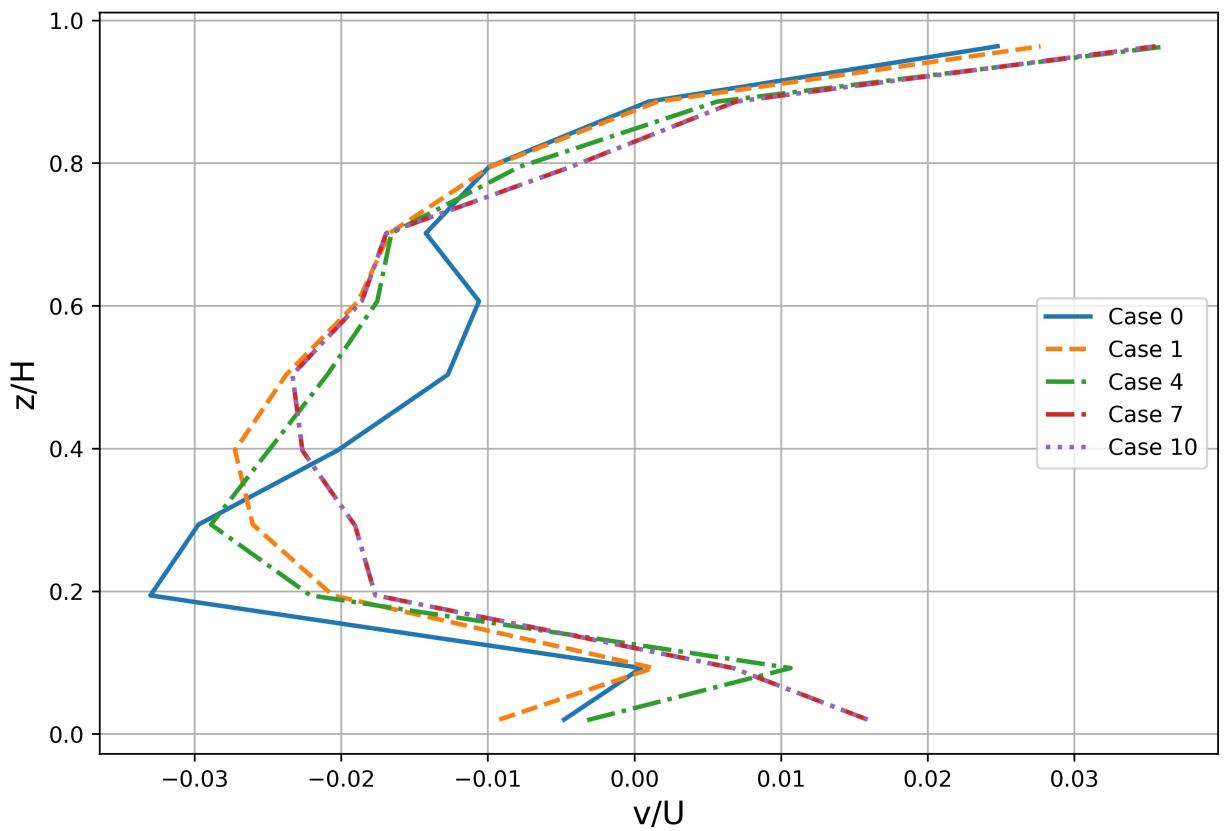


Figure 5.5: The variation of the transversal velocity in the interface between the cavity and the main channel along the z-axis: a) Cases from 0 to 5; b) Cases from 5 to 10. Positive values of  $v/U$  indicate the flow entering the cavity volume.

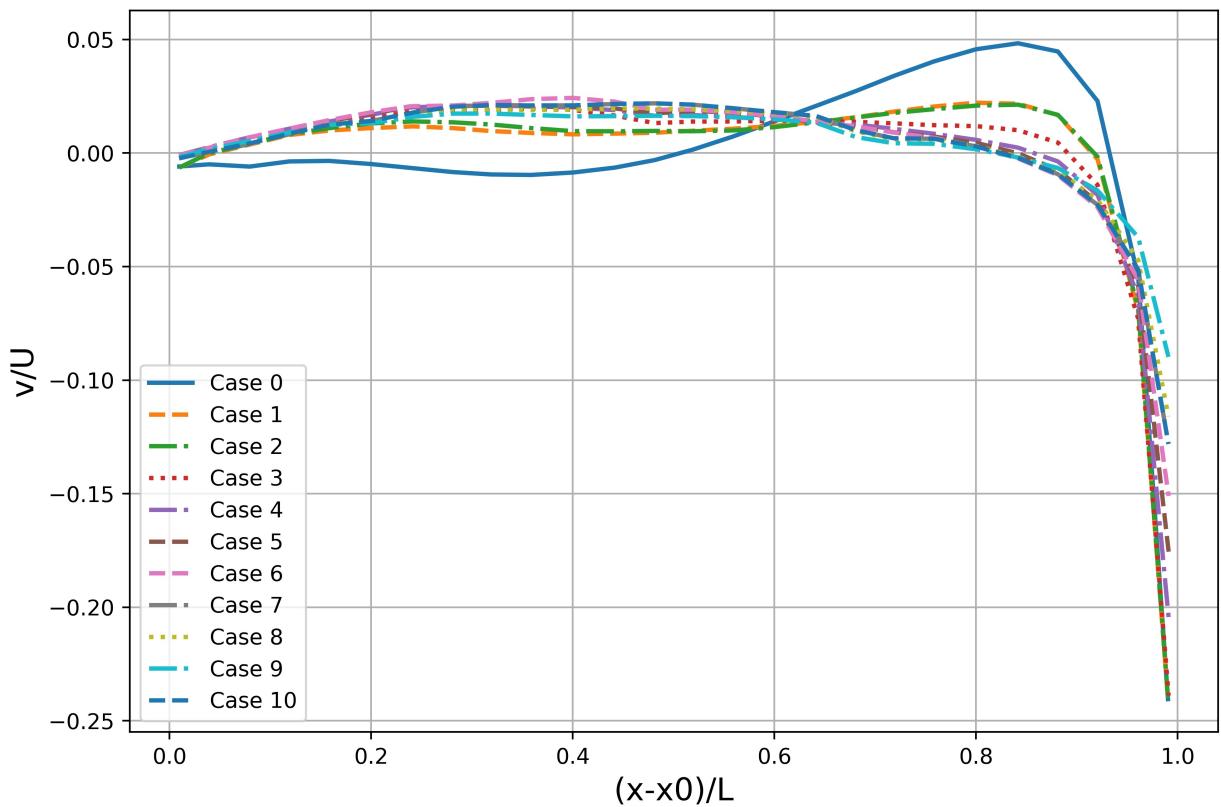


Figure 5.6: The variation of the transversal velocity in the interface between the cavity and the main channel along the x-axis. Positive values of  $v/U$  indicate the flow entering the cavity volume.

## 5.4 Hydrodynamics of the Mixing Layer

### 5.4.1 Thickness of the Mixing Layer

The mixing layer is a region that is developed along the interface due to a velocity gap between the lateral cavity and the main channel. The adoption of a thickness  $\delta$  (m) of the mixing layer is commonly used to describe the spreading angle of the mixing layer and the range of velocity gradients between the zones (XIANG; YANG; WU et al., 2020; MIGNOT et al., 2017; YOSSEF; VRIEND, 2011). Xiang, Yang, Wu et al. (2020) suggested that the thickness could be divided into an inner section  $\delta_{in}$  (m) (in the cavity) and an outer section  $\delta_{out}$  (m) (in the main channel). The total thickness is defined as:

$$\delta = \delta_{in} + \delta_{out} = \frac{U_i(x) - U_c(x)}{(\partial \bar{u} / \partial y)_{max}} + \frac{U_m(x) + U_i(x)}{(\partial \bar{u} / \partial y)_{max}} \quad (5.11)$$

where,  $U_i$ ,  $U_c$  and  $U_m$  (m/s) are the time-averaged streamwise velocities at the interface, in the cavity and the main channel. These velocities were extracted where the velocity gradient is negligibly small, i.e., lower than  $0.5 \text{ s}^{-1}$  in reference to Xiang, Yang, Wu et al. (2020) e Mignot et al. (2017).  $(\partial \bar{u} / \partial y)_{max}$  represents the maximum velocity gradient at each x position along the interface.

Figure 5.7 show the evolution of the thickness layer in the streamwise direction for all the cases. Overall, the mixing layer increased when  $(x - x_0)/L < 0.80$  and decreased when  $(x - x_0)/L > 0.80$  as the velocity gradient increased in the contact with the wall. Similar to Xiang, Yang, Wu et al. (2020), the vegetation density increase affected the width of the mixing layer, for both inner and outer sections. The increasing blockage limited the entrance of flow in the cavity (Figures 5.3 and 5.4), thus it limits the growth of the mixing layer. The wall behaviour of the cavity started to take place in case 8 and 9, although the presence of the secondary gyre in case 10 increased the thickness.

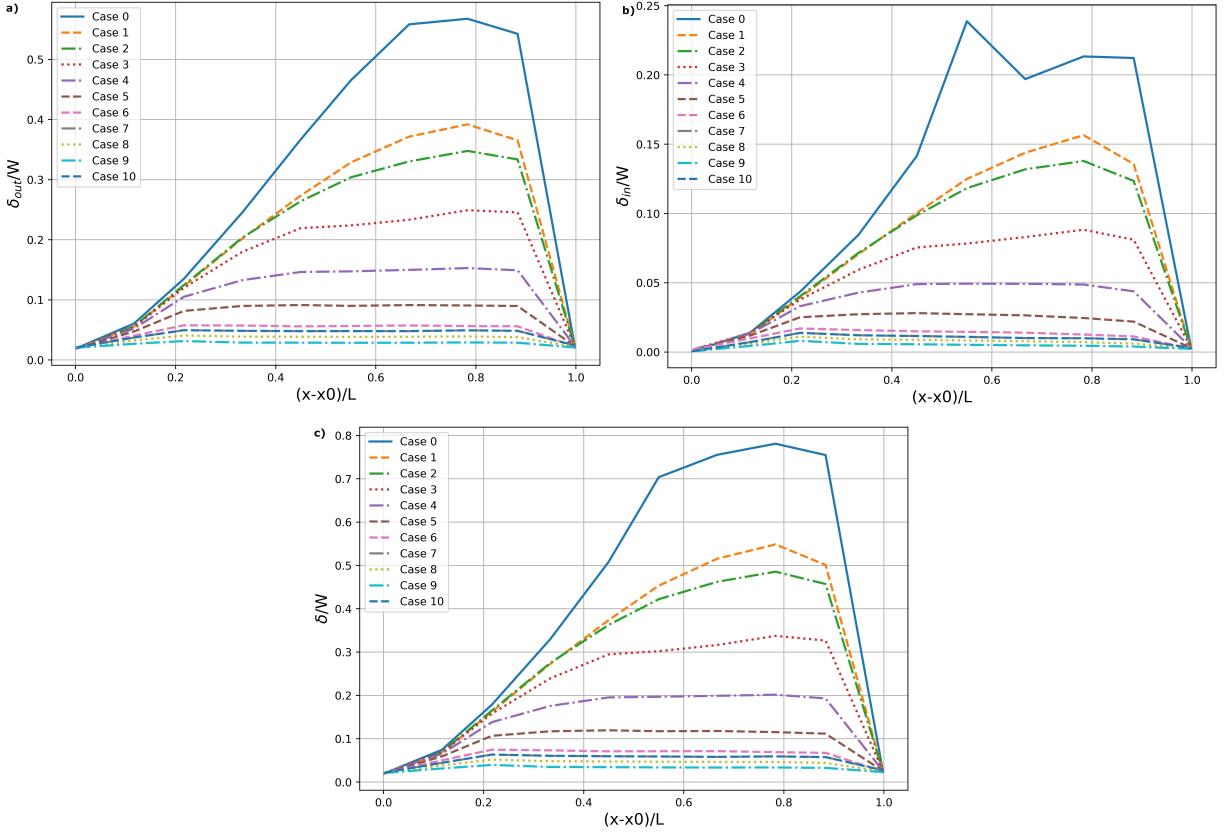


Figure 5.7: Evolution of the mixing layer thickness averaged at the z-axis: a) inner mixing layer; b) outer mixing layer and c) total mixing layer.

### 5.4.2 Vorticity

Figures 5.8 and 5.9 show the time-averaged vorticity magnitude (normalised by  $U/H$ ) at . The vorticity magnitude  $\Omega$  ( $s^{-1}$ ) is defined as:

$$\Omega = \nabla \times \vec{v} \quad (5.12)$$

where,  $\vec{v}$  (m/s) is the velocity vector.

For all cases, the vorticity remained high through all the interface between the cavity and the main channel. The maximum vorticity occurred at the upstream of the interface ( $x/L < 0.3$ ) and decreases in the downstream direction ( $x/L > 0.3$ ). Similar to groynes, this effect occurs to the shredding of vortex from the beginning of the cavity Xiang, Yang, Wu et al. (2020). As the eddies shred, the high vorticity region increases in width ( $y$ -axis) to its maximum value at the downstream wall. This width reduces as the vegetation density increases due to higher drag.

The increase of vegetation density gradually decreased the levels of vorticity inside the cavity volume up to  $a < 7.9920$ , when there was no more vorticity in the volume.

Although, it seems that vegetation increased the vorticity at the inner part of the mixing layer despite the blockage effect.

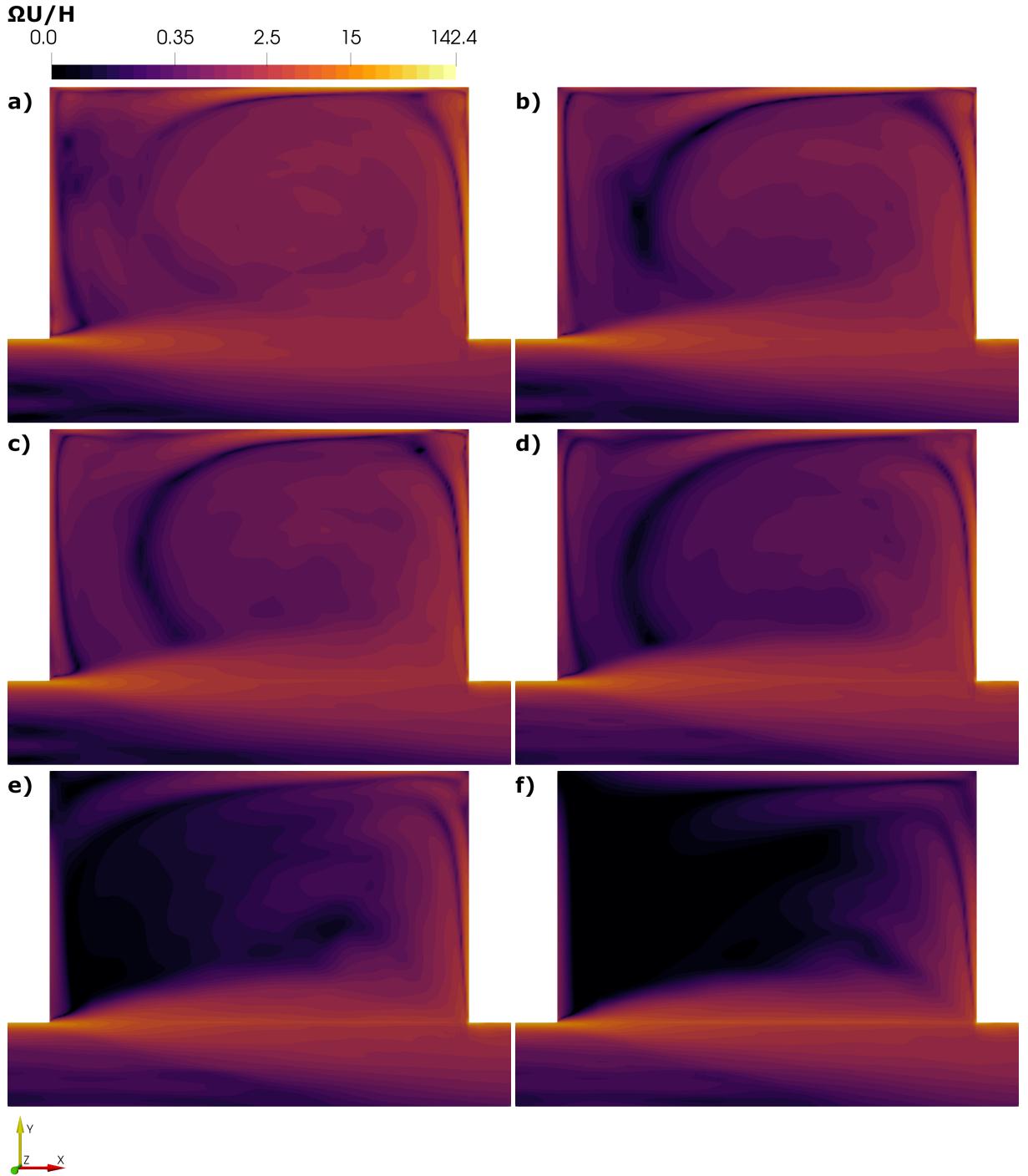


Figure 5.8: Time averaged vorticity at  $z/H = 0.6$ : a) Case 0, b) Case 1, c) Case 2, d) Case 3, e) Case 4 and f) Case 5.

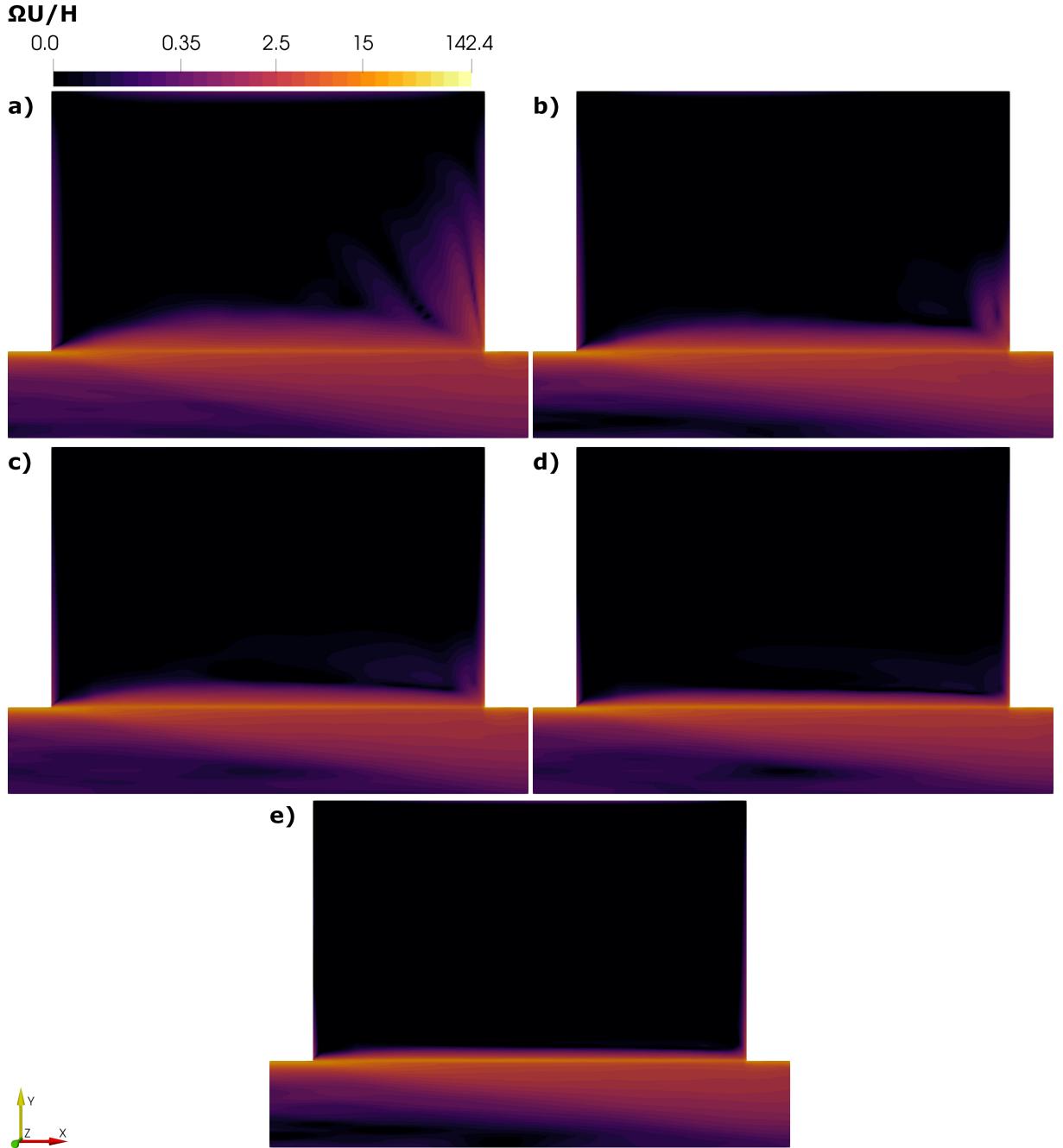


Figure 5.9: Time averaged vorticity at  $z/H = 0.6$ : a) Case 6, b) Case 7, c) Case 8, d) Case 9 and e) Case 10.

### 5.4.3 Turbulent Kinetic Energy (TKE)

The turbulent kinetic energy (TKE) in a LES simulation is defined as:

$$TKE = 0.5\text{tr}(R) + 0.5\text{tr}(u') \quad (5.13)$$

where,  $R$  ( $\text{m}^2/\text{s}^2$ ) is the Reynolds stress tensor and  $u'$  ( $\text{m/s}$ ) is the instantaneous fluctuation tensor.

A time-averaged TKE distribution, normalised by  $U^2$ , is presented in Figures 5.10

and 5.11. Through all interface, the values of TKE remained above  $TKE/U^2 > 0.05$ , at the downstream of the interface the maximum value occurred. At this same region, the vortex encounters the cavity lateral wall, the portion that enters the cavity reduces in magnitude as it moves through the vegetation. In an unvegetated scenario Figure 5.10 a) the TKE followed all the main circulation, behaviour that did not occur with the presence of vegetation. Hence the increase in vegetation density reduced the values of TKE, analogously to the vorticity.

As the vegetation density increased, the turbulent kinetic energy inside the cavity decreased, similarly to Xiang, Yang, Huai et al. (2019), although in a much faster rate than the vorticity. The blockage effect due to the vegetation density increase slowly reduces the TKE values inside the cavity. The last region in which  $TKE > 0$  is the downstream wall of the cavity, region where the first jet enters the volume. Similar to Xiang, Yang, Huai et al. (2019), the first levels of vegetation registered an increase of TKE at the interface. Although with the increase of vegetation beyond  $a > 0.33\%$  (Figure 5.10 d), the turbulence intensity was lower than the non vegetated scenario which can be attributed to the shrink of the inner part of the mixing layer (Figure 5.7 a) caused by the flow turbulence inhibition caused by high-density vegetation (NEPF, 2012). Furthermore, the shape of the TKE distribution on the outer part of the mixing layer changed with the increase of vegetation, on  $a = 0\%$  the region that the distribution width increases is up to approximately  $(x - x_0)/L < 0.8$ , when the jet entrance to the volume decreases its width. For the vegetated cases, specially Case 3, the vegetation blocks part of the jet that normally enters the downstream portion of the volume, this causes the TKE distribution to take a triangular shape which indicates an increased turbulence intensity in the main channel up to  $a < 5.328\%$ .

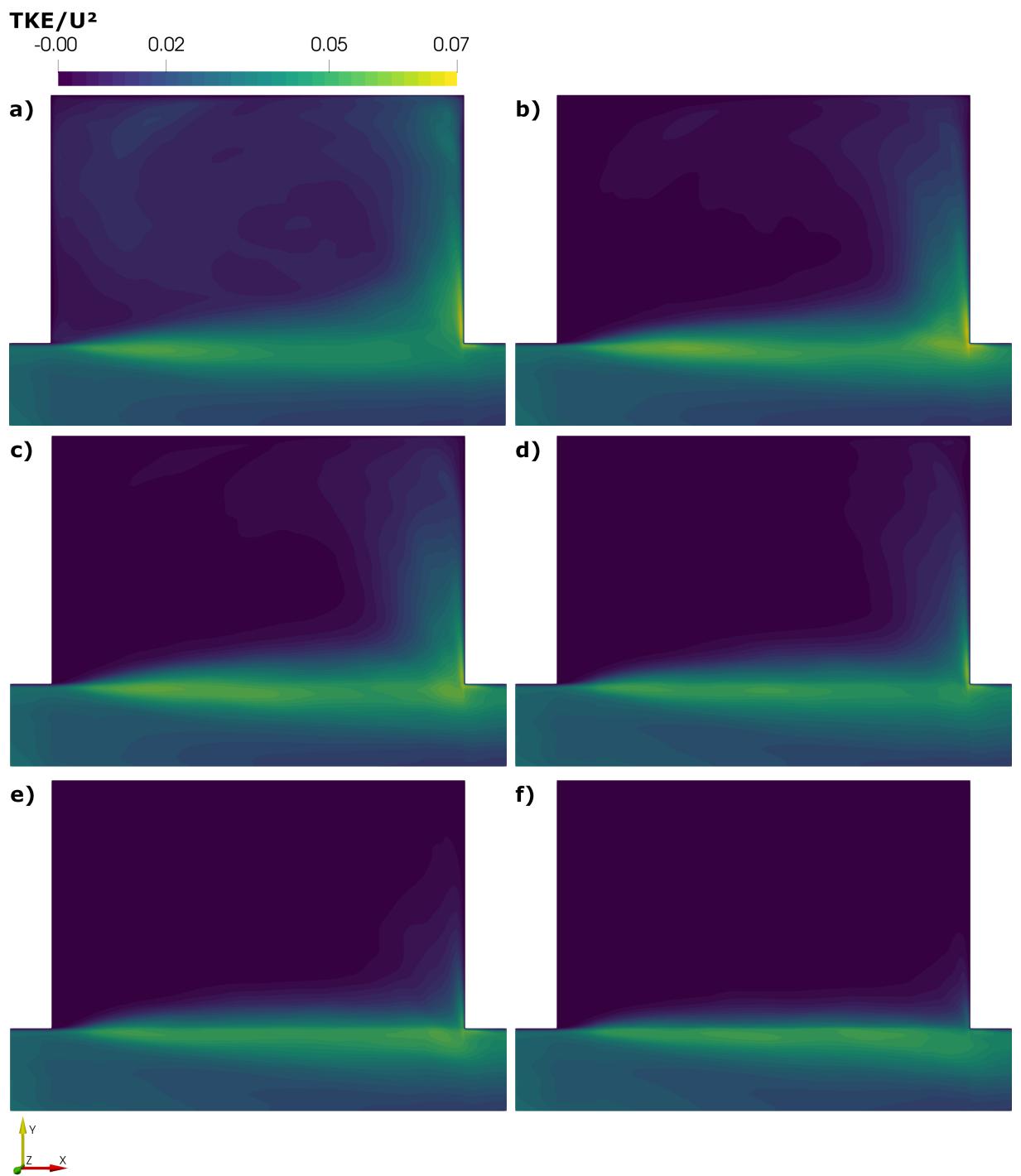


Figure 5.10: Time-averaged total kinetic energy (TKE) at  $z/h = 0.6$ : a) Case 0, b) Case 1, c) Case 2, d) Case 3, e) Case 4 and f) Case 5.

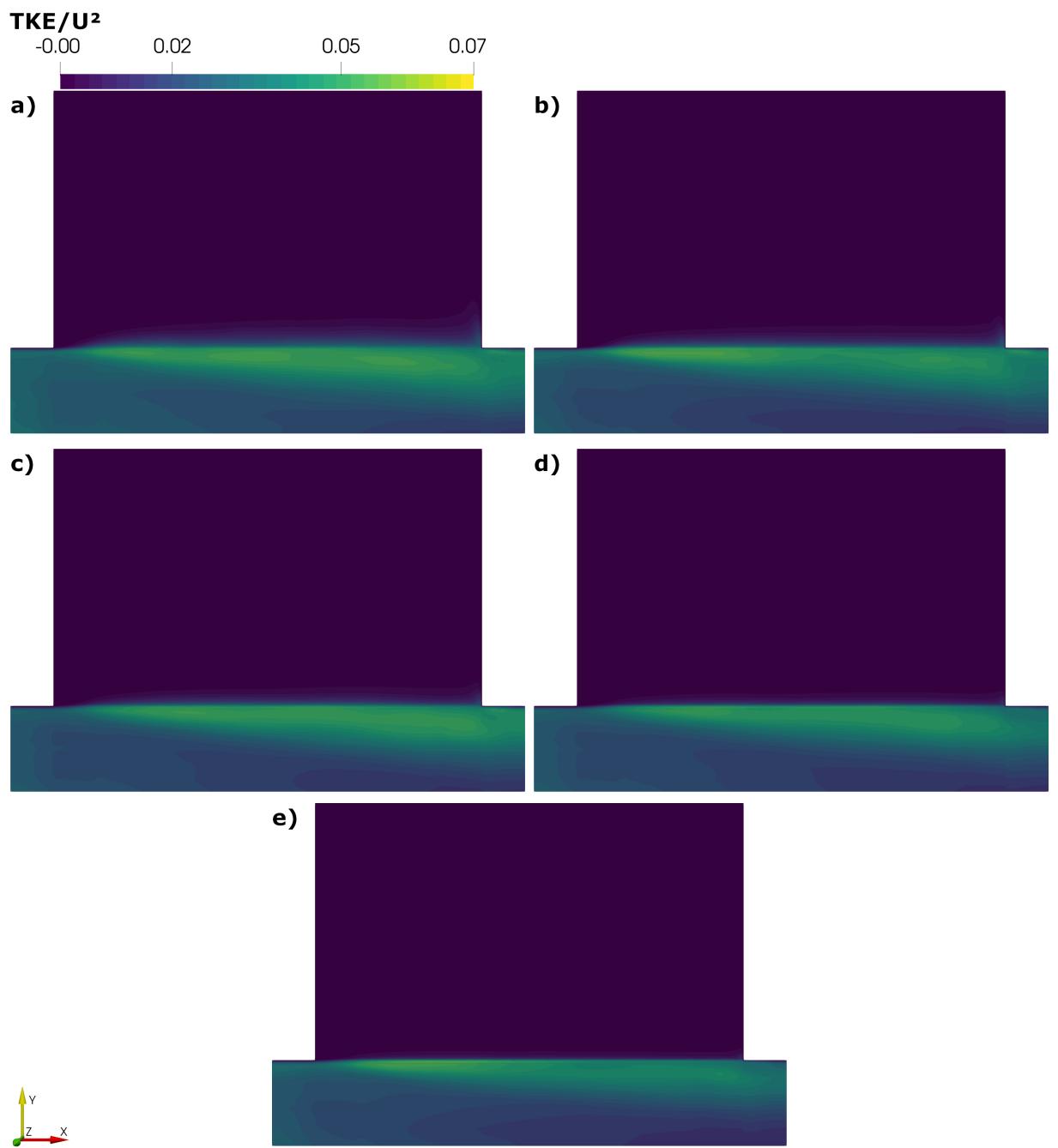


Figure 5.11: Time-averaged total kinetic energy (TKE) at  $z/h = 0.6$ : a) Case 6, b) Case 7, c) Case 8, d) Case 9 and e) Case 10.

## 5.5 Mass Exchange

The mass exchange coefficient  $k$  is an important parameter of the cavity, as one of its main characteristics is the transient storage of mass. This coefficient indicates the mass exchange rate between the cavity and the main channel. The evaluation of this parameter through tracer experiments was done using a first-order exponential decay in which the initial concentration was set to 1. Analogously, the mean retention time ( $T_{cav}$ ) is the time needed to completely replace the water volume in the cavity. This parameter was adjusted using a non-linear least square method to best approximate the value of  $T_{cav}$  to the volumetric-average tracer concentration through time (WEITBRECHT, 2004) (Figure 5.12).

$$C = C_0 e^{-t/T_{cav}} \quad (5.14)$$

$$k = \frac{W}{T_{cav} U} \quad (5.15)$$

where,  $C_0 = 1$  is the initial concentration,  $t$  (s) is the time and  $T_{cav}$  (s) is the mean retention time and  $k$  is the mass exchange coefficient.

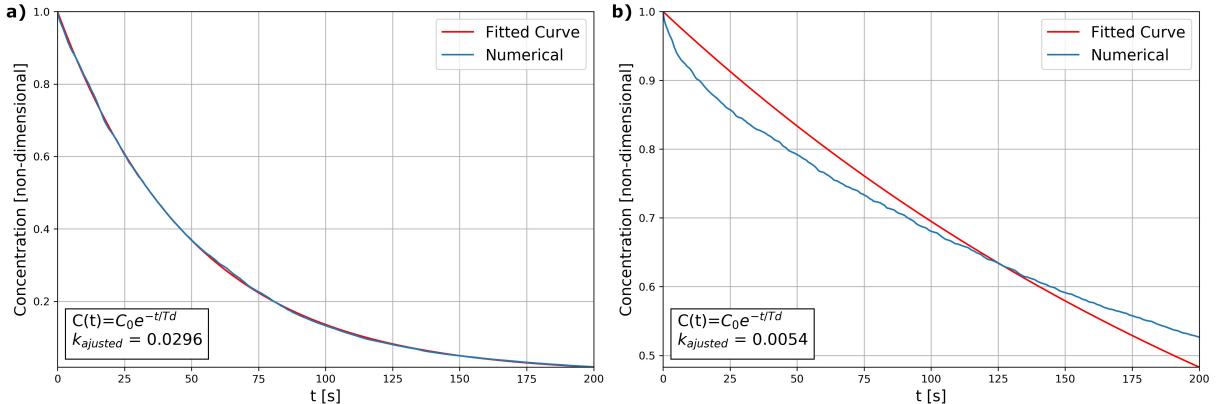


Figure 5.12: Volumetric-averaged tracer concentration decay inside the lateral cavity over time: a) Case 1 b) Case 8.

Figure 5.13 shows the variation of the mass exchange coefficient and the mean retention time with the increase of vegetation density  $a$ . Analogous to Xiang, Yang, Huai et al. (2019), the tracer fields indicated a deviation in the curve near  $a \approx 0.33$  which is related first to the plant-induced Karman vortex street and Kelvin-Helmholtz eddies (NEPF, 2012) which decreased the  $k$  decay rate and second the vegetation blockage that becomes the main effect further that point. Further that point, the mass exchange coefficient decreases with the increase of vegetation density in two different phases divided

at  $a \approx 4\%$ . It is possible to assume that this vegetation density acted as a wall as the flow cannot penetrate the cavity enough for the flow to occur, the remaining exchange occurred in a thin layer that further reduced its width as the density increased. The presence of the secondary circulation for  $a \geq 5.3280\%$  (Case 8) implied that a further increase in vegetation density could divide the secondary phase into a two slope section of the curve, where the first circulation ejects mass faster than the secondary with slower velocities and no contact with the main channel (OLIVEIRA; JANZEN, 2020) (Figure 5.12 a and b).

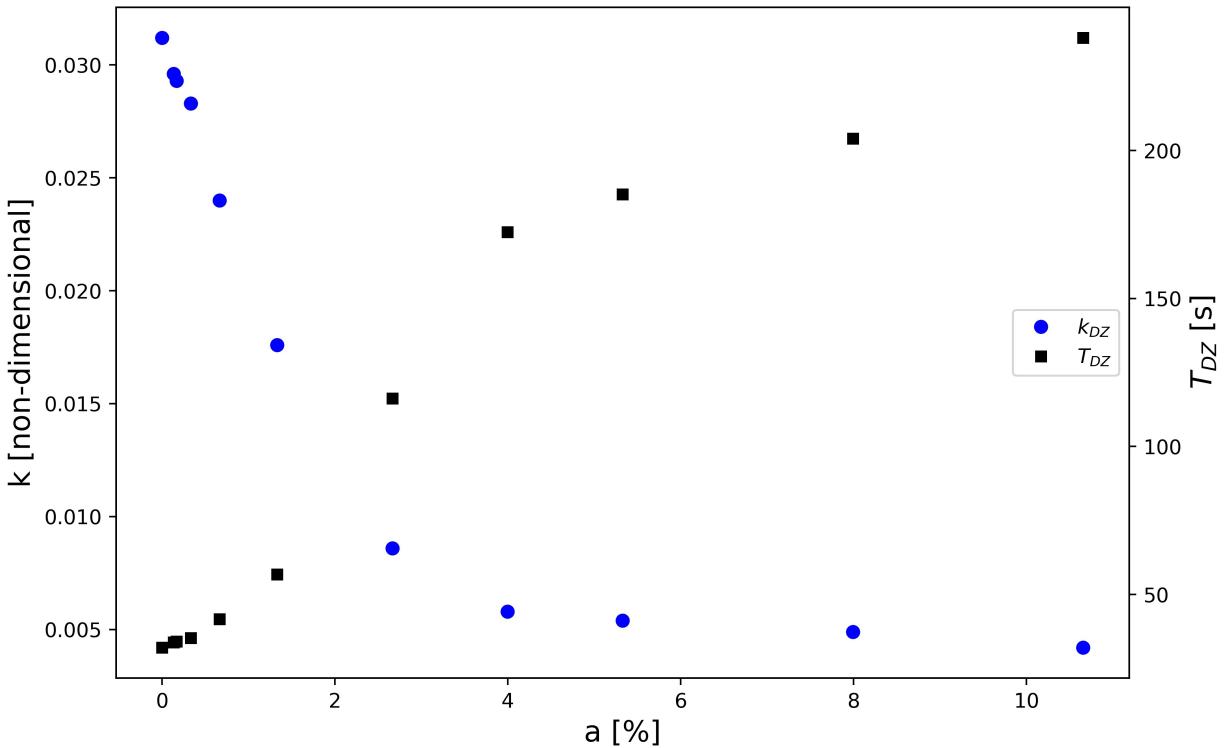


Figure 5.13: The variation of the mass exchange coefficient and the mean retention time with the increase of vegetation density.

For low-vegetation density ( $a < 4\%$ ),  $k$  drops off quickly with increasing vegetation density  $a$ . For high-vegetation density ( $a \geq 4\%$ ),  $k$  is small, but not zero, and decreases slowly with increasing  $a$ . As the vegetation drag becomes the dominant effect, the velocity within the cavity becomes negligibly small, and it behaves as if the cavity is fully blocked ( $\phi = 1$ ). The mass exchange, then, occurs mainly near the interface volume, this could be an influence of the increase of TKE in the outer part of the mixing layer. The effect of higher TKE levels in the outer mixing layer provides clear water volumes to scratch the vegetation at the interface. As the width of the TKE distribution decreases after a maximum at  $a = 5.3380\%$ , the diminishing rate that clear water is available at the interface further slows down the exchange between the zones. Furthermore, the presence of vorticity for  $a \leq 7.9920$  indicates that the small vortices could be moving tracer and promoting its diffusion particularly at the inner mixing layer. This behaviour contributed

to the mass exchange and the lack of this phenomenon can be seen past  $a \geq 7.9920$  when the mass exchange seems to change the asymptote of  $k$  decay.

## 5.6 Discussion

From a hydrodynamic perspective, the flow the vegetation drag slowly reduces the energy of the flow, be it inside the cavity or at the mixing layer. As the first jet-like flow collides with the downstream wall, the motion inside the cavity assumes a swirly pattern in which the vegetation slowly damps the energy impacting the magnitude of velocities inside the volume (SUKHODOLOV; SUKHODOLOVA; KRICK, 2017). From our results, it is clear to assume that this behaviour occurs as the velocity field did not only loss in magnitude but also shape as the density increased what demonstrates the correlation of vegetation density and the flow field in a lateral cavity. Furthermore, this process of reduction of velocity could promote sediment deposition, enhanced by the results of the turbulence fields that showed how the mixing inside the cavity slowly ceases. From a biological standpoint, the increase of vegetation could influence the spread of biota in streams promoting restoration along its path (e.g. fish breeding or crustacea habitat). The slow circulation allied with the deposition of sediments on its zone could further increase lateral heterogeneity which creates different environments and could be associated with a diversification of species in the implanted region. These deposited sediments can also represent a biological problem, as the cavity volume can become a source of pollutants once the system goes submerged (WEITBRECHT, 2004). Although, the presence of vegetation could improve water quality as the increased residence time could be long enough for plants to absorb these nutrients.

It is important, then, to assume levels in which the vegetation can favour different processes (e.g. vegetation growth or sediment catch). It seems that  $a = 0.6660\%$  (Case 4) is an import point as it represents a change in the format of the mixing layer length and also on the vorticity and TKE inside the cavity. As this after this value these variables or get smaller or cease to exist inside the volume it could be argued that this point represents a limit for the sparse vegetation. Another important metric for a threshold is the changes in mass exchange, from Figure 13, as this value of  $a$  represents the beginning of a faster decay in the mass exchange rate leading to a curve similar to the velocity in Chen et al. (2012).

Following the same logic, the point  $a = 3.9960\%$  (Case 7) can also be considered a milestone as this represents another inflection in the mass exchange curve. From another perspective, this density represents the end of TKE inside the volume further reducing the

mixing and thus the mass exchange. Furthermore, the mischaracterisation of the velocity fields beyond this density allowed the flow to assume another circulation pattern that would not be expected in a non-vegetated case.

Thus, we suggest a classification of the flow and its related mass exchange in a vegetated lateral cavity in three phases: 1) sparse ( $a < 0.6660 \%$ ), medium ( $0.6660 < a(\%) < 3.996$ ) and dense ( $a > 3.9960 \%$ ). In the sparse region, it is expected that the increased rate of mass exchange, compared to a non-vegetated case, might promote the settlement of particles evenly as it can be seen in Figure 5.3 a-d. This spread of particles could potentially promote uniform growth of vegetation while preserving the exchange with the main channel. Thus, making it an appropriate class for reducing the impact of pollutant spread in a reduced time (e.g. oil spill or first flush rain). The medium density class seems to be the best benefit for mass storage and mitigation of riverbank erosion. Although, one must have in mind that the reduced mass exchange rate might impact the catchment of a sporadic pollutant release, that being said we recommend this class for long term catchments (e.g. illegal sewage release). Lastly, the high-density class seems to be the most effective in mitigating the riverbank erosion, especially at the lower right corner of the lateral cavity ( $x/L = 1$  and  $(y - y_0)/H = 0$ ) as the blockage effect does not allow the impact of a jet at the wall of the cavity.

## 5.7 Conclusion

The hydrodynamics of a single lateral cavity with different vegetation densities was investigated numerically through LES. The results reveal that the single circulation system (non vegetated case) can be transformed into a two-gyre system with the increase of vegetation density. The influence of this secondary gyre decreased the rate in which the mass exchange coefficient diminished.

The dynamic of the flow was examined with both the vorticity and the turbulent kinetic energy (TKE) that both decreased in the downstream direction for all vegetation densities. For  $a < 2.6640 \%$ , the downstream section of the mixing layer has higher values of both TKE and vorticity due to reduced inflow of the shed vortices in the cavity. For  $a > 2.6640 \%$ , these higher values did not appear as the vegetation drag further increased, which is attributable to high blockage effect. The effect of these variables seems to play the dominant effect of the mass exchange in high-density vegetation, as the mass exchange mostly occurs at the inner mixing layer, region where these variables remain not null up to  $a = 5.3380 \%$  for TKE and  $a = 7.9920\%$  for vorticity.

This study enriched the knowledge of interactions between aquatic vegetation and the flow inside a lateral cavity. It shows that vegetation can drastically alter the flow by reducing the velocity, TKE and vorticity, this influence could promote the deposition of fine sediments and organic matter. Furthermore, it shows that the vegetation can cause a threshold in the mass exchange between the main channel and the lateral cavity, in which the rate is drastically reduced due to high blockage effects. This knowledge could help river managers to set limits and adjust the vegetation density inside the cavity in order to keep the desirable ecological function of the cavity.

## Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) -Finance Code 001.

This study was partly conducted in Lobo Carneiro cluster located in NACAD/-Coppe - Rio de Janeiro, Brazil

## References

- ALFRINK, Ben J.; RIJN, Leo C. van. Two-Equation Turbulence Model for Flow in Trenches. **Journal of Hydraulic Engineering**, v. 109, n. 7, p. 941–958, 1983. DOI: 10.1061/(asce)0733-9429(1983)109:7(941). Disponível em: <<http://ascelibrary.org/doi/10.1061/%5C%28ASCE%5C%290733-9429%5C%281983%5C%29109%5C%3A7%5C%28941%5C%29>>. Citado nas pp. 11, 43.
- ARENDE, Kristin K.; BAIN, Mark B. Fish communities in coastal freshwater ecosystems: The role of the physical and chemical setting. **BMC Ecology**, 2008. ISSN 14726785. DOI: 10.1186/1472-6785-8-23. Citado na p. 39.
- ASAEDA, Takashi et al. Growth of Phragmites japonica on a sandbar of regulated river: morphological adaptation of the plant to low water and nutrient availability in the substrate. **River Research and Applications**, v. 25, n. 7, p. 874–891, set. 2009. ISSN 15351459. DOI: 10.1002/rra.1191. Disponível em: <<http://doi.wiley.com/10.1002/rra.1191>>. Citado na p. 38.
- BARKO, John W.; GUNNISON, Douglas; CARPENTER, Stephen R. Sediment interactions with submersed macrophyte growth and community dynamics. **Aquatic Botany**, v. 41, n. 1-3, p. 41–65, jan. 1991. ISSN 03043770. DOI:

- 10.1016/0304-3770(91)90038-7. Disponível em:  
<<https://linkinghub.elsevier.com/retrieve/pii/0304377091900387>>. Citado na p. 38.
- BREVIS, W.; GARCÍA-VILLALBA, M.; NIÑO, Y. Experimental and large eddy simulation study of the flow developed by a sequence of lateral obstacles. **Environmental Fluid Mechanics**, v. 14, n. 4, p. 873–893, 2014. ISSN 15677419. DOI: 10.1007/s10652-013-9328-x. Citado nas pp. 11, 43.
- CHEN, Zhengbing et al. The wake structure behind a porous obstruction and its implications for deposition near a finite patch of emergent vegetation. **Water Resources Research**, v. 48, n. 9, 2012. DOI: <https://doi.org/10.1029/2012WR012224>. Disponível em:  
<<https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2012WR012224>>. Citado nas pp. 7, 40, 48, 60.
- CONSTANTINESCU, George; SUKHODOLOV, Alexander; MCCOY, Andrew. Mass exchange in a shallow channel flow with a series of groynes: Les study and comparison with laboratory and field experiments. **Environmental Fluid Mechanics**, v. 9, n. 6, p. 587–615, 2009. ISSN 15677419. DOI: 10.1007/s10652-009-9155-2. Citado na p. 40.
- COTTON, J. A. et al. The effects of seasonal changes to in-stream vegetation cover on patterns of flow and accumulation of sediment. **Geomorphology**, 2006. ISSN 0169555x. DOI: 10.1016/j.geomorph.2006.01.010. Citado na p. 38.
- DROST, Kevin J. et al. Parameterization of Mean Residence Times in Idealized Rectangular Dead Zones Representative of Natural Streams. **Journal of Hydraulic Engineering**, American Society of Civil Engineers, v. 140, n. 8, p. 04014035, ago. 2014. ISSN 0733-9429. DOI: 10.1061/(asce)hy.1943-7900.0000879. Disponível em:  
<[https://doi.org/10.1061/\(ASCE\)HY.1943-7900.0000879%20http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D29HY.1943-7900.0000879](https://doi.org/10.1061/(ASCE)HY.1943-7900.0000879%20http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D29HY.1943-7900.0000879)>. Citado nas pp. 3, 38.
- DURÓ, G. et al. Bank Erosion Processes in Regulated Navigable Rivers. **Journal of Geophysical Research: Earth Surface**, v. 125, n. 7, jul. 2020. ISSN 2169-9003. DOI: 10.1029/2019jf005441. Disponível em:  
<<https://onlinelibrary.wiley.com/doi/10.1029/2019JF005441>>. Citado na p. 39.
- DUTTA, Rabijit; XING, Tao. Five-equation and robust three-equation methods for solution verification of large eddy simulation. **Journal of Hydrodynamics**, v. 30, n. 1, p. 23–33, fev. 2018. ISSN 1001-6058. DOI: 10.1007/s42241-018-0002-0. Disponível em:  
<<http://link.springer.com/10.1007/s42241-018-0002-0>>. Citado nas pp. 45, 149.
- ELY, Joseph S.; EVANS, Dan K. Plant communities of selected embayments along the mid- to mid-upper ohio river floodplain. **Journal of the Botanical Research Institute of Texas**, 2010. ISSN 19345259. Citado na p. 39.

- ENSIGN, Scott H.; DOYLE, Martin W. In-channel transient storage and associated nutrient retention: Evidence from experimental manipulations. **Limnology and Oceanography**, 2005. ISSN 00243590. DOI: 10.4319/lo.2005.50.6.1740. Citado na p. 38.
- FORREST, Barrie M. et al. Multiple indicators reveal river plume influence on sediments and benthos in a New Zealand coastal embayment. **New Zealand Journal of Marine and Freshwater Research**, 2007. ISSN 11758805. DOI: 10.1080/00288330709509892. Citado na p. 38.
- GOOSEFF, Michael N. et al. Determining in-channel (dead zone) transient storage by comparing solute transport in a bedrock channel-alluvial channel sequence, Oregon. **Water Resources Research**, 2005. ISSN 00431397. DOI: 10.1029/2004wr003513. Citado na p. 38.
- GUALTIERI, Carlo; ANGELOUDIS, Athanasios et al. On the values for the turbulent schmidt number in environmental flows. **Fluids**, v. 2, n. 2, 2017. ISSN 23115521. DOI: 10.3390/fluids2020017. Citado na p. 44.
- HARVEY, J W. Chapter 1 - Hydrologic Exchange Flows and Their Ecological Consequences in River Corridors. In: JONES, Jeremy B; STANLEY, Emily H (Ed.). **Stream Ecosystems in a Changing Environment**. Boston: Academic Press, 2016. p. 1–83. ISBN 978-0-12-405890-3. DOI: <https://doi.org/10.1016/B978-0-12-405890-3.00001-4>. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9780124058903000014>>. Citado nas pp. 3, 38.
- HARVEY, Jud; GOOSEFF, Michael. River corridor science: Hydrologic exchange and ecological consequences from bedforms to basins. **Water Resources Research**, v. 51, n. 9, p. 6893–6922, set. 2015. ISSN 00431397. DOI: 10.1002/2015wr017617. Disponível em: <<http://doi.wiley.com/10.1002/2015WR017617>>. Citado na p. 38.
- JACKSON, T. R.; HAGGERTY, R.; APTE, S. V. A fluid-mechanics based classification scheme for surface transient storage in riverine environments: Quantitatively separating surface from hyporheic transient storage. **Hydrology and Earth System Sciences**, v. 17, n. 7, p. 2747–2779, 2013. ISSN 10275606. DOI: 10.5194/hess-17-2747-2013. Citado nas pp. 2–4, 23, 39.
- JACKSON, Tracie R.; APTE, Sourabh V. et al. Flow structure and mean residence times of lateral cavities in open channel flows: influence of bed roughness and shape. **Environmental Fluid Mechanics**, Springer Netherlands, v. 15, n. 5, p. 1069–1100, 2015. ISSN 15731510. DOI: 10.1007/s10652-015-9407-2. Disponível em: <<http://dx.doi.org/10.1007/s10652-015-9407-2>>. Citado nas pp. 3, 38.

JACKSON, Tracie R.; HAGGERTY, Roy; APTE, Sourabh V.; COLEMAN, Anthony et al. Defining and measuring the mean residence time of lateral surface transient storage zones in small streams. **Water Resources Research**, v. 48, n. 10, p. 1–20, 2012. ISSN 00431397. DOI: 10.1029/2012wr012096. Citado nas pp. 3, 38.

JACKSON, Tracie R.; HAGGERTY, Roy; APTE, Sourabh V.; O'CONNOR, Ben L. A mean residence time relationship for lateral cavities in gravel-bed rivers and streams: Incorporating streambed roughness and cavity shape. **Water Resources Research**, v. 49, n. 6, p. 3642–3650, 2013. ISSN 00431397. DOI: 10.1002/wrcr.20272. Citado na p. 38.

JONES, R. Christian. Recovery of a Tidal Freshwater Embayment from Eutrophication: a Multidecadal Study. **Estuaries and Coasts**, 2020. ISSN 15592731. DOI: 10.1007/s12237-020-00730-3. Citado na p. 39.

JUEZ, Carmelo et al. Transport of suspended sediments under the influence of bank macro-roughness. **Earth Surface Processes and Landforms**, v. 43, n. 1, p. 271–284, jan. 2017. ISSN 01979337. DOI: 10.1002/esp.4243. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/esp.4243>>. Citado na p. 38.

KRAUS, Richard T.; JONES, R. Christian. Fish abundances in shoreline habitats and submerged aquatic vegetation in a tidal freshwater embayment of the Potomac River. **Environmental Monitoring and Assessment**, 2012. ISSN 01676369. DOI: 10.1007/s10661-011-2192-6. Citado na p. 39.

KURZ, Marie J. et al. Impacts of water level on metabolism and transient storage in vegetated lowland rivers: Insights from a mesocosm study. **Journal of Geophysical Research: Biogeosciences**, 2017. ISSN 21698961. DOI: 10.1002/2016jg003695. Citado na p. 39.

LU, J.; DAI, H. C. Large eddy simulation of flow and mass exchange in an embayment with or without vegetation. **Applied Mathematical Modelling**, v. 40, n. 17-18, p. 7751–7767, 2016. ISSN 0307904x. DOI: 10.1016/j.apm.2016.03.026. Citado na p. 39.

MACEINA, Michael J.; SLIPKE, Jeffery W.; GRIZZLE, John M. Effectiveness of Three Barrier Types for Confining Grass Carp in Embayments of Lake Seminole, Georgia. **North American Journal of Fisheries Management**, 1999. ISSN 0275-5947. DOI: 10.1577/1548-8675(1999)019<0968:eotbt>2.0.co;2. Citado na p. 39.

MEILE, Tobias; BOILLAT, Jean Louis; SCHLEISS, Anton J. Water-surface oscillations in channels with axi-symmetric cavities. **Journal of Hydraulic Research**, 2011. ISSN 00221686. DOI: 10.1080/00221686.2010.534671. Citado na p. 38.

- MIGNOT, Emmanuel et al. Measurement of mass exchange processes and coefficients in a simplified open-channel lateral cavity connected to a main stream. **Environmental Fluid Mechanics**, Springer Netherlands, v. 17, n. 3, p. 429–448, 2017. ISSN 15731510. DOI: 10.1007/s10652-016-9495-7. Citado nas pp. 10, 38, 40, 51.
- MULHOLLAND, P. J. et al. Effect of periphyton biomass on hydraulic characteristics and nutrient cycling in streams. **Oecologia**, 1994. ISSN 00298549. DOI: 10.1007/bf00326088. Citado na p. 38.
- NEPF, Heidi M. Hydrodynamics of vegetated channels. **Journal of Hydraulic Research**, v. 50, n. 3, p. 262–279, 2012. ISSN 00221686. DOI: 10.1080/00221686.2012.696559. Citado nas pp. 38, 44, 55, 58.
- NICOUD, Franck; DUCROS, Frédéric. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. **Flow, Turbulence and Combustion**, Springer Verlag (Germany), v. 62, n. 3, p. 183–200, 1999. DOI: 10.1023/a:1009995426001. Disponível em: <<https://hal.archives-ouvertes.fr/hal-00910373>>. Citado na p. 41.
- OLDHAM, C. E.; STURMAN, J. J. The effect of emergent vegetation on convective flushing in shallow wetlands: Scaling and experiments. **Limnology and Oceanography**, v. 46, n. 6, p. 1486–1493, 2001. DOI: 10.4319/lo.2001.46.6.1486. eprint: <https://aslopubs.onlinelibrary.wiley.com/doi/pdf/10.4319/lo.2001.46.6.1486>. Disponível em: <<https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lo.2001.46.6.1486>>. Citado nas pp. 31, 42.
- OLESEN, Birgit. Regulation of light attenuation and eelgrass *Zostera marina* depth distribution in a Danish embayment. **Marine Ecology Progress Series**, 1996. ISSN 01718630. DOI: 10.3354/meps134187. Citado na p. 39.
- OLIVEIRA, Luiz E D de; JANZEN, Johannes G. Mass Exchange in Dead Water Zones: A Numerical Approach. In: **Water, Energy and Food Nexus in the Context of Strategies for Climate Change Mitigation**. Edição: Walter Leal Filho e José Baltazar Salgueirinho de Andrade Guerra. Cham: Springer International Publishing, 2020. p. 59–68. ISBN 978-3-030-57235-8. DOI: 10.1007/978-3-030-57235-8\_5. Disponível em: <<https://doi.org/10.1007/978-3-030-57235-8%7B%5C%5F%7D5>>. Citado nas pp. 3, 59.
- OURO, Pablo; JUEZ, Carmelo; FRANCA, Mário. Drivers for mass and momentum exchange between the main channel and river bank lateral cavities. **Advances in Water Resources**, Elsevier Ltd, v. 137, mar. 2020. ISSN 03091708. DOI: 10.1016/j.advwatres.2020.103511. Citado na p. 38.

POLETTO, R.; CRAFT, T.; REVELL, A. A New Divergence Free Synthetic Eddy Method for the Reproduction of Inlet Flow Conditions for LES. **Flow, Turbulence and Combustion**, v. 91, n. 3, p. 519–539, 2013. DOI: 10.1007/s10494-013-9488-2. Citado na p. 44.

RIBI, J.-M. et al. Attractiveness of a lateral shelter in a channel as a refuge for juvenile brown trout during hydropeaking. **Aquatic Sciences**, v. 76, n. 4, p. 527–541, 2014. ISSN 1420-9055. DOI: 10.1007/s00027-014-0351-x. Disponível em: <<https://doi.org/10.1007/s00027-014-0351-x>>. Citado nas pp. 3, 38.

SANJOU, Michio; AKIMOTO, Tetsuro; OKAMOTO, Takaaki. Three-dimensional turbulence structure of rectangular side-cavity zone in open-channel streams. **International Journal of River Basin Management**, v. 10, n. 4, p. 293–305, 2012. ISSN 15715124. DOI: 10.1080/15715124.2012.717943. Citado na p. 38.

SUKHODOLOV, Alexander; UIJTTEWAAL, Wim S J; ENGELHARDT, Christof. On the correspondence between morphological and hydrodynamical patterns of groyne fields. **Earth Surface Processes and Landforms**, v. 27, n. 3, p. 289–305, 2002. ISSN 01979337. DOI: 10.1002/esp.319. Citado na p. 47.

SUKHODOLOV, Alexander N. Hydrodynamics of groyne fields in a straight river reach: insight from field experiments. **Journal of Hydraulic Research**, Taylor & Francis, v. 52, n. 1, p. 105–120, 2014. DOI: 10.1080/00221686.2014.880859. eprint: <https://doi.org/10.1080/00221686.2014.880859>. Disponível em: <<https://doi.org/10.1080/00221686.2014.880859>>. Citado nas pp. 3, 10, 16, 25, 44.

SUKHODOLOV, Alexander N.; SUKHODOLOVA, Tatiana A.; KRICK, Julian. Effects of vegetation on turbulent flow structure in groyne fields. **Journal of Hydraulic Research**, v. 55, n. 1, p. 1–15, 2017. ISSN 00221686. DOI: 10.1080/00221686.2016.1211183. Citado nas pp. 6, 23, 39–41, 60.

UIJTTEWAAL, By W S J; LEHMANN, D; MAZIJK, a Van. Exchange Processes Between a River and Model Experiments Its Groyne Fields : Model Experiments. **Journal of Hydraulic Engineering**, v. 127, November, p. 928–936, 2001. Citado nas pp. 31, 40, 43.

VANDENBRUWAENE, W. et al. Flow interaction with dynamic vegetation patches: Implications for biogeomorphic evolution of a tidal landscape. **Journal of Geophysical Research: Earth Surface**, v. 116, F1, n/a–n/a, mar. 2011. ISSN 01480227. DOI: 10.1029/2010jf001788. Disponível em: <<http://doi.wiley.com/10.1029/2010JF001788>>. Citado na p. 38.

- WARD, Larry G.; MICHAEL KEMP, W.; BOYNTON, Walter R. The influence of waves and seagrass communities on suspended particulates in an estuarine embayment. **Marine Geology**, 1984. ISSN 00253227. DOI: 10.1016/0025-3227(84)90089-6. Citado nas pp. 38, 39.
- WATTS, Robyn J.; JOHNSON, Michael S. Estuaries, lagoons and enclosed embayments: Habitats that enhance population subdivision of inshore fishes. **Marine and Freshwater Research**, 2004. ISSN 13231650. DOI: 10.1071/mf04051. Citado na p. 38.
- WEITBRECHT, V.; JIRKA, G. Flow Patterns In Dead Zones of Rivers and their Effect On Exchange Processes. In: citado nas pp. 3, 4, 11, 15, 40.
- WEITBRECHT, Volker. **Influence of dead-water zones on the dispersive mass-transport in rivers**. 2004. f. 130. 130 f. Tese (Doutorado) – Karlsruhe Institute of Technology, Karlsruhe. ISBN 3-937300-07-4. DOI: 10.5445/ksp/1242004. Disponível em: <<https://publikationen.bibliothek.kit.edu/1242004>>. Citado nas pp. 1–3, 5, 11, 15, 40, 48, 58, 60.
- XIANG, Ke; YANG, Zhonghua; HUAI, Wenxin et al. Large eddy simulation of turbulent flow structure in a rectangular embayment zone with different population densities of vegetation. **Environmental Science and Pollution Research**, Environmental Science e Pollution Research, v. 26, n. 14, p. 14583–14597, mai. 2019. ISSN 1614-7499. DOI: 10.1007/s11356-019-04709-x. Disponível em: <<https://doi.org/10.1007/s11356-019-04709-x>>. Citado nas pp. 6, 10, 23–25, 29–32, 39–43, 45, 55, 58.
- XIANG, Ke; YANG, Zhonghua; WU, Shiqiang et al. Flow hydrodynamics of the mixing layer in consecutive vegetated groyne fields. **Physics of Fluids**, AIP Publishing, LLC, v. 32, n. 6, p. 1–22, 2020. ISSN 10897666. DOI: 10.1063/5.0006317. Disponível em: <<https://doi.org/10.1063/5.0006317>>. Citado nas pp. 2, 3, 6, 39–41, 51, 52.
- YOSSEF, Mohamed F. M.; VRIEND, Huib J. de. Flow Details near River Groynes: Experimental Investigation. **Journal of Hydraulic Engineering**, v. 137, n. 5, p. 504–516, mai. 2011. ISSN 0733-9429. DOI: 10.1061/(asce)hy.1943-7900.0000326. Disponível em: <<http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D29HY.1943-7900.0000326>>. Citado na p. 51.

# Chapter 6

## Conclusion and Recommendations

The objective of this study was the description of the hydrodynamics and mass exchange in dead waters. In the present study, numerical experiments were performed to describe the flow in groynes and lateral cavities.

First, a literature review of the flow conditions and methods used to describe the flow was performed which showed gaps in knowledge to be fulfilled. Second, the first description of the groyne flow was presented along with the study of tracer fields, which showed that the mass exchange between the DZ and the main channel is not unique and occurs at two different rates depending on the elapsed time. Third, the first description of lateral cavities was presented, in this paper, we described the flow using a model developed in commercial software. Forth, the description of lateral cavities was further developed using another approach to the turbulence fields and an open-source package. Finally, the effects of vegetation in lateral cavities were described.

The differences in modelling groynes and lateral cavities are significant. The study of groynes implies a periodicity that must be fulfilled by two means: a) a series of groyne fields or b) a pair of cyclic/periodic surfaces. The difficulties inherent by both methods rely on the high computational cost, as in the option a) the domain is extensive and the meshing process is harder and usually means in a loss of detail as a refined mesh becomes prohibitive. On another hand, the second option provides a more accurate description of the flow as the meshing can be concentrated only on one groyne field, although the implementation of a periodic surface in a zone of high mixture implies in a requirement for a small cell size especially in the groyne head, a region of intense vortex shredding. In contrast to groynes, cavities do not require repetition and can be represented in a simple inlet/outlet scheme, this implies reduced computational costs.

For both studies, as turbulence is the main phenomena in DZ the selection of the

turbulence model is primordial to the model. Through the investigation process, it was noticed that the Reynolds Averaging Navier-Stokes can only give an approximation of the flow, we analysed multiple models that rely on this spectra and only the  $k-\omega$  SST model was suitable for this kind of flow. Another hybrid model such as the Detached Eddy Simulation was a further improvement to the description of turbulence. Although, some structures were clearer when the Large Eddy Simulation was introduced, as the instantaneous flow was considered.

For vegetated flows, the approximation using porous media to represent the vegetation drag was used. The results of this model proved that this approach is viable for DZ. The effects of the vegetation density in the hydrodynamics and mass exchange proved to follow different phases. The lateral cavity can present a structure that was not anticipated for the given  $W/L$  region as a secondary circulation appears when  $a > 5.3280\%$ . This secondary circulation changes how mass is exchanged, similar to the first paper of this dissertation, the exchange values are changed due to the concentration of mass in the secondary gyre that has no contact with the main channel.

This dissertation enriched the knowledge of dead zones vegetated/non-vegetated. It showed different modelling techniques for two types of DZ: lateral cavity and groyne fields. Furthermore, it shows that vegetation can drastically alter the flow by reducing the velocity, TKE and vorticity, this influence could promote the deposition of fine sediments and organic matter. Additionally, it shows that the vegetation can cause a threshold in the mass exchange between the main channel and the lateral cavity, in which the rate is drastically reduced due to high blockage effects. This knowledge could help river managers to set limits and adjust the vegetation density inside the cavity to keep the desirable ecological function of the cavity. All codes generated within this dissertation can be found in: <https://github.com/Worth-Option/massExchangeInDeadWaters-ANumericalApproach>.

# References

- ALFRINK, Ben J.; RIJN, Leo C. van. Two-Equation Turbulence Model for Flow in Trenches. **Journal of Hydraulic Engineering**, v. 109, n. 7, p. 941–958, 1983. DOI: 10.1061/(asce)0733-9429(1983)109:7(941). Disponível em: <<http://ascelibrary.org/doi/10.1061/%5C%28ASCE%5C%290733-9429%5C%281983%5C%29109%5C%3A7%5C%28941%5C%29>>. Citado nas pp. 11, 43.
- ARENDE, Kristin K.; BAIN, Mark B. Fish communities in coastal freshwater ecosystems: The role of the physical and chemical setting. **BMC Ecology**, 2008. ISSN 14726785. DOI: 10.1186/1472-6785-8-23. Citado na p. 39.
- ARGERICH, Alba et al. Influence of transient storage on stream nutrient uptake based on substrata manipulation. **Aquatic Sciences**, v. 73, n. 3, p. 365–376, 2011. ISSN 1420-9055. DOI: 10.1007/s00027-011-0184-9. Disponível em: <<https://doi.org/10.1007/s00027-011-0184-9>>. Citado na p. 29.
- ASAEDA, Takashi et al. Growth of Phragmites japonica on a sandbar of regulated river: morphological adaptation of the plant to low water and nutrient availability in the substrate. **River Research and Applications**, v. 25, n. 7, p. 874–891, set. 2009. ISSN 15351459. DOI: 10.1002/rra.1191. Disponível em: <<http://doi.wiley.com/10.1002/rra.1191>>. Citado na p. 38.
- BARKO, John W.; GUNNISON, Douglas; CARPENTER, Stephen R. Sediment interactions with submersed macrophyte growth and community dynamics. **Aquatic Botany**, v. 41, n. 1-3, p. 41–65, jan. 1991. ISSN 03043770. DOI: 10.1016/0304-3770(91)90038-7. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/0304377091900387>>. Citado na p. 38.
- BREVIS, W.; GARCÍA-VILLALBA, M.; NIÑO, Y. Experimental and large eddy simulation study of the flow developed by a sequence of lateral obstacles. **Environmental Fluid Mechanics**, v. 14, n. 4, p. 873–893, 2014. ISSN 15677419. DOI: 10.1007/s10652-013-9328-x. Citado nas pp. 11, 43.
- BUCZYŃSKA, Edyta et al. Human impact on large rivers: the influence of groynes of the River Oder on larval assemblages of caddisflies (Trichoptera). **Hydrobiologia**,

v. 819, n. 1, p. 177–195, 2018. ISSN 15735117. DOI: 10.1007/s10750-018-3636-6. Citado na p. 10.

BUCZYŃSKI, P. et al. Groynes: A factor modifying the occurrence of dragonfly larvae (Odonata) on a large lowland river. **Marine and Freshwater Research**, v. 68, n. 9, p. 1653–1663, 2017. ISSN 13231650. DOI: 10.1071/mf16217. Citado na p. 10.

CELIK, Ismail et al. Procedure of Estimation and Reporting of Uncertainty Due to Discretization in CFD Applications. **J. Fluids Eng.**, v. 130, p. 078001, jul. 2008. DOI: 10.1115/1.2960953. Citado na p. 149.

CHANG, Kyoungsik; CONSTANTINESCU, George; PARK, Seung-O. Analysis of the flow and mass transfer processes for the incompressible flow past an open cavity with a laminar and a fully turbulent incoming boundary layer. **Journal of Fluid Mechanics**, Cambridge University Press, v. 561, p. 113, ago. 2006. ISSN 0022-1120. DOI: 10.1017/s0022112006000735. Disponível em:  
<http://www.journals.cambridge.org/abstract%7B%5C%5F%7DS0022112006000735>. Citado na p. 29.

CHEN, Zhengbing et al. The wake structure behind a porous obstruction and its implications for deposition near a finite patch of emergent vegetation. **Water Resources Research**, v. 48, n. 9, 2012. DOI: <https://doi.org/10.1029/2012WR012224>. Disponível em:  
<https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2012WR012224>. Citado nas pp. 7, 40, 48, 60.

CONSTANTINESCU, George; SUKHODOLOV, Alexander; MCCOY, Andrew. Mass exchange in a shallow channel flow with a series of groynes: Les study and comparison with laboratory and field experiments. **Environmental Fluid Mechanics**, v. 9, n. 6, p. 587–615, 2009. ISSN 15677419. DOI: 10.1007/s10652-009-9155-2. Citado na p. 40.

COTTON, J. A. et al. The effects of seasonal changes to in-stream vegetation cover on patterns of flow and accumulation of sediment. **Geomorphology**, 2006. ISSN 0169555x. DOI: 10.1016/j.geomorph.2006.01.010. Citado na p. 38.

DROST, Kevin J. et al. Parameterization of Mean Residence Times in Idealized Rectangular Dead Zones Representative of Natural Streams. **Journal of Hydraulic Engineering**, American Society of Civil Engineers, v. 140, n. 8, p. 04014035, ago. 2014. ISSN 0733-9429. DOI: 10.1061/(asce)hy.1943-7900.0000879. Disponível em:  
[https://doi.org/10.1061/\(ASCE\)HY.1943-7900.0000879%20http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D29HY.1943-7900.0000879](https://doi.org/10.1061/(ASCE)HY.1943-7900.0000879%20http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D29HY.1943-7900.0000879). Citado nas pp. 3, 38.

- DURÓ, G. et al. Bank Erosion Processes in Regulated Navigable Rivers. **Journal of Geophysical Research: Earth Surface**, v. 125, n. 7, jul. 2020. ISSN 2169-9003. DOI: 10.1029/2019jf005441. Disponível em: <<https://onlinelibrary.wiley.com/doi/10.1029/2019JF005441>>. Citado na p. 39.
- DUTTA, Rabijit; XING, Tao. Five-equation and robust three-equation methods for solution verification of large eddy simulation. **Journal of Hydrodynamics**, v. 30, n. 1, p. 23–33, fev. 2018. ISSN 1001-6058. DOI: 10.1007/s42241-018-0002-0. Disponível em: <<http://link.springer.com/10.1007/s42241-018-0002-0>>. Citado nas pp. 45, 149.
- ELY, Joseph S.; EVANS, Dan K. Plant communities of selected embayments along the mid- to mid-upper ohio river floodplain. **Journal of the Botanical Research Institute of Texas**, 2010. ISSN 19345259. Citado na p. 39.
- ENSIGN, Scott H.; DOYLE, Martin W. In-channel transient storage and associated nutrient retention: Evidence from experimental manipulations. **Limnology and Oceanography**, 2005. ISSN 00243590. DOI: 10.4319/lo.2005.50.6.1740. Citado na p. 38.
- FORREST, Barrie M. et al. Multiple indicators reveal river plume influence on sediments and benthos in a New Zealand coastal embayment. **New Zealand Journal of Marine and Freshwater Research**, 2007. ISSN 11758805. DOI: 10.1080/00288330709509892. Citado na p. 38.
- GOOSEFF, Michael N. et al. Determining in-channel (dead zone) transient storage by comparing solute transport in a bedrock channel-alluvial channel sequence, Oregon. **Water Resources Research**, 2005. ISSN 00431397. DOI: 10.1029/2004wr003513. Citado na p. 38.
- GUALTIERI, Carlo; ANGELOUDIS, Athanasios et al. On the values for the turbulent schmidt number in environmental flows. **Fluids**, v. 2, n. 2, 2017. ISSN 23115521. DOI: 10.3390/fluids2020017. Citado na p. 44.
- GUALTIERI, Carlo; LÓPEZ-JIMÉNEZ, P.; MORA-RODRÍGUEZ, Jesús. Modelling turbulence and solute transport in a square dead zone. In: citado na p. 32.
- HARVEY, J W. Chapter 1 - Hydrologic Exchange Flows and Their Ecological Consequences in River Corridors. In: JONES, Jeremy B; STANLEY, Emily H (Ed.). **Stream Ecosystems in a Changing Environment**. Boston: Academic Press, 2016. p. 1–83. ISBN 978-0-12-405890-3. DOI: <https://doi.org/10.1016/B978-0-12-405890-3.00001-4>. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9780124058903000014>>. Citado nas pp. 3, 38.

- HARVEY, Jud; GOOSEFF, Michael. River corridor science: Hydrologic exchange and ecological consequences from bedforms to basins. **Water Resources Research**, v. 51, n. 9, p. 6893–6922, set. 2015. ISSN 00431397. DOI: 10.1002/2015wr017617. Disponível em: <<http://doi.wiley.com/10.1002/2015WR017617>>. Citado na p. 38.
- HINTERBERGER, C.; FRÖHLICH, J.; RODI, W. Three-Dimensional and Depth-Averaged Large-Eddy Simulations of Some Shallow Water Flows. **Journal of Hydraulic Engineering**, v. 133, n. 8, p. 857–872, ago. 2007. ISSN 0733-9429. DOI: 10.1061/(asce)0733-9429(2007)133:8(857). Disponível em: <<http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D290733-9429%7B%5C%7D282007%7B%5C%7D29133%7B%5C%7D3A8%7B%5C%7D28857%7B%5C%7D29>>. Citado na p. 11.
- JACKSON, T. R.; HAGGERTY, R.; APTE, S. V. A fluid-mechanics based classification scheme for surface transient storage in riverine environments: Quantitatively separating surface from hyporheic transient storage. **Hydrology and Earth System Sciences**, v. 17, n. 7, p. 2747–2779, 2013. ISSN 10275606. DOI: 10.5194/hess-17-2747-2013. Citado nas pp. 2–4, 23, 39.
- JACKSON, Tracie R.; APTE, Sourabh V. et al. Flow structure and mean residence times of lateral cavities in open channel flows: influence of bed roughness and shape. **Environmental Fluid Mechanics**, Springer Netherlands, v. 15, n. 5, p. 1069–1100, 2015. ISSN 15731510. DOI: 10.1007/s10652-015-9407-2. Disponível em: <<http://dx.doi.org/10.1007/s10652-015-9407-2>>. Citado nas pp. 3, 38.
- JACKSON, Tracie R.; HAGGERTY, Roy; APTE, Sourabh V.; COLEMAN, Anthony et al. Defining and measuring the mean residence time of lateral surface transient storage zones in small streams. **Water Resources Research**, v. 48, n. 10, p. 1–20, 2012. ISSN 00431397. DOI: 10.1029/2012wr012096. Citado nas pp. 3, 38.
- JACKSON, Tracie R.; HAGGERTY, Roy; APTE, Sourabh V.; O'CONNOR, Ben L. A mean residence time relationship for lateral cavities in gravel-bed rivers and streams: Incorporating streambed roughness and cavity shape. **Water Resources Research**, v. 49, n. 6, p. 3642–3650, 2013. ISSN 00431397. DOI: 10.1002/wrcr.20272. Citado na p. 38.
- JONES, R. Christian. Recovery of a Tidal Freshwater Embayment from Eutrophication: a Multidecadal Study. **Estuaries and Coasts**, 2020. ISSN 15592731. DOI: 10.1007/s12237-020-00730-3. Citado na p. 39.
- JUEZ, C. et al. Morphological resilience to flow fluctuations of fine sediment deposits in bank lateral cavities. **Advances in Water Resources**, 2018. ISSN 03091708. DOI: 10.1016/j.advwatres.2018.03.004. Citado nas pp. 3, 29.

- JUEZ, Carmelo et al. Transport of suspended sediments under the influence of bank macro-roughness. **Earth Surface Processes and Landforms**, v. 43, n. 1, p. 271–284, jan. 2017. ISSN 01979337. DOI: 10.1002/esp.4243. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/esp.4243>>. Citado na p. 38.
- KRAUS, Richard T.; JONES, R. Christian. Fish abundances in shoreline habitats and submerged aquatic vegetation in a tidal freshwater embayment of the Potomac River. **Environmental Monitoring and Assessment**, 2012. ISSN 01676369. DOI: 10.1007/s10661-011-2192-6. Citado na p. 39.
- KURZ, Marie J. et al. Impacts of water level on metabolism and transient storage in vegetated lowland rivers: Insights from a mesocosm study. **Journal of Geophysical Research: Biogeosciences**, 2017. ISSN 21698961. DOI: 10.1002/2016jg003695. Citado na p. 39.
- LANDWÜST, Christian. Expansion of Proterorhinus marmoratus (Teleostei , Gobiidae) into the River Moselle (Germany). **Folia Zoologica**, v. 55, n. 1, p. 107–111, 2006. Citado na p. 29.
- LU, J.; DAI, H. C. Large eddy simulation of flow and mass exchange in an embayment with or without vegetation. **Applied Mathematical Modelling**, v. 40, n. 17-18, p. 7751–7767, 2016. ISSN 0307904x. DOI: 10.1016/j.apm.2016.03.026. Citado na p. 39.
- MACEINA, Michael J.; SLIPKE, Jeffery W.; GRIZZLE, John M. Effectiveness of Three Barrier Types for Confining Grass Carp in Embayments of Lake Seminole, Georgia. **North American Journal of Fisheries Management**, 1999. ISSN 0275-5947. DOI: 10.1577/1548-8675(1999)019<0968:eotbt>2.0.co;2. Citado na p. 39.
- MCCOY, Andrew; CONSTANTINESCU, George; WEBER, Larry J. Numerical Investigation of Flow Hydrodynamics in a Channel with a Series of Groynes. **Journal of Hydraulic Engineering**, v. 134, n. 2, p. 157–172, 2008. ISSN 0733-9429. DOI: 10.1061/(asce)0733-9429(2008)134:2(157). Disponível em: <<http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D290733-9429%7B%5C%7D282008%7B%5C%7D29134%7B%5C%7D3A2%7B%5C%7D28157%7B%5C%7D29>>. Citado nas pp. 3, 4, 10.
- MEILE, Tobias; BOILLAT, Jean Louis; SCHLEISS, Anton J. Water-surface oscillations in channels with axi-symmetric cavities. **Journal of Hydraulic Research**, 2011. ISSN 00221686. DOI: 10.1080/00221686.2010.534671. Citado na p. 38.
- MENTER, F. R. et al. Transition Modelling for General Purpose CFD Codes. **Engineering Turbulence Modelling and Experiments 6**, August, p. 31–48, 2005. ISSN 1386-6184. DOI: 10.1016/b978-008044544-1/50003-0. Citado na p. 12.

MIGNOT, Emmanuel et al. Measurement of mass exchange processes and coefficients in a simplified open-channel lateral cavity connected to a main stream. **Environmental Fluid Mechanics**, Springer Netherlands, v. 17, n. 3, p. 429–448, 2017. ISSN 15731510. DOI: 10.1007/s10652-016-9495-7. Citado nas pp. 10, 38, 40, 51.

MULHOLLAND, P. J. et al. Effect of periphyton biomass on hydraulic characteristics and nutrient cycling in streams. **Oecologia**, 1994. ISSN 00298549. DOI: 10.1007/bf00326088. Citado na p. 38.

NEPF, Heidi M. Hydrodynamics of vegetated channels. **Journal of Hydraulic Research**, v. 50, n. 3, p. 262–279, 2012. ISSN 00221686. DOI: 10.1080/00221686.2012.696559. Citado nas pp. 38, 44, 55, 58.

NICOUD, Franck; DUCROS, Frédéric. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. **Flow, Turbulence and Combustion**, Springer Verlag (Germany), v. 62, n. 3, p. 183–200, 1999. DOI: 10.1023/a:1009995426001. Disponível em: <<https://hal.archives-ouvertes.fr/hal-00910373>>. Citado na p. 41.

OLDHAM, C. E.; STURMAN, J. J. The effect of emergent vegetation on convective flushing in shallow wetlands: Scaling and experiments. **Limnology and Oceanography**, v. 46, n. 6, p. 1486–1493, 2001. DOI: 10.4319/lo.2001.46.6.1486. eprint: <https://aslopubs.onlinelibrary.wiley.com/doi/pdf/10.4319/lo.2001.46.6.1486>. Disponível em: <<https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lo.2001.46.6.1486>>. Citado nas pp. 31, 42.

OLESEN, Birgit. Regulation of light attenuation and eelgrass *Zostera marina* depth distribution in a Danish embayment. **Marine Ecology Progress Series**, 1996. ISSN 01718630. DOI: 10.3354/meps134187. Citado na p. 39.

OLIVEIRA, Luiz E D de; JANZEN, Johannes G. Mass Exchange in Dead Water Zones: A Numerical Approach. In: **Water, Energy and Food Nexus in the Context of Strategies for Climate Change Mitigation**. Edição: Walter Leal Filho e José Baltazar Salgueirinho de Andrade Guerra. Cham: Springer International Publishing, 2020. p. 59–68. ISBN 978-3-030-57235-8. DOI: 10.1007/978-3-030-57235-8\_5. Disponível em: <<https://doi.org/10.1007/978-3-030-57235-8%7B%5C%5F%7D5>>. Citado nas pp. 3, 59.

OURO, Pablo; JUEZ, Carmelo; FRANCA, Mário. Drivers for mass and momentum exchange between the main channel and river bank lateral cavities. **Advances in Water Resources**, Elsevier Ltd, v. 137, mar. 2020. ISSN 03091708. DOI: 10.1016/j.advwatres.2020.103511. Citado na p. 38.

PANDEY, Manish; AHMAD, Z.; SHARMA, P. K. **Scour around impermeable spur dikes: a review**. [S.l.: s.n.], 2018. DOI: 10.1080/09715010.2017.1342571. Citado na p. 23.

POLETTO, R.; CRAFT, T.; REVELL, A. A New Divergence Free Synthetic Eddy Method for the Reproduction of Inlet Flow Conditions for LES. **Flow, Turbulence and Combustion**, v. 91, n. 3, p. 519–539, 2013. DOI: 10.1007/s10494-013-9488-2. Citado na p. 44.

RIBI, J.-M. et al. Attractiveness of a lateral shelter in a channel as a refuge for juvenile brown trout during hydropeaking. **Aquatic Sciences**, v. 76, n. 4, p. 527–541, 2014. ISSN 1420-9055. DOI: 10.1007/s00027-014-0351-x. Disponível em: <<https://doi.org/10.1007/s00027-014-0351-x>>. Citado nas pp. 3, 38.

SANJOU, Michio; AKIMOTO, Tetsuro; OKAMOTO, Takaaki. Three-dimensional turbulence structure of rectangular side-cavity zone in open-channel streams. **International Journal of River Basin Management**, v. 10, n. 4, p. 293–305, 2012. ISSN 15715124. DOI: 10.1080/15715124.2012.717943. Citado na p. 38.

SCHWARTZ, René; KOZERSKI, Hans-Peter. Entry and Deposits of Suspended Particulate Matter in Groyne Fields of the Middle Elbe and its Ecological Relevance. **Acta hydrochimica et hydrobiologica**, v. 31, 4-5, p. 391–399, 2003. DOI: 10.1002/aheh.200300496. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/aheh.200300496>. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/aheh.200300496>>. Citado nas pp. 2, 10.

SONNENWALD, Fred; GUYMER, Ian; STOVIN, Virginia. Computational fluid dynamics modelling of residence times in vegetated stormwater ponds. **Proceedings of the Institution of Civil Engineers - Water Management**, v. 171, n. 2, p. 76–86, 2017. ISSN 1741-7589. DOI: 10.1680/jwama.16.00117. Citado na p. 25.

SUKHODOLOV, Alexander; UIJTTEWAAL, Wim S J; ENGELHARDT, Christof. On the correspondence between morphological and hydrodynamical patterns of groyne fields. **Earth Surface Processes and Landforms**, v. 27, n. 3, p. 289–305, 2002. ISSN 01979337. DOI: 10.1002/esp.319. Citado na p. 47.

SUKHODOLOV, Alexander N. Hydrodynamics of groyne fields in a straight river reach: insight from field experiments. **Journal of Hydraulic Research**, Taylor & Francis, v. 52, n. 1, p. 105–120, 2014. DOI: 10.1080/00221686.2014.880859. eprint: <https://doi.org/10.1080/00221686.2014.880859>. Disponível em: <<https://doi.org/10.1080/00221686.2014.880859>>. Citado nas pp. 3, 10, 16, 25, 44.

SUKHODOLOV, Alexander N.; SUKHODOLOVA, Tatiana A.; KRICK, Julian. Effects of vegetation on turbulent flow structure in groyne fields. **Journal of Hydraulic Research**, v. 55, n. 1, p. 1–15, 2017. ISSN 00221686. DOI: 10.1080/00221686.2016.1211183. Citado nas pp. 6, 23, 39–41, 60.

SUKHODOLOVA, T. et al. Longitudinal dispersion in a lowland river with submersed vegetation. **Proceedings of the International Conference on Fluvial Hydraulics - River Flow 2006**, v. 1, p. 631–638, ago. 2006. DOI: 10.1201/9781439833865.ch65. Citado na p. 5.

SZLAUER-ŁUKASZEWSKA, Agnieszka. Substrate type as a factor affecting the ostracod assemblages in groyne fields of the Oder River (Poland). **North-Western Journal of Zoology**, v. 11, n. 2, p. 274–287, 2015. ISSN 18435629. Citado na p. 10.

UIJTTEWAAL, By W S J; LEHMANN, D; MAZIJK, a Van. Exchange Processes Between a River and Model Experiments Its Groyne Fields : Model Experiments. **Journal of Hydraulic Engineering**, v. 127, November, p. 928–936, 2001. Citado nas pp. 31, 40, 43.

UIJTTEWAAL, Wim S. Effects of Groyne Layout on the Flow in Groyne Fields: Laboratory Experiments. **Journal of Hydraulic Engineering**, v. 131, n. 9, p. 782–791, 2005. DOI: 10.1061/(asce)0733-9429(2005)131:9(782). eprint: <https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9429%282005%29131%3A9%28782%29>. Disponível em: <<https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9429%282005%29131%3A9%28782%29>>. Citado nas pp. 3, 10, 32.

VANDENBRUWAENE, W. et al. Flow interaction with dynamic vegetation patches: Implications for biogeomorphic evolution of a tidal landscape. **Journal of Geophysical Research: Earth Surface**, v. 116, F1, n/a–n/a, mar. 2011. ISSN 01480227. DOI: 10.1029/2010jf001788. Disponível em: <<http://doi.wiley.com/10.1029/2010JF001788>>. Citado na p. 38.

WARD, Larry G.; MICHAEL KEMP, W.; BOYNTON, Walter R. The influence of waves and seagrass communities on suspended particulates in an estuarine embayment. **Marine Geology**, 1984. ISSN 00253227. DOI: 10.1016/0025-3227(84)90089-6. Citado nas pp. 38, 39.

WATTS, Robyn J.; JOHNSON, Michael S. Estuaries, lagoons and enclosed embayments: Habitats that enhance population subdivision of inshore fishes. **Marine and Freshwater Research**, 2004. ISSN 13231650. DOI: 10.1071/mf04051. Citado na p. 38.

WEITBRECHT, V.; JIRKA, G. Flow Patterns In Dead Zones of Rivers and their Effect On Exchange Processes. In: citado nas pp. 3, 4, 11, 15, 40.

WEITBRECHT, Volker. **Influence of dead-water zones on the dispersive mass-transport in rivers**. 2004. f. 130. 130 f. Tese (Doutorado) – Karlsruhe Institute of Technology, Karlsruhe. ISBN 3-937300-07-4. DOI: 10.5445/ksp/1242004. Disponível em: <<https://publikationen.bibliothek.kit.edu/1242004>>. Citado nas pp. 1–3, 5, 11, 15, 40, 48, 58, 60.

WEITBRECHT, Volker; SOCOLOFSKY, Scott a.; JIRKA, Gerhard H. Experiments on Mass Exchange between Groin Fields and Main Stream in Rivers. **Journal of Hydraulic Engineering**, v. 134, n. 2, p. 173–183, 2008. ISSN 0733-9429. DOI: 10.1061/(asce)0733-9429(2008)134:2(173). Citado na p. 23.

XIANG, Ke; YANG, Zhonghua; HUAI, Wenxin et al. Large eddy simulation of turbulent flow structure in a rectangular embayment zone with different population densities of vegetation. **Environmental Science and Pollution Research**, Environmental Science e Pollution Research, v. 26, n. 14, p. 14583–14597, mai. 2019. ISSN 1614-7499. DOI: 10.1007/s11356-019-04709-x. Disponível em: <<https://doi.org/10.1007/s11356-019-04709-x>>. Citado nas pp. 6, 10, 23–25, 29–32, 39–43, 45, 55, 58.

XIANG, Ke; YANG, Zhonghua; WU, Shiqiang et al. Flow hydrodynamics of the mixing layer in consecutive vegetated groyne fields. **Physics of Fluids**, AIP Publishing, LLC, v. 32, n. 6, p. 1–22, 2020. ISSN 10897666. DOI: 10.1063/5.0006317. Disponível em: <<https://doi.org/10.1063/5.0006317>>. Citado nas pp. 2, 3, 6, 39–41, 51, 52.

YAMASAKI, Taís N. et al. From patch to channel scale: The evolution of emergent vegetation in a channel. **Advances in Water Resources**, 2019. ISSN 03091708. DOI: 10.1016/j.advwatres.2019.05.009. Citado na p. 24.

YOSSEF, Mohamed F. M.; VRIEND, Huib J. de. Flow Details near River Groynes: Experimental Investigation. **Journal of Hydraulic Engineering**, v. 137, n. 5, p. 504–516, mai. 2011. ISSN 0733-9429. DOI: 10.1061/(asce)hy.1943-7900.0000326. Disponível em: <<http://ascelibrary.org/doi/10.1061/%7B%5C%7D28ASCE%7B%5C%7D29HY.1943-7900.0000326>>. Citado na p. 51.

## **Appendix A**

### **Total Turbulent Kinetic Energy Function**

As the default function to write the TKE values does not account for the instantaneous values generated from the LES model, a new function was written to account both the instantaneous and averaged values. The code of the function totalTKE calculates the TKE based on the Resolved Reynolds Stress Tensor took from the instantaneous fluctuation of the velocity ( $u'$ ) and the Subgrid Reynolds Stress Tensor ( $R$ ).

$$totalTKE = 0.5tr(R) + 0.5tr(u') \quad (\text{A.1})$$

---

```

1  totalTKE
2  {
3      type          coded;
4      libs          ("libutilityFunctionObjects.so");
5      name          totalTKE;
6      executeControl timeStep;
7      writeControl   writeTime;
8      timeStart     155;
9      enabled        true;

10
11  /*-----*/
12
13  Total Turbulent Kinetic Energy Evaluation
14  ** Requires fieldAverage Function to Obtain UPrime2Mean**
15  ** Resolved Reynolds Stress Tensor
16  ** Requires turbulenceFields Function to Obtain R**
17  ** Subgrid Reynolds Stress Tensor
18
19  \*-----*/
20
21  codeExecute
22  #{
23      static autoPtr<volScalarField> totalTKE;
24
25      if
26      (
27          mesh().foundObject<volSymmTensorField>("UPrime2Mean")
28          &&
29          mesh().foundObject<volSymmTensorField>("turbulenceProperties:R")
30          &&
31          mesh().foundObject<volScalarField>("totalTKE") == 0
32      )
33      {
34          Info << "Turbulent Kinetic Energy:" << endl;

```

```

35     Info << "  Initialising" << endl;
36     Info << "  Calculating" << nl << endl;
37
38     totalTKE.set
39     (
40         new volScalarField
41         (
42             IOobject
43             (
44                 "totalTKE",
45                 mesh().time().timeName(),
46                 mesh(),
47                 IOobject::NO_READ,
48                 IOobject::AUTO_WRITE
49             ),
50             mesh(),
51             dimensionedScalar
52             (
53                 "totalTKE",
54                 dimensionSet(0,2,-2,0,0,0,0),
55                 0
56             )
57         )
58     );
59
60     const volSymmTensorField& R =
61         mesh().lookupObjectRef<volSymmTensorField>("turbulenceProperties:R");
62
63     const volSymmTensorField& UPrime2Mean =
64         mesh().lookupObjectRef<volSymmTensorField>("UPrime2Mean");
65
66     volScalarField& totalTKE =
67         mesh().lookupObjectRef<volScalarField>("totalTKE");
68     totalTKE = (0.5 * tr(R)) + (0.5 * tr(UPrime2Mean));
69 }
70
71 else if
72 (
73     mesh().foundObject<volSymmTensorField>("UPrime2Mean")
74     &&
75     mesh().foundObject<volSymmTensorField>("turbulenceProperties:R")
76     &&

```

```

73         mesh().foundObject<volScalarField>("totalTKE")
74     )
75     {
76         Info << "Turbulent Kinect Energy:" << endl;
77         Info << " Calculating" << nl << endl;
78
79         const volSymmTensorField& R =
80             mesh().lookupObjectRef<volSymmTensorField>("turbulenceProperties:R");
81         const volSymmTensorField& UPrime2Mean =
82             mesh().lookupObjectRef<volSymmTensorField>("UPrime2Mean");
83
84         volScalarField& totalTKE =
85             mesh().lookupObjectRef<volScalarField>("totalTKE");
86         totalTKE = (0.5 * tr(R)) + (0.5 * tr(UPrime2Mean));
87     }
88
89     else
90     {
91         Info << "Turbulent Kinect Energy:" << endl;
92         Warning << endl
93             << "    Unable to Calculate Turbulent Kinect Energy" << endl
94             << "    UPrime2Mean and/or R Unavailable" << endl
95             << "    Enable fieldAverage and turbulenceFields Functions"
96             << nl << endl;
97     }
98 #};
99 }
```

---

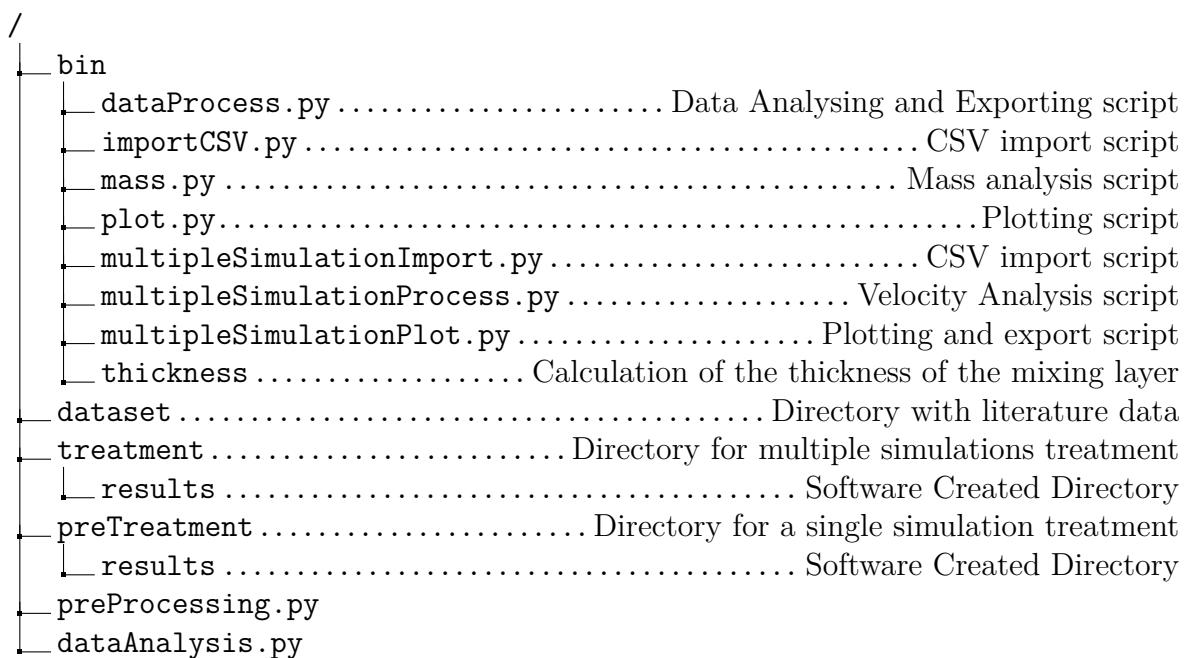
## **Appendix B**

### **Data Processing**

In this appendix the script used to process simulation data is presented. This code uses python to calculate the ensemble averaging properties in a 2D plane. Note, the only process that is not fully automated is the calculation of the mixing length that requires the user to manually fill the maximum value of the absolute velocity gradient in the y direction  $(\partial \bar{u} / \partial y)_{max}$ .

## B.1 File Structure

The file structure of the script is shown below:



The requirements of the script are:

- Python 3.x
- Scipy
- Numpy
- Pandas
- Matplotlib

## B.2 preProcessing.py

The execution of the preProcessing.py script depends on the preTreatment directory that contains the .csv files to be analysed and the dataset directory that contains the csv extracted from literature.

---

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # preProcessing.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24
25 """
26 Main module
27
28 This script analyses the output of simulations ran on OpenFoam
29 The analysis steps are performed by the modules in the bin folder
30 """
31
32 import sys
33 import os
34 import shutil
35 import time
```

```

36 start_time = time.time()
37
38 # Check for necessary directories
39 if not os.path.exists('preTreatment'):
40     os.makedirs('preTreatment')
41     print("The directory preTreatment/ was created, please populate with the "
42           "desired csv files to be analysed.")
43     sys.exit('The directory preTreatment/ did not exist.')
44 elif not os.listdir('preTreatment'):
45     sys.exit('The directory preTreatment/ is empty.')
46
47 # Clear the previous results directories
48 if os.path.exists('preTreatment/results'):
49     shutil.rmtree('preTreatment/results')
50 os.makedirs('preTreatment/results')
51 os.makedirs('preTreatment/results/Excel')
52 os.makedirs('preTreatment/results/CSV')
53 os.makedirs('preTreatment/results/Plot')
54
55 # Define Global Variables
56 H = 0.10
57 U = 0.101
58 W = 0.15
59 L = 0.25
60 Y0 = 0.30
61 X0 = 0.25
62 RHO = 1e-6
63
64 # Import CSV
65 exec(open("bin/importCSV.py").read())
66 print("""Importing Done...
67 Elapsed Time %.3f s\n"""\ %(time.time() - start_time))
68
69 # Data Processing
70 try:
71     exec(open("bin/dataProcess.py").read())
72     print("""Processing Done...
73 Elapsed Time %.3f s\n"""\ %(time.time() - start_time))
74 except:
75     print("""No data was processed.
76 The script jumped into the next section: Mass Fitting""")

```

```

77     print("Elapsed Time %.3f s\n" %(time.time() - start_time))

78

79 # Mass Fitting
80 try:
81     exec(open("bin/mass.py").read())
82     print("""Mass Fitting Done...
83 Elapsed Time %.3f s\n"""\ %(time.time() - start_time))
84 except:
85     print("""No mass data was processed.
86 The script jumped into the next section: Mixing Layer Thickness""")
87     print("Elapsed Time %.3f s\n" %(time.time() - start_time))

88

89 # Mixing Layer Thickness
90 try:
91     exec(open("bin/thickness.py").read())
92     print("""Mixing Layer Thickness Calculated...
93 Elapsed Time %.3f s\n"""\ %(time.time() - start_time))
94 except:
95     print("""No mixing layer thickness data was processed.
96 The script jumped into the next section: Plotting""")
97     print("Elapsed Time %.3f s\n" %(time.time() - start_time))

98

99 # Plot Data
100 try:
101     exec(open("bin/plot.py").read())
102     print("""Plotting Done...
103 Elapsed Time %.3f s\n"""\ %(time.time() - start_time))
104 except:print("No plotting was done.\n")

105

106 print("""All Done...
107 Execution Time %.3f seconds"""\ %(time.time() - start_time))
108 del start_time

```

---

### B.3 dataAnalysis.py

The execution of the dataAnalysis.py script depends on the treatment directory that contains the .csv files to be analysed. These files must be pre processed using the previous script.

---

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  #
4  # dataAnalysis.py
5  #
6  # Copyright 2020 Luiz Oliveira <luiz@luizLinux>
7  #
8  # This program is free software; you can redistribute it and/or modify
9  # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24 """
25 Main module
26
27 This script analyses the output of preProcessing.py
28 The analysis steps are performed by the modules in the bin folder
29 """
30
31 import sys
32 import os
33 import shutil
34 import time
35 start_time = time.time()
36
37 # Check for necessary directories
38 if not os.path.exists('treatment'):
39     os.makedirs('treatment')
40     print("The directory treatment/ was created, please populate with the "
41           "desired csv files to be analysed.")

```

```

42     sys.exit('The directory treatment/ did not exist.')
43 elif not os.listdir('treatment'):
44     sys.exit('The directory treatment/ is empty.')
45
46 # Clear the previous results directories
47 if os.path.exists('treatment/results'):
48     shutil.rmtree('treatment/results')
49 os.makedirs('treatment/results')
50 os.makedirs('treatment/results/Plots')
51 os.makedirs('treatment/results/SelectPlots')
52 os.makedirs('treatment/results/CSV')
53
54 # Define Global Variables
55 H = 0.10
56 U = 0.101
57 W = 0.15
58 L = 0.25
59 Y0 = 0.30
60 X0 = 0.25
61 RHO = 1e-6
62
63 # Import CSV
64 exec(open("bin/multipleSimulationImport.py").read())
65 print("""Importing Done...
66 Elapsed Time %.3f s\n"""\ %(time.time() - start_time))
67
68 # Process Data
69 exec(open("bin/multipleSimulationProcess.py").read())
70 print("""Processing Done...
71 Elapsed Time %.3f s\n"""\ %(time.time() - start_time))
72
73 # Data plot
74 exec(open("bin/multipleSimulationPlot.py").read())
75 print("""Plotting Done...
76 Elapsed Time %.3f s\n"""\ %(time.time() - start_time))
77
78 print("""All Done...
79 Execution Time %.3f seconds"""\ %(time.time() - start_time))
80 del start_time

```

---

## B.4 preProcessing Scripts

### B.4.1 importCSV.py

---

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # importCSV.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24
25 """
26 Data is imported from text files to be later processed and plotted
27 """
28
29 # Libraries
30 import os
31 import re
32 import pandas as pd
33
34 # Import Literature
35 literatureExp = pd.read_csv('dataset/fig4/fig4a.csv', header = 1,
36                             usecols=(0,1))
36 literatureExp.columns = ['(y-y0)/H', 'u/U']
```

```

37 literatureExp = literatureExp.dropna()
38 literatureLES = pd.read_csv('dataset/fig4/fig4a.csv', header = 1,
39     usecols=(2,3))
40 literatureLES.columns = ['(y-y0)/H', 'u/U']
41 massLiterature = pd.read_csv('dataset/mass/mass.csv', header = 1)
42 massLiterature.columns = ['Vegetation Density', 'Td']
43 massLiterature.Td = massLiterature.Td * U / H
44
45 # Tracer data
46 try:
47     tracerData = pd.read_csv('preTreatment/tracerVolAve.dat',
48         delimiter='\t', header = 3)
49     tracerData.columns = ['time', 'tracerVol']
50     tracerData.tracerVol[0] = 1
51     massTimeZero = tracerData.time[0]
52     tracerData.time = tracerData.time - massTimeZero
53 except:pass
54
55 # Partial Tracer at Interface
56 try:
57     interfaceTracer = dict()
58     regions = ['Bottom', 'Middle', 'Top']
59     for reg in regions:
60         interfaceTracer[reg] = pd.read_csv('preTreatment/tracer'+reg+'.dat', \
61             delimiter='\t', header = 4)
62         interfaceTracer[reg].columns = ['time', 'tracer']
63         interfaceTracer[reg].time = interfaceTracer[reg].time - massTimeZero
64     Eraw = pd.read_csv('preTreatment/velocityInterface.dat', delimiter='\t', \
65         header = 4)
66     Eraw.columns = ['time', 'absVelInt']
67     Eraw.time = Eraw.time - massTimeZero
68     Eraw.absVelInt = Eraw.absVelInt/(2*H*L)
69 except:pass
70
71 # Generic Planes
72 files = os.listdir('preTreatment')
73
74 # Check for csv files
75 rawFiles = list()
76 csvFiles = list()
77 datFiles = list()

```

```

77 uniqueRaw = list()
78 uniqueVar = list()
79
80 # Removes ':' from file name
81 for item in files:
82     if ':' in item:
83         newname = item.split(':')[1]
84         os.rename('preTreatment/'+item, 'preTreatment/'+newname)
85
86 files = os.listdir('preTreatment')
87
88 for item in files:
89     if re.search('.\raw', item):
90         rawFiles.append(item)
91
92 for item in files:
93     if re.search('.\csv', item):
94         csvFiles.append(item)
95
96 for item in files:
97     if re.search('.\dat', item):
98         datFiles.append(item)
99
100 csvFiles.sort()
101 datFiles.sort()
102 rawFiles.sort()
103
104 for item in rawFiles:
105     try:
106         plane = re.findall("_([\d\D]..)", item)[0]
107         variableName = re.split("_", item)[0]
108         if plane not in uniqueRaw:
109             uniqueRaw.append(plane)
110         if variableName not in uniqueVar:
111             uniqueVar.append(variableName)
112     except:continue
113
114 def cleanHeader(name):
115     fh = open('preTreatment/'+name, "rt")
116     data = fh.read()
117     # data = re.sub(r':\S+', ' ', data)

```

```

118     data = data.replace('# ', '')
119     data = data.replace(' ', ' ') #removes double spacing
120
121     fh.close()
122     fh = open('preTreatment/'+name, "wt")
123     fh.write(data)
124     fh.close()
125
126 # Import generated data
127 for item in rawFiles:
128     for item2 in uniqueRaw:
129         try:
130             plane = re.findall("_([\d\w]..)", item)[0]
131             if plane == item2:
132                 cleanHeader(item)
133                 variableName = re.split("_", item)[0]
134                 aux = pd.read_csv('preTreatment/'+item, sep=" ", header=1,
135                                   float_precision="high", skipinitialspace=True)
136                 #if aux.isnull().values.any():continue
137                 try:
138                     if variableName not in locals():vars()[variableName] = aux
139                 else:
140                     vars()[variableName] =
141                         pd.concat([vars()[variableName],aux],
142                                   ignore_index=True, axis=1)
143                     vars()[variableName] =
144                         vars()[variableName].dropna(axis=0, how='all')
145                     vars()[variableName] =
146                         vars()[variableName].dropna(axis=1, how='all')
147                 except:continue
148             except:continue
149
150             thickness = dict()
151             for item in csvFiles:
152                 try:
153                     thickness['raw'] = pd.read_csv('preTreatment/'+item, header=0,\n
154                                         float_precision='high')
155                     if len(thickness['raw'].columns) == 8:
156                         thickness['raw'].drop(['Gradients_0','Gradients_1','Gradients_2'],\n
157                                         axis=1, inplace=True)
158                     colNames = ['x', 'y', 'z', 'UMean_X', 'absGradient']

```

```

156     thickness[‘raw’].columns = colNames
157
158     del colNames
159 except:continue
160
161 try:
162     del aux, variableName, item2, plane
163 except:pass
164
165 del files, item, rawFiles, csvFiles, reg

```

---

## B.4.2 dataProcess.py

---

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # dataProcess.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24 """
25 This code processes the data imported from importcsv.py
26 The data is ensembled averaged and then exported to plot.py script
27 """

```

```

28
29 import re
30 import pandas as pd
31 import openpyxl
32 from scipy.interpolate import interp1d
33
34 def dfRename(var, dtf):
35     names = [ 'x' , 'y' , 'z' ]
36     names.extend(var)
37     dtf.columns = names
38
39 def clearLimits(df,x0,x1,y0,y1,z0,z1):
40 # =====
41 #      Clear extra values inside variables.
42 #      This script uses user values of the bound coordinates
43 # =====
44
45     df.drop(df[df.x < x0].index, inplace=True)
46     df.drop(df[df.x > x1].index, inplace=True)
47     df.drop(df[df.y < y0].index, inplace=True)
48     df.drop(df[df.y > y1].index, inplace=True)
49     df.drop(df[df.z < z0].index, inplace=True)
50     df.drop(df[df.z > z1].index, inplace=True)
51
52     return df
53
54 def excelExport(var, name):
55 # =====
56 #      Creates and appends planes into an spreadsheet
57 # =====
58
59     if not os.path.isfile('preTreatment/results/Excel/' +name+ '.xlsx'):
60         wb = openpyxl.Workbook()
61         wb.save('preTreatment/results/Excel/' +name+ '.xlsx')
62
63     with pd.ExcelWriter('preTreatment/results/Excel/' +name+ '.xlsx',
64                         engine="openpyxl", mode='a') as writer:
65         for df_name, df in var.items():
66             df.to_excel(writer, sheet_name=df_name, index=False)
67
68 def csvExport(df, name):
69 # =====

```

```

69 #     Creates and appends planes into separated csv files
70 # =====
71 df.to_csv('preTreatment/results/CSV/' +name+ '.csv')
72
73 def varTreatment(planes, physicalVar, colNames, nColumns, direction,
74                   varName, roundPar, first, last):
75 # =====
76 #     Treats data in an ensemble averaging procedure in the provided direction
77 # =====
78
79     # Local Variable Declaration
80     varDict = dict()
81     kk = first * nColumns
82     startPos = first
83     stopPos = last * nColumns
84     ll = 0
85
86     # Reads all the files and ensemble in a dict in that each ii is a plane
87     for ii in planes:
88         key = ii
89         key = int(re.sub('\D', ' ',key))
90         if key < startPos:
91             continue
92         if kk > stopPos:
93             break
94         for jj in range(nColumns):
95             ll = kk + jj
96             if ll%nColumns == 0: # number of columns
97                 varDict[ii] = physicalVar.iloc[:,ll]
98             else:
99                 varDict[ii] = \
100                     pd.concat([varDict[ii], physicalVar.iloc[:,ll]], axis=1)
101
102         kk = kk + nColumns
103
104         dfRename(colNames, varDict[ii])
105         clearLimits(varDict[ii], 0.25, 0.50, 0.30, 0.45, 0, 0.1)
106         varDict[ii] = varDict[ii].dropna(axis=0, how='all')
107         varDict[ii] = varDict[ii].dropna(axis=1, how='all')
108
109     # Get vector magnitude

```

```

110     if nColumns > 4:
111         df = pd.DataFrame()
112         for i in colNames:
113             df[i] = varDict[ii][i]**2
114         df['mag'] = (df.sum(axis=1))**(1/2)
115         varDict[ii]['mag'] = df.mag
116         expColNames = colNames + ['mag']
117     else:
118         expColNames = colNames
119
120     if direction == "x":
121         varDict[ii] = varDict[ii].drop(columns=['y', 'z'])
122         varDict[ii] = varDict[ii].\
123             groupby(varDict[ii].x.round(roundPar),as_index=False).mean()
124         varDict[ii][direction] = (varDict[ii][direction] - 0.25)/L
125         varDict[ii].columns = [(''+direction+'-x0')+'/'+'L'] + expColNames
126     elif direction == 'y':
127         varDict[ii] = varDict[ii].drop(columns=['x', 'z'])
128         varDict[ii] = varDict[ii].\
129             groupby(varDict[ii].y.round(roundPar),as_index=False).mean()
130         varDict[ii][direction] = (varDict[ii][direction] - 0.30)/H
131         varDict[ii].columns = [(''+direction+'-y0')+'/'+'H'] + expColNames
132     elif direction == 'z':
133         varDict[ii] = varDict[ii].drop(columns=['x', 'y'])
134         varDict[ii] = varDict[ii].\
135             groupby(varDict[ii].z.round(roundPar),as_index=False).mean()
136         varDict[ii][direction] = varDict[ii][direction]/H
137         varDict[ii].columns = [direction+'/'+'H'] + expColNames
138
139     excelExport(varDict, varName+"Dir_"+direction)
140     csvName = varName.split("_")[0]
141     csvExport(varDict[ii], csvName+"_"+ii+"Dir_"+direction)
142
143
144
145 # =====
146 # Planes 0 -> 4
147 # Vertical Planes Varying the Y axis from Y = 0.30 to Y = 0.45
148 # =====
149 ## RMean
150 colNames = ['xx', 'yy', 'zz', 'xy', 'yz', 'xz']

```

```

151 RMean_00_04_Dirx = varTreatment(uniqueRaw, RMean, colNames, 9, 'x',
152                                     'RMean_00_4_', 2, 0, 4)
153 RMean_00_04_Dirz = varTreatment(uniqueRaw, RMean, colNames, 9, 'z',
154                                     'RMean_00_4_', 2, 0, 4)
155
156 ## UMean
157 colNames = ['u', 'v', 'w']
158 UMean_00_04_Dirx = varTreatment(uniqueRaw, UMean, colNames, 6, 'x',
159                                     'UMean_00_4_', 2, 0, 4)
160 UMean_00_04_Dirz = varTreatment(uniqueRaw, UMean, colNames, 6, 'z',
161                                     'UMean_00_4_', 2, 0, 4)
162
163 ## lambVectorMean
164 colNames = ['lambVectorMean_x', 'lambVectorMean_y', 'lambVectorMean_z']
165 lambVectorMean_00_04_Dirx = varTreatment(uniqueRaw, lambVectorMean, colNames,
166                                         6,
167                                         'x', 'lambVectorMean_00_4_', 3, 0, 4)
168 lambVectorMean_00_04_Dirz = varTreatment(uniqueRaw, lambVectorMean, colNames,
169                                         6, 'z', 'lambVectorMean_00_4_', 2, 0, 4)
170
171 ## pMean
172 colNames = ['pMean']
173 pMean_00_04_Dirx = varTreatment(uniqueRaw, pMean, colNames, 4, 'x',
174                                     'pMean_00_4_', 3, 0, 4)
175 pMean_00_04_Dirz = varTreatment(uniqueRaw, pMean, colNames, 4, 'z',
176                                     'pMean_00_4_', 2, 0, 4)
177
178 ## vorticityMean
179 colNames = ['vorticityMean_x', 'vorticityMean_y', 'vorticityMean_z']
180 vorticityMean_00_04_Dirx = varTreatment(uniqueRaw, vorticityMean, colNames, 6,
181                                         'x', 'vorticityMean_00_4_', 3, 0, 4)
182 vorticityMean_00_04_Dirz = varTreatment(uniqueRaw, vorticityMean, colNames, 6,
183                                         'z', 'vorticityMean_00_4_', 2, 0, 4)
184 # =====
185 # Planes 5 -> 11
186 # Vertical Planes Varying the X axis from X = 0.25 to X = 0.50
187 # =====
188 ## RMean
189 colNames = ['xx', 'yy', 'zz', 'xy', 'yz', 'xz']
190 RMean_05_11_Diry = varTreatment(uniqueRaw, RMean, colNames, 9, 'y',

```

```

191                               'RMean_05_11_', 2, 5, 11)
192 RMean_05_11_Dirz = varTreatment(uniqueRaw, RMean, colNames, 9, 'z',
193                               'RMean_05_11_', 2, 5, 11)
194
195 ## UMean
196 colNames = ['u', 'v', 'w']
197 UMean_05_11_Diry = varTreatment(uniqueRaw, UMean, colNames, 6, 'y',
198                               'UMean_05_11_', 2, 5, 11)
199 UMean_05_11_Dirz = varTreatment(uniqueRaw, UMean, colNames, 6, 'z',
200                               'UMean_05_11_', 2, 5, 11)
201
202 ## lambVectorMean
203 colNames = ['lambVectorMean_x', 'lambVectorMean_y', 'lambVectorMean_z']
204 lambVectorMean_05_11_Diry = varTreatment(uniqueRaw, lambVectorMean, colNames,
205                                         6,
206                                         'y', 'lambVectorMean_05_11_', 3, 5, 11)
207 lambVectorMean_05_11_Dirz = varTreatment(uniqueRaw, lambVectorMean, colNames,
208                                         6, 'z', 'lambVectorMean_05_11_', 2, 5, 11)
209
210 ## pMean
211 colNames = ['pMean']
212 pMean_05_11_Diry = varTreatment(uniqueRaw, pMean, colNames, 4, 'y',
213                               'pMean_05_11_', 3, 5, 11)
214 pMean_05_11_Dirz = varTreatment(uniqueRaw, pMean, colNames, 4, 'z',
215                               'pMean_05_11_', 2, 5, 11)
216
217 ## vorticityMean
218 colNames = ['vorticityMean_x', 'vorticityMean_y', 'vorticityMean_z']
219 vorticityMean_05_11_Diry = varTreatment(uniqueRaw, vorticityMean, colNames, 6,
220                                         'y', 'vorticityMean_05_11_', 3, 5, 11)
221 vorticityMean_05_11_Dirz = varTreatment(uniqueRaw, vorticityMean, colNames, 6,
222                                         'z', 'vorticityMean_05_11_', 2, 5, 11)
223 # =====
224 # Planes 12 -> 21
225 # Horizontal Planes Varying the Z axis from Z = 0 to Z = 0.10
226 # =====
227 ## RMean
228 colNames = ['xx', 'yy', 'zz', 'xy', 'yz', 'xz']
229 RMean_12_21_Dirx = varTreatment(uniqueRaw, RMean, colNames, 9, 'x',
230                               'RMean_12_21_', 2, 12, 21)

```

```

231 RMean_12_21_Diry = varTreatment(uniqueRaw, RMean, colNames, 9, 'y',
232                                     'RMean', 2, 12, 21)
233
234 ## UMean
235 colNames = ['u', 'v', 'w']
236 UMean_12_21_Dirx = varTreatment(uniqueRaw, UMean, colNames, 6, 'x',
237                                     'UMean_12_21_', 2, 12, 21)
238 UMean_12_21_Diry = varTreatment(uniqueRaw, UMean, colNames, 6, 'y',
239                                     'UMean_12_21_', 2, 12, 21)
240
241 ## lambVectorMean
242 colNames = ['lambVectorMean_x', 'lambVectorMean_y', 'lambVectorMean_z']
243 lambVectorMean_12_21_Dirx = varTreatment(uniqueRaw, lambVectorMean, colNames,
244                                         6,
245                                         'x', 'lambVectorMean_12_21_', 3, 12, 21)
246 lambVectorMean_12_21_Diry = varTreatment(uniqueRaw, lambVectorMean, colNames,
247                                         6, 'y', 'lambVectorMean_12_21_', 2, 12, 21)
248
249 ## pMean
250 colNames = ['pMean']
251 pMean_12_21_Dirx = varTreatment(uniqueRaw, pMean, colNames, 4, 'x',
252                                     'pMean_12_21_', 3, 12, 21)
253 pMean_12_21_Diry = varTreatment(uniqueRaw, pMean, colNames, 4, 'y',
254                                     'pMean_12_21_', 2, 12, 21)
255
256 ## vorticityMean
257 colNames = ['vorticityMean_x', 'vorticityMean_y', 'vorticityMean_z']
258 vorticityMean_12_21_Dirx = varTreatment(uniqueRaw, vorticityMean, colNames, 6,
259                                         'x', 'vorticityMean_12_21_', 3, 12, 21)
260 vorticityMean_12_21_Diry = varTreatment(uniqueRaw, vorticityMean, colNames, 6,
261                                         'y', 'vorticityMean_5_11_', 2, 12, 21)
262 # =====
263 # Validation Data
264 # =====
265 ## Data Treatment
266 colNames = ['u', 'v', 'w']
267 fig4aOur = varTreatment(['p17'], UMean, colNames, 6, "y", 'UMean_p17_',
268                         3, 17, 17)
269 fig4aOur = fig4aOur['p17']
270 fig4aOur.u = fig4aOur.u/U

```

```

271
272 # Errors from experimental
273 error = dict()
274 error['X'] = literatureExp.iloc[:,0]
275 error['Numerical_u'] =
276     interp1d(fig4aOur.iloc[:,0],fig4aOur.u)(literatureExp.iloc[:,0])
277 error['Experimental_u'] = literatureExp.iloc[:,1]
278
279 error = pd.DataFrame(data=error)
280 error.eval('Error = Experimental_u - Numerical_u', inplace=True)
281 error.eval('Abs_Error = abs(Experimental_u - Numerical_u)', inplace=True)
282 error.eval('Rel_Error = (Experimental_u - Numerical_u)/Experimental_u',
283             inplace=True)
284 error.eval('Abs_Rel_Error = abs((Experimental_u -
285             Numerical_u)/Experimental_u)', inplace=True)
286
287 description = error.describe()
288
289 if not os.path.isfile('preTreatment/results/Excel/validationData.xlsx'):
290     wb = openpyxl.Workbook()
291     wb.save('preTreatment/results/Excel/validationData.xlsx')
292
293 with pd.ExcelWriter('preTreatment/results/Excel/validationData.xlsx',
294                     engine="openpyxl", mode='a') as writer:
295     error.to_excel(writer, sheet_name='Errors', index=False)
296     description.to_excel(writer, sheet_name='Statistical Description')
297
298 del lambVectorMean, pMean, RMean, vorticityMean, colNames

```

---

### B.4.3 mass.py

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # mass.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by

```

```

10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24
25 """
26 Mass quantities are analysed in two ways: by tracer volume and y-velocity
27 """
28
29 # Libraries
30 from datetime import datetime
31 import numpy as np
32 import pandas as pd
33 import openpyxl
34 from scipy.optimize import curve_fit
35
36
37 # Extracting date for report
38 now = datetime.now()
39 today = now.strftime("%d/%m/%Y %H:%M:%S")
40
41 # Define Fitting Function
42 def model(x, td):
43     """
44     First Order Mass Decay Equation
45     """
46     return np.exp(-x/td)
47
48 td, pcov = curve_fit(model, tracerData.time, tracerData.tracerVol, p0=(40),
49                      maxfev=5000)
50

```

```

51 k = W / (td * U)
52
53 modelmass = model(tracerData.time, td)
54
55 tdExp = massLiterature.iloc[1,1]
56
57 tdRelError = ((td - tdExp)/tdExp)*100
58 tdAbsError = tdExp - td
59
60 kexp = W / (tdExp * U) # Non-dimensional experimental value
61 kRelError = ((k - kexp)/kexp)*100
62 kAbsError = kexp - k
63
64 # Mass as function of velocity
65 E = Eraw.absVelInt.mean()
66
67 tdvel = W/E
68 kvel = W / (tdvel * U)
69
70 # Mass Summary
71 file = open("preTreatment/results/massExchange.txt", "w")
72 file.write("Mass Exchange Values (Simulated - Tracer)\n")
73 file.write("ktracer = %.4f\n" %k)
74 file.write("Mean Residence Time = %.2f\n---\n" %td)
75 file.write("Mass Exchange Values (Simulated - Interface Velocity)\n")
76 file.write("kvelocity = %.4f\n" %kvel)
77 file.write("Mean Residence Time = %.2f\n---\n" %tdvel)
78 file.write("Mass Exchange Values (Xiang)\n")
79 file.write("kexp = %.4f\n" %kexp)
80 file.write("Mean Residence Time = %.2f\n---\n" %tdExp)
81 file.write("Error analysis\n")
82 file.write("Relative error\n")
83 file.write("\tError = (Simulated.our - Xiang)/(Xiang)\n")
84 file.write("MRT = %.2f %%\n" %tdRelError)
85 file.write("k = %.2f %%\n" %kRelError)
86 file.write("Absolute error\n")
87 file.write("MRT = %.2f\n" %tdAbsError)
88 file.write("k = %.2f\n" %kAbsError)
89 file.write("---\nData analysed in {} (GMT-4)".format(today))
90 file.close()
91

```

```

92 # Construct mass DataFrame
93 tracerDataExport = tracerData
94 tracerDataExport['modelled'] = modelmass
95 colNames = ['Time', 'Numerical', 'Modelled']
96 tracerDataExport.columns = colNames
97
98 tracerDataExport.to_csv('preTreatment/results/CSV/tracerData.csv')
99 with pd.ExcelWriter('preTreatment/results/Excel/tracerData.xlsx',
100                     engine="openpyxl", mode='w') as writer:
101     for df_name, df in tracerDataExport.items():
102         df.to_excel(writer, sheet_name=df_name, index=False)
103
104 del file, now, today

```

---

#### B.4.4 plot.py

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # plot.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24

```

```

25 """
26 Data is imported from dataProcess.py and plotted into png figures
27 """
28
29 # Libraries
30 import re
31 import matplotlib.pyplot as plt
32 from matplotlib import ticker
33 from matplotlib.offsetbox import AnchoredText
34
35 def plotVar(varName, axis, title, name, col, admensional, first, last):
36 # =====
37 # Runs through all plots from a variable and plots it
38 # =====
39 fig, ax = plt.subplots(figsize=(9,6), dpi=300)
40
41 for key, df in varName.items():
42     nKey = key
43     nKey = int(re.sub('\D', '', nKey))
44     if nKey >= first and nKey <= last:
45         if 'z' not in varName[key].iloc[:,0].name:
46             ax.plot(varName[key].iloc[:,0],
47                     varName[key].iloc[:,col]/admensional, label=key)
48         else:
49             ax.plot(varName[key].iloc[:,col]/admensional,
50                     varName[key].iloc[:,0], label=key)
51
52     ax.legend(loc='best', fontsize='x-large')
53     if title != "None":
54         ax.set_title(title, fontsize='xx-large')
55
56     plt.grid()
57     plt.autoscale(enable=True, tight=True)
58     plt.xlabel(axis[0], fontsize='x-large')
59     plt.ylabel(axis[1], fontsize='x-large')
60     plt.savefig('preTreatment/results/Plot/' + name + '.png', bbox_inches='tight',
61                 format='png')
62     plt.close()
63
64 # =====
65 # Planes 0 -> 4

```

```

66 # Vertical Planes Varying the Y axis from Y = 0.30 to Y = 0.45
67 # =====
68
69 noTitle = "None"
70
71 ## RMean Dir_x_00_04
72 axisNames = ['(x-x0)/L', 'Rmag/U2']
73 plotTitle = 'Time Averaged Reynolds Stresses Magnitude at vertical XZ planes'
74 figureName = 'RMean_mag_Dir_x_00_04'
75 plotVar(RMean_00_04_Dirx, axisNames, noTitle, figureName, 7, U**2, 0, 4)
76
77 ## RMean Dir_x_00_04
78 axisNames = ['Rmag/U2', 'z/H']
79 plotTitle = 'Time Averaged Reynolds Stresses Magnitude at vertical XZ planes'
80 figureName = 'RMean_mag_Dir_x_00_04'
81 plotVar(RMean_00_04_Dirz, axisNames, noTitle, figureName, 7, U**2, 0, 4)
82
83 ## UMean Dir_x_00_04
84 axisNames = ['(x-x0)/L', 'u/U']
85 plotTitle = 'Time Averaged x-velocity at vertical XZ planes'
86 figureName = 'UMean_U_Dir_x_00_04'
87 plotVar(UMean_00_04_Dirx, axisNames, noTitle, figureName, 1, U, 0, 4)
88
89 axisNames = ['(x-x0)/L', 'v/U']
90 plotTitle = 'Time Averaged y-velocity at vertical XZ planes'
91 figureName = 'UMean_V_Dir_x_00_04'
92 plotVar(UMean_00_04_Dirx, axisNames, noTitle, figureName, 2, U, 0, 4)
93
94 axisNames = ['(x-x0)/L', 'w/U']
95 plotTitle = 'Time Averaged z-velocity at vertical XZ planes'
96 figureName = 'UMean_W_Dir_x_00_04'
97 plotVar(UMean_00_04_Dirx, axisNames, noTitle, figureName, 3, U, 0, 4)
98
99 axisNames = ['(x-x0)/L', 'uMag/U']
100 plotTitle = 'Time Averaged velocity magnitude at vertical XZ planes'
101 figureName = 'UMean_mag_Dir_x_00_04'
102 plotVar(UMean_00_04_Dirx, axisNames, noTitle, figureName, 4, U, 0, 4)
103
104 ## UMean Dir_z_00_04
105 axisNames = ['u/U', 'z/H']
106 plotTitle = 'Time Averaged x-velocity at vertical XZ planes'

```

```

107  figureName = 'UMean_U_Dir_z_00_04'
108  plotVar(UMean_00_04_Dirz, axisNames, noTitle, figureName, 1, U, 0, 4)
109
110  axisNames = ['v/U', 'z/H']
111  plotTitle = 'Time Averaged y-velocity at vertical XZ planes'
112  figureName = 'UMean_V_Dir_z_00_04'
113  plotVar(UMean_00_04_Dirz, axisNames, noTitle, figureName, 2, U, 0, 4)
114
115  axisNames = ['w/U', 'z/H']
116  plotTitle = 'Time Averaged z-velocity at vertical XZ planes'
117  figureName = 'UMean_W_Dir_z_00_04'
118  plotVar(UMean_00_04_Dirz, axisNames, noTitle, figureName, 3, U, 0, 4)
119
120  axisNames = ['uMag/U', 'z/H']
121  plotTitle = 'Time Averaged velocity magnitude at vertical XZ planes'
122  figureName = 'UMean_mag_Dir_z_00_04'
123  plotVar(UMean_00_04_Dirz, axisNames, noTitle, figureName, 4, U, 0, 4)
124
125  ## lambVectorMean Dir_x_00_04
126  axisNames = ['(x-x0)/L', 'lambVectorMean [m/s2]']
127  plotTitle = 'Time Averaged Lamb Vector magnitude at vertical XZ planes'
128  figureName = 'lambVectorMean_mag_Dir_x_00_04'
129  plotVar(lambVectorMean_00_04_Dirx, axisNames, noTitle, figureName, 4, 1, 0, 4)
130
131  ## lambVectorMean Dir_z_00_04
132  axisNames = ['lambVectorMean [m/s2]', 'z/H']
133  plotTitle = 'Time Averaged Lamb Vector magnitude at vertical XZ planes'
134  figureName = 'lambVectorMean_mag_Dir_z_00_04'
135  plotVar(lambVectorMean_00_04_Dirz, axisNames, noTitle, figureName, 4, 1, 0, 4)
136
137  ## pMean Dir_x_00_04
138  axisNames = ['(x-x0)/L', r'(pL)/($\rho$ U)']
139  plotTitle = 'Time Averaged Pressure at vertical XZ planes'
140  figureName = 'pMean_Dir_x_00_04'
141  plotVar(pMean_00_04_Dirx, axisNames, noTitle, figureName, 1, L/(RHO*U), 0, 4)
142
143  ## pMean Dir_z_00_04
144  axisNames = [r'(pL)/($\rho$ U)', 'z/H']
145  plotTitle = 'Time Averaged Pressure at vertical XZ planes'
146  figureName = 'pMean_Dir_z_00_04'
147  plotVar(pMean_00_04_Dirz, axisNames, noTitle, figureName, 1, L/(RHO*U), 0, 4)

```

```

148
149 ## vorticity Dir_x_00_04
150 axisNames = ['(x-x0)/L', 'vorticity[1/s]']
151 plotTitle = 'Time Averaged vorticity magnitude at vertical XZ planes'
152 figureName = 'vorticity_Dir_x_00_04'
153 plotVar(vorticityMean_00_04_Dirx, axisNames, noTitle, figureName, 4, 1, 0, 4)
154
155 ## vorticity Dir_z_00_04
156 axisNames = ['vorticity [1/s]', 'z/H']
157 plotTitle = 'Time Averaged vorticity magnitude at vertical XZ planes'
158 figureName = 'vorticity_Dir_z_00_04'
159 plotVar(vorticityMean_00_04_Dirz, axisNames, noTitle, figureName, 4, 1, 0, 4)
160
161 # -----
162 # Planes 5 -> 11
163 # Vertical Planes Varying the X axis from X = 0.25 to X = 0.50
164 # -----
165 ## RMean Dir_y_05_11
166 axisNames = ['Rmag/U2', '(y-y0)/H']
167 plotTitle = 'Time Averaged Reynolds Stresses Magnitude at vertical YZ planes'
168 figureName = 'RMean_mag_Dirx_05_11'
169 plotVar(RMean_05_11_Diry, axisNames, noTitle, figureName, 7, U**2, 5, 11)
170
171 ## RMean Dir_z_05_11
172 axisNames = ['Rmag/U2', 'z/H']
173 plotTitle = 'Time Averaged Reynolds Stresses Magnitude at vertical YZ planes'
174 figureName = 'RMean_mag_Dir_z_05_11'
175 plotVar(RMean_05_11_Dirz, axisNames, noTitle, figureName, 7, U**2, 5, 11)
176
177 ## UMean Dir_y_05_11
178 axisNames = ['(y-y0)/H', 'u/U']
179 plotTitle = 'Time Averaged x-velocity at vertical YZ planes'
180 figureName = 'UMeanYZPlanes_U_Diry'
181 plotVar(UMean_05_11_Diry, axisNames, noTitle, figureName, 1, U, 5, 11)
182
183 axisNames = ['(y-y0)/H', 'v/U']
184 plotTitle = 'Time Averaged y-velocity at vertical YZ planes'
185 figureName = 'UMeanYZPlanes_V_Diry'
186 plotVar(UMean_05_11_Diry, axisNames, noTitle, figureName, 2, U, 5, 11)
187
188 axisNames = ['(y-y0)/H', 'w/U']

```

```

189 plotTitle = 'Time Averaged z-velocity at vertical YZ planes'
190 figureName = 'UMeanYZPlanes_W_Diry'
191 plotVar(UMean_05_11_Diry, axisNames, noTitle, figureName, 3, U, 5, 11)
192
193 axisNames = [ '(y-y0)/H', 'uMag/U' ]
194 plotTitle = 'Time Averaged velocity magnitude at vertical YZ planes'
195 figureName = 'UMeanYZPlanes_mag_Diry'
196 plotVar(UMean_05_11_Diry, axisNames, noTitle, figureName, 4, U, 5, 11)
197
198 ## UMean Dir_z_05_11
199 axisNames = [ 'u/U', 'z/H' ]
200 plotTitle = 'Time Averaged x-velocity at vertical YZ planes'
201 figureName = 'UMeanYZPlanes_U_Dirz'
202 plotVar(UMean_05_11_Dirz, axisNames, noTitle, figureName, 1, U, 5, 11)
203
204 axisNames = [ 'v/U', 'z/H' ]
205 plotTitle = 'Time Averaged y-velocity at vertical YZ planes'
206 figureName = 'UMeanYZPlanes_V_Dirz'
207 plotVar(UMean_05_11_Dirz, axisNames, noTitle, figureName, 2, U, 5, 11)
208
209 axisNames = [ 'w/U', 'z/H' ]
210 plotTitle = 'Time Averaged z-velocity at vertical YZ planes'
211 figureName = 'UMeanYZPlanes_W_Dirz'
212 plotVar(UMean_05_11_Dirz, axisNames, noTitle, figureName, 3, U, 5, 11)
213
214 axisNames = [ 'uMag/U', 'z/H' ]
215 plotTitle = 'Time Averaged velocity magnitude at vertical YZ planes'
216 figureName = 'UMeanYZPlanes_mag_Dirz'
217 plotVar(UMean_05_11_Dirz, axisNames, noTitle, figureName, 4, U, 5, 11)
218
219 ## lambVectorMean Dir_y_05_11
220 axisNames = [ '(x-x0)/L', 'lambVectorMean [m/s2]' ]
221 plotTitle = 'Time Averaged Lamb Vector magnitude at vertical YZ planes'
222 figureName = 'lambVectorMean_mag_Dir_y_05_11'
223 plotVar(lambVectorMean_05_11_Diry, axisNames, noTitle, figureName, 4, 1, 5, 11)
224
225 ## lambVectorMean Dir_z_05_11
226 axisNames = [ 'lambVectorMean [m/s2]', 'z/H' ]
227 plotTitle = 'Time Averaged Lamb Vector magnitude at vertical YZ planes'
228 figureName = 'lambVectorMean_mag_Dir_z_05_11'
229 plotVar(lambVectorMean_05_11_Dirz, axisNames, noTitle, figureName, 4, 1, 5, 11)

```

```

230
231 ## pMean Dir_y_05_11
232 axisNames = ['(x-x0)/L', r'(pL)/($\rho$ U)']
233 plotTitle = 'Time Averaged Pressure at vertical YZ planes'
234 figureName = 'pMean_Dir_y_05_11'
235 plotVar(pMean_05_11_Diry, axisNames, noTitle, figureName, 1, L/(RHO*U), 5, 11)
236
237 ## pMean Dir_z_05_11
238 axisNames = [r'(pL)/($\rho$ U)', 'z/H']
239 plotTitle = 'Time Averaged Pressure at vertical YZ planes'
240 figureName = 'pMean_Dir_z_05_11'
241 plotVar(pMean_05_11_Dirz, axisNames, noTitle, figureName, 1, L/(RHO*U), 5, 11)
242
243 ## vorticity Dir_y_05_11
244 axisNames = ['(x-x0)/L', 'vorticity[1/s]']
245 plotTitle = 'Time Averaged vorticity magnitude at vertical YZ planes'
246 figureName = 'vorticity_Dir_y_05_11'
247 plotVar(vorticityMean_05_11_Diry, axisNames, noTitle, figureName, 4, 1, 5, 11)
248
249 ## vorticity Dir_z_05_11
250 axisNames = ['vorticity [1/s]', 'z/H']
251 plotTitle = 'Time Averaged vorticity magnitude at vertical YZ planes'
252 figureName = 'vorticity_Dir_z_05_11'
253 plotVar(vorticityMean_05_11_Dirz, axisNames, noTitle, figureName, 4, 1, 5, 11)
254
255
256 # =====
257 # Planes 12 -> 21
258 # Horizontal Planes Varying the Z axis from Z = 0 to Z = 0.10
259 # =====
260 ## RMean Dir_x_12_21
261 axisNames = ['(x-x0)/L', 'Rmag/U2']
262 plotTitle = 'Time Averaged Reynolds Stresses Magnitude at horizontal XY planes'
263 figureName = 'RMean_mag_Dir_x_12_21'
264 plotVar(RMean_12_21_Dirx, axisNames, noTitle, figureName, 7, U**2, 12, 21)
265
266 ## RMean Dir_y_12_21
267 axisNames = ['(y-y0)/H', 'Rmag/U2']
268 plotTitle = 'Time Averaged Reynolds Stresses Magnitude at horizontal XY planes'
269 figureName = 'RMean_mag_Dir_y_12_21'
270 plotVar(RMean_12_21_Diry, axisNames, noTitle, figureName, 7, U**2, 12, 21)

```

```

271
272 ## UMean Dir_x_12_21
273 axisNames = [ '(x-x0)/L' , 'u/U' ]
274 plotTitle = 'Time Averaged x-velocity at horizontal XY planes'
275 figureName = 'UMean_U_Dir_x_12_21'
276 plotVar(UMean_12_21_Dirx, axisNames, noTitle, figureName, 1, U, 12, 21)
277
278 axisNames = [ '(x-x0)/L' , 'v/U' ]
279 plotTitle = 'Time Averaged y-velocity at horizontal XY planes'
280 figureName = 'UMean_V_Dir_x_12_21'
281 plotVar(UMean_12_21_Dirx, axisNames, noTitle, figureName, 2, U, 12, 21)
282
283 axisNames = [ '(x-x0)/L' , 'w/U' ]
284 plotTitle = 'Time Averaged z-velocity at horizontal XY planes'
285 figureName = 'UMean_W_Dir_x_12_21'
286 plotVar(UMean_12_21_Dirx, axisNames, noTitle, figureName, 3, U, 12, 21)
287
288 axisNames = [ '(x-x0)/L' , 'uMag/U' ]
289 plotTitle = 'Time Averaged velocity magnitude at horizontal XY planes'
290 figureName = 'UMean_mag_Dir_x_12_21'
291 plotVar(UMean_12_21_Dirx, axisNames, noTitle, figureName, 4, U, 12, 21)
292
293 ## UMean Dir_y_12_21
294 axisNames = [ '(y-y0)/H' , 'u/U' ]
295 plotTitle = 'Time Averaged x-velocity at horizontal XY planes'
296 figureName = 'UMean_U_Dir_y_12_21'
297 plotVar(UMean_12_21_Diry, axisNames, noTitle, figureName, 1, U, 12, 21)
298
299 axisNames = [ '(y-y0)/H' , 'v/U' ]
300 plotTitle = 'Time Averaged y-velocity at horizontal XY planes'
301 figureName = 'UMean_V_Dir_y_12_21'
302 plotVar(UMean_12_21_Diry, axisNames, noTitle, figureName, 2, U, 12, 21)
303
304 axisNames = [ '(y-y0)/H' , 'w/U' ]
305 plotTitle = 'Time Averaged z-velocity at horizontal XY planes'
306 figureName = 'UMean_W_Dir_y_12_21'
307 plotVar(UMean_12_21_Diry, axisNames, noTitle, figureName, 3, U, 12, 21)
308
309 axisNames = [ '(y-y0)/H' , 'uMag/U' ]
310 plotTitle = 'Time Averaged velocity magnitude at horizontal XY planes'
311 figureName = 'UMean_mag_Dir_y_12_21'

```

```

312 plotVar(UMean_12_21_Diry, axisNames, noTitle, figureName, 4, U, 12, 21)
313
314 ## lambVectorMean Dir_y_12_21
315 axisNames = ['(x-x0)/L','lambVectorMean [m/s2]']
316 plotTitle = 'Time Averaged Lamb Vector magnitude at horizontal XY planes'
317 figureName = 'lambVectorMean_mag_Dir_y_12_21'
318 plotVar(lambVectorMean_12_21_Diry, axisNames, noTitle, figureName, 4, 1, 12,
319           21)
320
321 ## lambVectorMean Dir_z_12_21
322 axisNames = ['lambVectorMean [m/s2]', 'z/H']
323 plotTitle = 'Time Averaged Lamb Vector magnitude at horizontal XY planes'
324 figureName = 'lambVectorMean_mag_Dir_z_12_21'
325 plotVar(lambVectorMean_12_21_Diry, axisNames, noTitle, figureName, 4, 1, 12,
326           21)
327
328 ## pMean Dir_y_12_21
329 axisNames = ['(x-x0)/L',r'(pL)/($\rho$ U)']
330 plotTitle = 'Time Averaged Pressure at horizontal XY planes'
331 figureName = 'pMean_Dir_y_12_21'
332 plotVar(pMean_12_21_Diry, axisNames, noTitle, figureName, 1, L/(RHO*U), 12, 21)
333
334 ## pMean Dir_z_12_21
335 axisNames = [r'(pL)/($\rho$ U)', 'z/H']
336 plotTitle = 'Time Averaged Pressure at horizontal XY planes'
337 figureName = 'pMean_Dir_z_12_21'
338 plotVar(pMean_12_21_Diry, axisNames, noTitle, figureName, 1, L/(RHO*U), 12, 21)
339
340 ## vorticity Dir_y_12_21
341 axisNames = ['(x-x0)/L','vorticity[1/s]']
342 plotTitle = 'Time Averaged vorticity magnitude at horizontal XY planes'
343 figureName = 'vorticity_Dir_y_12_21'
344 plotVar(vorticityMean_12_21_Diry, axisNames, noTitle, figureName, 4, 1, 12, 21)
345
346 ## vorticity Dir_z_12_21
347 axisNames = ['vorticity [1/s]', 'z/H']
348 plotTitle = 'Time Averaged vorticity magnitude at horizontal XY planes'
349 figureName = 'vorticity_Dir_z_12_21'
350 plotVar(vorticityMean_12_21_Diry, axisNames, noTitle, figureName, 4, 1, 12, 21)
351 # =====

```

```

351 # Validation Graph
352 # =====
353
354 ## Figure 4
355 fig4, ax4 = plt.subplots(figsize=(9,6), dpi=300)
356 ax4.plot(literatureExp.iloc[:,0], literatureExp.iloc[:,1],'k.',
357 #           label='Experimental (Xiang et al., 2019)')
358 ax4.plot(literatureLES.iloc[:,0], literatureLES.iloc[:,1], '--',
359 #           label='Numerical (Xiang et al., 2019)')
360 #l, caps, c = plt.errorbar(errorbarcsv.iloc[:,1], errorbarcsv.u/U,
361 #                           errorbarcsv.iloc[:,9]/U,
362 #                           elinewidth = 2, capsize = 5, capthick = 1,
363 #                           marker = 'o', markevery=5, errorevery = 5,
364 #                           uplims = True, lolims = True,
365 #                           lw=1.5, aa = True, label='Presented Model')
366 #
367 #for cap in caps:
368 #    cap.set_marker("_")
369 #
370 #ax4.legend(loc='best', fontsize='x-large')
371 #
372 ##ax4.set_title('Time Averaged x-velocity at 0.6H'
373 #                  , fontsize='xx-large')
374 #
375 #plt.grid()
376 #plt.autoscale(enable=True, tight=True)
377 #plt.xlabel('(y-y0)/H', fontsize='x-large')
378 #plt.ylabel('u/U', fontsize='x-large')
379 #plt.savefig('preTreatment/results/Plot/validationWithErrorbar.jpg',
380 #             bbox_inches='tight')

381 # Figure 4
382 fig4, ax4 = plt.subplots(figsize=(9,6), dpi=300)
383 ax4.plot(literatureExp.iloc[:,0], literatureExp.iloc[:,1],'k.',
384 #           label='Experimental (Xiang et al., 2019)')
385 ax4.plot(literatureLES.iloc[:,0], literatureLES.iloc[:,1], '--',
386 #           label='Numerical (Xiang et al., 2019)')
387 ax4.plot(fig4aOur.iloc[:,0], fig4aOur.u,
388 #           label='Presented Model')

389 ax4.legend(loc='best', fontsize='x-large')

```

```

390
391 #ax4.set_title('Time Averaged x-velocity at 0.6H'
392 #           ,fontsize='xx-large')
393
394 plt.grid()
395 plt.autoscale(enable=True, tight=True)
396 plt.xlabel('(y-y0)/H', fontsize='x-large')
397 plt.ylabel('u/U', fontsize='x-large')
398 plt.savefig('preTreatment/results/Plot/validation.jpg', bbox_inches='tight')
399
400 # Mass Decay
401 figm, axm = plt.subplots(figsize=(9,6), dpi=300)
402 axm.plot(tracerData.time, modelmass, label='Fitted Curve', color='r')
403 axm.plot(tracerData.time, tracerData.tracerVol, label='Numerical')
404
405 axm.legend(loc='best', fontsize='x-large')
406
407 #axm.set_title('Mass Ejection from Groyne Field Volume',
408 #               ,fontsize='xx-large')
409
410 at = AnchoredText('C(t)=$C_{0}$$e^{-t/Td}$$\n{k_{adjusted}}$ = %.4f' % k,
411                     prop=dict(size=15), frameon=True,
412                     loc='lower left')
413 axm.add_artist(at)
414
415 #axm.set_yscale('log')
416 plt.autoscale(enable=True, tight=True)
417 plt.grid()
418 plt.xlabel('t [s]', fontsize='x-large')
419 plt.ylabel('Concentration [non-dimensional]', fontsize='x-large')
420 plt.savefig('preTreatment/results/Plot/massDecay.jpg', bbox_inches='tight')
421
422 # Mass Decay semilogy
423 figm, axm = plt.subplots(figsize=(9,6), dpi=300)
424 axm.semilogy(tracerData.time, modelmass, label='Fitted Curve', color='r')
425 axm.semilogy(tracerData.time, tracerData.tracerVol, label='Numerical')
426
427 axm.legend(loc='best', fontsize='x-large')
428
429 #axm.set_title('Mass Ejection from Groyne Field Volume',
430 #               ,fontsize='xx-large')

```

```

431
432 at = AnchoredText('C(t)=$C_{0}$$e^{-t/T_d}$$\n{k_{adjusted}}$ = %.4f' % k,
433 prop=dict(size=15), frameon=True,
434 loc='lower left')
435 axm.add_artist(at)
436
437 axm.yaxis.set_major_formatter(ticker.FormatStrFormatter('%.1f'))
438 axm.yaxis.set_minor_formatter(ticker.FormatStrFormatter('%.1f'))
439 plt.autoscale(enable=True, tight=True)
440 plt.grid()
441 plt.xlabel('t [s]', fontsize='x-large')
442 plt.ylabel('Concentration [non-dimensional]', fontsize='x-large')
443 plt.savefig('preTreatment/results/Plot/massDecaySemiLogY.jpg',
444             bbox_inches='tight')
445
446 del figm, axm, at, fig4, ax4, axisNames, noTitle, figureName , plotTitle
447
448 # Mass Decay per Part
449 figm, axm = plt.subplots(figsize=(9,6), dpi=300)
450 for ii in regions:
451     axm.plot(interfaceTracer[ii].time,interfaceTracer[ii].tracer, label=ii)
452     axm.legend(loc='best', fontsize='x-large')
453
454     axm.yaxis.set_major_formatter(ticker.FormatStrFormatter('%.1f'))
455     axm.yaxis.set_minor_formatter(ticker.FormatStrFormatter('%.1f'))
456     plt.autoscale(enable=True, tight=True)
457     plt.grid()
458     plt.xlabel('t [s]', fontsize='x-large')
459     plt.ylabel('Concentration [non-dimensional]', fontsize='x-large')
460     plt.savefig('preTreatment/results/Plot/massDecayPerPart.jpg',
461                 bbox_inches='tight')
462
463 # Mass Decay per Part semilog y
464 figm, axm = plt.subplots(figsize=(9,6), dpi=300)
465 for ii in regions:
466     axm.semilogy(interfaceTracer[ii].time,interfaceTracer[ii].tracer, label=ii)
467     axm.legend(loc='best', fontsize='x-large')
468
469     axm.yaxis.set_major_formatter(ticker.FormatStrFormatter('%.3f'))

```

```

470 axm.yaxis.set_minor_formatter(ticker.FormatStrFormatter('%.3f'))
471 plt.autoscale(enable=True, tight=True)
472 plt.grid()
473 plt.xlabel('t [s]', fontsize='x-large')
474 plt.ylabel('Concentration [non-dimensional]', fontsize='x-large')
475 plt.savefig('preTreatment/results/Plot/massDecayPerPartSemiLogY.jpg',
476             bbox_inches='tight')

```

---

### B.4.5 thickness.py

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # thickness.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24
25 """
26 Data related to the mixing layer thickness is calculated in this module
27 """
28
29 import re
30 import numpy as np

```

```

31 import pandas as pd
32
33 def clearLimits(df,x0,x1,y0,y1,z0,z1):
34 # =====
35 #     Clear extra values inside variables.
36 #     This script uses user values of the bound coordinates
37 # =====
38
39     df.drop(df[df.x < x0].index, inplace=True)
40     df.drop(df[df.x > x1].index, inplace=True)
41     df.drop(df[df.y < y0].index, inplace=True)
42     df.drop(df[df.y > y1].index, inplace=True)
43     df.drop(df[df.z < z0].index, inplace=True)
44     df.drop(df[df.z > z1].index, inplace=True)
45
46     return df
47
48
49
50
51
52 def dfRename(var, dtf):
53     names = [ 'x', 'y', 'z' ]
54     names.extend(var)
55     dtf.columns = names
56
57
58
59
60
61
62
63
64
65
66 def ui(planes, physicalVar, colNames, nColumns, first, last):
67 # =====
68 #     Treats data in an ensemble averaging procedure in the provided direction
69 # =====
70
71     # Local Variable Declaration

```

```

72     varDict = dict()
73
74     kk = first * nColumns
75
76     startPos = first
77
78     stopPos = last * nColumns
79
80     ll = 0
81
82
83     # Reads all the files and ensemble in a dict in that each ii is a plane
84     for ii in planes:
85
86         key = ii
87
88         key = int(re.sub('\D', ' ',key))
89
90         if key < startPos:
91
92             continue
93
94         if kk > stopPos:
95
96             break
97
98         for jj in range(nColumns):
99
100            ll = kk + jj
101
102            if ll%nColumns == 0: # number of columns
103
104                varDict[ii] = physicalVar.iloc[:,ll]
105
106            else:
107
108                varDict[ii] = \
109                    pd.concat([varDict[ii], physicalVar.iloc[:,ll]], axis=1)
110
111            kk = kk + nColumns
112
113
114            dfRename(colNames, varDict[ii])
115
116            clearLimits(varDict[ii], 0.25, 0.50, 0, 0.45, 0, 0.1)
117
118            varDict[ii] = varDict[ii].dropna(axis=0, how='all')
119
120            varDict[ii] = varDict[ii].dropna(axis=1, how='all')
121
122            varDict[ii].drop(columns=['y', 'v', 'w'], inplace=True)
123
124        return varDict['p00']
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
857
858
859
859
860
861
862
863
864
865
866
867
867
868
869
869
870
871
872
873
874
875
876
877
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
896
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
967
968
969
969
970
971
972
973
974
975
976
977
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1066
1067
1068
1068
1069
1070
1071
1072
1073
1074
1075
1075
1076
1077
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1135
1136
1137
1137
1138
1139
1139
1140
1141
1142
1143
1143
1144
1145
1145
1146
1147
1147
1148
1149
1149
1150
1151
1152
1153
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1162
1163
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1171
1172
1173
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1181
1182
1183
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1191
1192
1193
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1201
1202
1203
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1211
1212
1213
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1831
1832
1833
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1841
1842
1843
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1851
1852
1853
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1861
1862
1863
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1871
1872
1873
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1881
1882
1883
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1891
1892
1893
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
1900
1901
1901
1902
1903
1903
1904
1905
1905
1906
1907
1907
1908
1909
1909
1910
1911
1911
1912
1913
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921
1921
1922
1923
1923
1924
1925
1925
1926
1927
1927
1928
1929
1929
1930
1931
1931
1932
1933
1933
1934
1935
1935
1936
1937
1937
1938
1939
1939
1940
1941
1941
1942
1943
1943
1944
1945
1945
1946
1947
1947
1948
1949
1949
1950
1951
1951
1952
1953
1953
1954
1955
1955
1956
1957
1957
1958
1959
1959
1960
1961
1961
1962
1963
1963
1964
1965
1965
1966
1967
1967
1968
1969
1969
1970
1971
1971
1972
1973
1973
1974
1975
1975
1976
1977
1977
1978
1979
1979
1980
1981
1981
1982
1983
1983
1984
1985
1985
1986
1987
1987
1988
1989
1989
1990
1991
1991
1992
1993
1993
1994
1995
1995
1996
1997
1997
1998
1999
1999
2000
2001
2001
2002
2003
2003
2004
2005
2005
2006
2007
2007
2008
2009
2009
2010
2011
2011
2012
2013
2013
2014
2015
2015
2016
2017
2017
2018
2019
2019
2020
2021
2021
2022
2023
2023
2024
2025
2025
2026
2027
2027
2028
2029
2029
2030
2031
2031
2032
2033
2033
2034
2035
2035
2036
2037
2037
2038
2039
2039
2040
2041
2041
2042
2043
2043
2044
2045
2045
2046
2047
2047
2048
2049
2049
2050
2051
2051
2052
2053
2053
2054
2055
2055
2056
2057
2057
2058
2059
2059
2060
2061
2061
2062
2063
2063
2064
2065
2065
2066
2067
2067
2068
2069
2069
2070
2071
2071
2072
2073
2073
2074
2075
2075
2076
2077
2077
2078
2079
2079
2080
2081
2081
2082
2083
2083
2084
2085
2085
2086
2087
2087
2088
2089
2089
2090
2091
2091
2092
2093
2093
2094
2095
2095
2096
2097
2097
2098
2099
2099
2100
2101
2101
2102
2103
2103
2104
2105
2105
2106
2107
2107
2108
2109
2109
2110
2111
2111
2112
2113
2113
2114
2115
2115
2116
2117
2117
2118
2119
2119
2120
21
```

```

113 xMax = max(thickness['raw'].x)
114 zMax = max(thickness['raw'].z)
115
116 xtol = round(xMax/((numXPlanes + 1)*16), 6)
117 ztol = round(zMax/((numZPlanes + 1)*16), 6)
118
119 zz = zMax/(numZPlanes + 1)
120 aux = thickness['raw']
121 for ii in range(numZPlanes):
122     xx = 0
123     nameZ = 'z' + str(ii)
124     thickness[nameZ] = dict()
125     for jj in range(numXPlanes+2): #Origin and Destination
126         nameX = 'x' + str(jj)
127         xlim = [xx-xtol, xx+xtol]
128         zlim = [zz-ztol, zz+ztol]
129         thickness[nameZ][nameX] = dict()
130         aux2 = aux[np.logical_and(\n
131                         np.logical_and(aux['z'] > zlim[0], aux['z'] < zlim[1]),\n
132                         np.logical_and(aux['x'] > xlim[0], aux['x'] < xlim[1]))]
133         thickness[nameZ][nameX]['cav'] = aux2[aux2['y'] > 0.3].mean()
134         thickness[nameZ][nameX]['channel'] = aux2[aux2['y'] < 0.3].mean()
135         thickness[nameZ][nameX]['absGradient'] = max(aux2['absGradient'])
136
137         xx = xx + xMax/(numXPlanes + 1)
138         zz = zz + zMax/(numZPlanes + 1)
139
140 del ii, jj, aux, aux2, xx, zz, nameX, nameZ, xlim, zlim
141
142 # Organise data by planes
143 aux = thickness
144 thickness = dict()
145 zz = zMax/(numZPlanes + 1)
146 for ii in range(numZPlanes):
147     xx = 0
148     nameZ = 'z' + str(ii)
149     thickness[nameZ] = dict()
150     for jj in range(numXPlanes+2):
151         nameX = 'x' + str(jj)
152         if 'Ue' not in thickness[nameZ].keys():

```

```

153 thickness[nameZ] ['Ue'] =
154     aux[nameZ][nameX]['cav'].to_frame().transpose()
155 thickness[nameZ] ['Um'] =
156     aux[nameZ][nameX]['channel'].to_frame().transpose()
157 thickness[nameZ] ['maxGrad'] = dict() #k: x coord v: maxGrad
158 else:
159     thickness[nameZ] ['Ue'] = thickness[nameZ] ['Ue'] \
160         .append(aux[nameZ][nameX]['cav'].to_frame().transpose(), \
161             ignore_index = True)
162     thickness[nameZ] ['Um'] = thickness[nameZ] ['Um'] \
163         .append(aux[nameZ][nameX]['channel'].to_frame().transpose(), \
164             ignore_index = True)
165 thickness[nameZ] ['maxGrad'][jj] = [aux[nameZ][nameX]['absGradient']]
166 xx = xx + xMax/(numXPlanes + 1)
167 thickness[nameZ] ['Ue'].drop(columns=['y', 'absGradient'], inplace = True)
168 thickness[nameZ] ['Um'].drop(columns=['y', 'absGradient'], inplace = True)
169 ue = ['x', 'z', 'Ue']
170 um = ['x', 'z', 'Um']
171 thickness[nameZ] ['U'] = thickness[nameZ] ['Ue']
172 thickness[nameZ] ['U'] ['Um'] = thickness[nameZ] ['Um'] ['Um']
173 thickness[nameZ] ['maxGrad'] =
174     pd.DataFrame(data=thickness[nameZ] ['maxGrad'])
175 thickness[nameZ] = thickness[nameZ] ['U'].join(thickness[nameZ] ['maxGrad'] .\
176         transpose())
177 colNames = ['x', 'z', 'Ue', 'Um', 'maxGrad']
178 thickness[nameZ].columns = colNames
179 zz = zz + zMax/(numZPlanes + 1)
180
181 del ii, jj, ue, um, colNames
182
183 # Calculates and appends Ui
184 colNames = ['u', 'v', 'w']
185 Uinterface = ui(uniqueRaw, UMean, colNames, 6, 0, 0)
186 Uinterface.x = Uinterface.x - 0.25
187
188 del colNames, UMean
189 zz = zMax/(numZPlanes + 1)
190 aux = Uinterface

```

```

191 Uinterface = dict()
192 for ii in range(numZPlanes):
193     xx = 0
194     nameZ = 'z' + str(ii)
195     Uinterface[nameZ] = dict()
196     for jj in range(numXPlanes+2): #Origin and Destination
197         nameX = 'x' + str(jj)
198         xlim = [xx-xtol, xx+xtol]
199         zlim = [zz-ztol, zz+ztol]
200         aux2 = aux[np.logical_and(\n
201                         np.logical_and(aux['z'] > zlim[0], aux['z'] < zlim[1]),\n
202                         np.logical_and(aux['x'] > xlim[0], aux['x'] < xlim[1]))]
203         Uinterface[nameZ][nameX] = aux2.mean()
204
205         xx = xx + xMax/(numXPlanes + 1)
206         zz = zz + zMax/(numZPlanes + 1)
207
208 del ii, jj, aux, aux2, xx, zz, xtol, ztol, nameX, nameZ, xlim, zlim
209
210 # Organise data by planes
211 aux = Uinterface
212 Uinterface = dict()
213 zz = zMax/(numZPlanes + 1)
214 for ii in range(numZPlanes):
215     xx = 0
216     nameZ = 'z' + str(ii)
217     Uinterface[nameZ] = dict()
218     for jj in range(numXPlanes+2):
219         nameX = 'x' + str(jj)
220         Uinterface[nameZ][jj] = [aux[nameZ][nameX]['u']]
221         xx = xx + xMax/(numXPlanes + 1)
222
223 Uinterface[nameZ] = pd.DataFrame(data=Uinterface[nameZ]).transpose()
224 try:
225     thickness[nameZ].insert(3, 'Ui', Uinterface[nameZ])
226     thickness[nameZ].eval('internalThickness = (Ui-Ue)/maxGrad',
227                           inplace=True)
228     thickness[nameZ].eval('externalThickness = (Um-Ui)/maxGrad',
229                           inplace=True)
230     thickness[nameZ].eval('totalThickness = internalThickness +\n'
231                           'externalThickness', \

```

```

229         inplace=True)
230     thickness[nameZ].eval('deltaInPerW = internalThickness/@W',
231                           inplace=True)
232     thickness[nameZ].eval('deltaOutPerW = externalThickness/@W',
233                           inplace=True)
234     thickness[nameZ].eval('deltaTotalPerW = totalThickness/@W',
235                           inplace=True)
236     except:pass
237
238     zz = zz + zMax/(numZPlanes + 1)
239
240
241     del ii, jj, zz, aux, Uinterface, nameX, nameZ
242
243
244     # Save to Excel
245     excelExport(thickness, 'thickness')

```

---

## B.5 dataAnalysis Scripts

### B.5.1 multipleSimulationImport.py

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Libraries
5  import os
6  import re
7  import pandas as pd
8  import openpyxl
9
10 files = os.listdir('treatment')
11 folder = os.path.abspath('treatment')
12
13 # Check for csv files
14 csvFiles = list()
15 for item in files:
16     if re.search('.\csv', item):
17         csvFiles.append(item)
18
19 # Check for xlsx files
20 xlsxFiles = list()

```

```

21 for item in files:
22     if re.search('.\xlsx', item):
23         xlsxFiles.append(item)
24
25 # Check for txt files
26 txtFiles = list()
27 for item in files:
28     if re.search('.\txt', item):
29         txtFiles.append(item)
30
31 # Import generated data
32 uniqueSim = list()
33 uniqueVar = list()
34 xlsxVar = list()
35 direction = list()
36 planes = list()
37 data = dict()
38
39 for item in csvFiles:
40     try:
41         sim = re.split("_", item)[0]
42         variableName = re.split("_", item)[1]
43         if variableName[-4:] == '.csv':
44             variableName = variableName[:-4]
45
46         if sim not in uniqueSim:
47             uniqueSim.append(sim)
48         if variableName not in uniqueVar:
49             uniqueVar.append(variableName)
50         try:
51             plane = re.split("_", item)[2]
52             axis = re.split("_", item)[4]
53             axis = axis[:-4] #Removes '.csv'
54             if axis not in direction:
55                 direction.append(axis)
56             if plane not in planes:
57                 planes.append(plane)
58                 del variableName, axis, plane
59         except:continue
60     except:continue
61

```

```

62 for item in xlsxFiles:
63     try:
64         sim = re.split("_", item)[0]
65         variableName = re.split("_", item)[1]
66         variableName = variableName[:-5]
67         if sim not in uniqueSim:
68             uniqueSim.append(sim)
69         if variableName not in xlsxVar:
70             xlsxVar.append(variableName)
71     except:
72         continue
73
74 for item in txtFiles:
75     file = open(os.path.join(folder, item), "r")
76     for line in file:
77         if re.search('ktracer.', line):
78             words = line.split()
79             ktracer = float(words[2])
80             continue
81         elif re.search('kvelocity.', line):
82             words = line.split()
83             kvelocity = float(words[2])
84
85     d = {'Simulation':[re.split("_", item)[0]], 'kTracer':[ktracer],
86           'kVelocity':[kvelocity]}
87     df = pd.DataFrame(data=d)
88
89     if 'massExchange' in locals() or 'massExchange' in globals():
90         massExchange = massExchange.append(df, ignore_index=True)
91     else:
92         massExchange = df
93
94     del item, d, df, words, line
95     del ktracer, kvelocity
96
97     tracerData = dict()
98     for var in uniqueVar:
99         if var != 'tracerData.csv':
100             data[var] = dict()
101             for sim in uniqueSim:
102                 data[var][sim] = dict()

```

```

102     if var == 'tracerData':
103         file = sim+"_tracerData.csv"
104         pathToFile = os.path.join(folder, file)
105         if os.path.exists(pathToFile):
106             tracerData[sim] = pd.read_csv(pathToFile, index_col=0,
107                                         float_precision="high")
108     else:
109         for plane in planes:
110             data[var][sim][plane] = dict()
111             for axis in direction:
112                 file = sim+"_"+var+"_"+plane+"_Dir_"+axis+".csv"
113                 pathToFile = os.path.join(folder, file)
114                 if os.path.exists(pathToFile):
115                     data[var][sim][plane][axis] = pd.read_csv(pathToFile,\n
116                                                 index_col=0, float_precision="high")
117
118     thickness = dict()
119     for sim in uniqueSim:
120         file = sim+"_thickness.xlsx"
121         pathToFile = os.path.join(folder, file)
122         if os.path.exists(pathToFile):
123             thickness[sim] = pd.read_excel(pathToFile)
124
125 #del files, txtFiles, file, csvFiles, plane, var, sim, axis, pathToFile,
126 direction

```

---

## B.5.2 multipleSimulationProcess.py

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5  import openpyxl
6  import pandas as pd
7
8  # Append densities to mass exchange
9  try:
10     densities = pd.read_csv(os.path.join(folder, 'densities.csv'), index_col=0)
11 except:
12     print("Imported Simulations:\n",uniqueSim)

```

```

13     print("Please enter the vegetation density of each simulation:")
14     density = dict()
15     for sim in uniqueSim:
16         density[sim] = float(input(sim+":"))
17     densities = pd.DataFrame.from_dict(density, orient = 'index')
18     densities.reset_index(level=0, inplace=True)
19     colName = ['Simulation', 'Density']
20     densities.columns = colName
21     densities.to_csv(os.path.join(folder, 'densities.csv'))
22     del sim, colName
23
24 try:
25     massExchange.insert(1, 'Veg. Density', densities['Density'])
26 except:pass
27
28 massExchange.style.format({'Veg. Density': "{:.4%}"})
29 massExchange.sort_values(by=['Veg. Density'], inplace=True)
30
31 massExchange['Case'] = range(len(massExchange))
32
33 # Retrieve mean residence time
34 massExchange.eval('mrtTracer = 1/kTracer', inplace=True)
35 massExchange.eval('mrtVelocity = 1/kVelocity', inplace=True)
36
37 fileName = os.path.join(folder, 'results/CSV/massExchange.xlsx')
38 massExchange.to_excel(fileName, index=False)
39
40 densities.sort_values(by=['Density'], inplace=True)
41 densities.reset_index(drop=True, inplace=True)
42 del fileName
43
44 # Mixing Layer Thickness
45 try:
46     for sim in uniqueSim:
47         thickness[sim].eval('xL = x/0.25', inplace=True)
48         thickness[sim].rename(columns={'xL': '(x-x0)/L'}, inplace=True)
49 except:pass

```

---

### B.5.3 multipleSimulationPlot.py

---

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # multipleSimulationPlot.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24
25 """
26 Data is imported from multipleSimulationProcess.py and plotted into jpg figures
27 """
28
29 # Libraries
30 import os
31 import matplotlib.pyplot as plt
32
33 #plt.rcParams.update({
34 #    "text.usetex": True,
35 #    "font.family": "sans-serif",
36 #    "font.sans-serif": ["Helvetica"]})
37
38 figFolder = os.path.abspath('treatment/results/Plots')
39 selFigFolder = os.path.abspath('treatment/results/SelectPlots')
40
```

```

41 def plotVar(varName, axis, title, col, admensional):
42 # =====
43 #     Runs through all plots from a variable and plots it
44 # =====
45
46     for plane in planes:
47         anySim = uniqueSim[0]
48         for direction in data[varName][anySim][plane].keys():
49             fig, ax = plt.subplots(figsize=(9,6), dpi=300)
50             for sim in uniqueSim:
51                 df = data[varName][sim][plane][direction]
52                 lbl = densities.loc[densities['Simulation'] == sim]
53                 lbl = lbl['Density'].iloc[0]
54                 ax.plot(df.iloc[:,0], df.iloc[:,col]/admensional,
55                         label='{:,.4%}'.format(lbl))
56
57             ax.legend(loc='best', fontsize='x-large')
58             if title != "None":
59                 ax.set_title(title, fontsize='xx-large')
60             if direction == 'x':
61                 axis[0] = '(x-x0)/L'
62             elif direction == 'y':
63                 axis[0] = '(y-y0)/H'
64             elif direction == 'z':
65                 axis[0] = 'z/H'
66
67             plt.grid()
68             plt.autoscale(enable=True, tight=True)
69             plt.xlabel(axis[0], fontsize='x-large')
70             plt.ylabel(axis[1], fontsize='x-large')
71
72             # Save the image in memory in JPG format
73             figName = varName+'_'+plane+'_Dir_'+direction+'.jpg'
74             figName = os.path.join(figFolder, figName)
75             plt.savefig(figName, box_inches='tight')
76             plt.close()
77
78 # =====
79 # Variables
80 # =====
81

```

```

82 noTitle = "None"
83
84 ## RMean
85 axisNames = ['(x-x0)/L', 'Rmag/U2']
86 plotTitle = 'Time Averaged Reynolds Stresses Magnitude'
87 plotVar('RMean', axisNames, noTitle, 7, U**2)
88
89 ## UMean
90 axisNames = ['(x-x0)/L', 'uMag/U']
91 plotTitle = 'Time Averaged velocity magnitude'
92 plotVar('UMean', axisNames, noTitle, 4, U)
93
94 axisNames = ['(x-x0)/L', 'u/U']
95 plotTitle = 'Time Averaged velocity magnitude'
96 plotVar('UMean', axisNames, noTitle, 1, U)
97
98 axisNames = ['(x-x0)/L', 'v/U']
99 plotTitle = 'Time Averaged velocity magnitude'
100 plotVar('UMean', axisNames, noTitle, 2, U)
101
102 axisNames = ['(x-x0)/L', 'w/U']
103 plotTitle = 'Time Averaged velocity magnitude'
104 plotVar('UMean', axisNames, noTitle, 3, U)
105
106 ## lambVectorMean
107 axisNames = ['(x-x0)/L', 'lambVectorMag']
108 plotTitle = 'Time Averaged Reynolds Stresses Magnitude'
109 plotVar('lambVectorMean', axisNames, noTitle, 4, 1)
110
111 ## pMean
112 axisNames = ['(x-x0)/L', r'(pL)/($\rho$ U)']
113 plotTitle = 'Time Averaged Reynolds Stresses Magnitude'
114 plotVar('pMean', axisNames, noTitle, 1, L/(RHO*U))
115
116 ## vorticity
117 axisNames = ['z/H', 'vorticity [1/s]']
118 plotTitle = 'Time Averaged vorticity magnitude'
119 plotVar('vorticityMean', axisNames, noTitle, 4, 1)
120
121 # =====#
122 # Mass Exchange

```

```

123 # =====
124 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
125 ax1 = ax.twinx()
126
127 #for sim in uniqueSim:
128 #    case = massExchange.loc[massExchange['Simulation'] == sim]
129 #    vDensity = case['Veg. Density'].iloc[0]*100
130 #    caseName = 'Case '+str(densities.loc[densities['Simulation'] == sim].index[0])
131
132 ln1 = ax.plot(massExchange['Veg. Density']*100, massExchange['kTracer'], 'bo',
133                 label=r'$k_{DZ}$', lw=2, ms=6)
134 ln2 = ax1.plot(massExchange['Veg. Density']*100, massExchange['mrtTracer'],
135                  'ks',
136                  label=r'$T_{DZ}$', lw=2, ms=5)
137
138 # Primary Axis
139 #ax.set_xlabel('Vegetation Density [%]', fontsize='x-large')
140 #ax.set_ylabel('Mass Exchange Coefficient
141 #                [non-dimensional]', fontsize='x-large')
142 ax.set_xlabel('a [%]', fontsize='x-large')
143 ax.set_ylabel('k [non-dimensional]', fontsize='x-large')
144 ax.set_xlim(0, 11)
145
146 # Secondary Axis
147 ax1.set_ylabel(r'$T_{DZ}$ [s]', fontsize='x-large')
148
149 # Legend
150 lns = ln1+ln2
151 labs = [l.get_label() for l in lns]
152 ax.legend(lns, labs, loc=7)
153
154 #plt.legend(bbox_to_anchor=(1.15,1), loc="upper left")
155 #plt.tight_layout(rect=[0,0,0.75,1])
156 #ax.set_title('Mass Exchange variation through all vegetation densities',
157 #              fontsize='xx-large')
158 #plt.subplots_adjust(right=0.7)
159
160 # Save the image in memory in JPG format

```

```

161 figName = 'massExchange.jpg'
162 figName = os.path.join(selFigFolder, figName)
163 plt.savefig(figName, box_inches='tight')
164 plt.close()
165
166 del lns, ln1, ln2, ax, fig, labs
167
168 # =====
169 # Tracer decay
170 # =====
171 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
172
173 ln0 = ax.plot(tracerData['x068']['Time'],
174                 tracerData['x068']['Numerical'],
175                 '--', label='Case 0', lw=2, ms=6)
176 ln1 = ax.plot(tracerData['x062']['Time'],
177                 tracerData['x062']['Numerical'],
178                 '--', label='Case 1', lw=2, ms=6)
179 ln2 = ax.plot(tracerData['x063']['Time'],
180                 tracerData['x063']['Numerical'],
181                 '--', label='Case 2', lw=2, ms=6)
182 ln3 = ax.plot(tracerData['x064']['Time'],
183                 tracerData['x064']['Numerical'],
184                 ':', label='Case 3', lw=2, ms=6)
185 ln4 = ax.plot(tracerData['x065']['Time'],
186                 tracerData['x065']['Numerical'],
187                 '-.', label='Case 4', lw=2, ms=6)
188 ln5 = ax.plot(tracerData['x066']['Time'],
189                 tracerData['x066']['Numerical'],
190                 '--', label='Case 5', lw=2, ms=6)
191 ln6 = ax.plot(tracerData['x067']['Time'],
192                 tracerData['x067']['Numerical'],
193                 '--', label='Case 6', lw=2, ms=6)
194 ln7 = ax.plot(tracerData['x115']['Time'],
195                 tracerData['x115']['Numerical'],
196                 '--', label='Case 7', lw=2, ms=6)
197 ln8 = ax.plot(tracerData['x116']['Time'],
198                 tracerData['x116']['Numerical'],
199                 ':', label='Case 8', lw=2, ms=6)
200 ln9 = ax.plot(tracerData['x117']['Time'],
201                 tracerData['x117']['Numerical'],

```

```

202             '_.', label='Case 9', lw=2, ms=6)
203 ln10 = ax.plot(tracerData['x115']['Time'],
204                 tracerData['x115']['Numerical'],
205                 '--', label='Case 10', lw=2, ms=6)
206
207 # Primary Axis
208 ax.set_xlabel('Time [s]', fontsize='x-large')
209 ax.set_ylabel('Concentration', fontsize='x-large')
210
211 plt.autoscale(enable=True, tight=True)
212 plt.grid()
213
214 # Legend
215 lns = ln0+ln1+ln2+ln3+ln4+ln5+ln6+ln7+ln8+ln9+ln10
216 labs = [l.get_label() for l in lns]
217 ax.legend(lns, labs)
218
219 # Save the image in memory in JPG format
220 figName = 'tracerDecay.jpg'
221 figName = os.path.join(selFigFolder, figName)
222 plt.savefig(figName, box_inches='tight')
223 plt.close()
224
225 del lns, ax, fig, labs
226
227 # =====
228 # Tracer decay (SemiLog Y)
229 # =====
230 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
231
232 ln0 = ax.semilogy(tracerData['x068']['Time'],
233                     tracerData['x068']['Numerical'],
234                     '_', label='Case 0', lw=2, ms=6)
235 ln1 = ax.semilogy(tracerData['x062']['Time'],
236                     tracerData['x062']['Numerical'],
237                     '--', label='Case 1', lw=2, ms=6)
238 ln2 = ax.semilogy(tracerData['x063']['Time'],
239                     tracerData['x063']['Numerical'],
240                     '_.', label='Case 2', lw=2, ms=6)
241 ln3 = ax.semilogy(tracerData['x064']['Time'],
242                     tracerData['x064']['Numerical'],

```

```

243             ':', label='Case 3', lw=2, ms=6)
244 ln4 = ax.semilogy(tracerData['x065']['Time'],
245                     tracerData['x065']['Numerical'],
246                     '-.', label='Case 4', lw=2, ms=6)
247 ln5 = ax.semilogy(tracerData['x066']['Time'],
248                     tracerData['x066']['Numerical'],
249                     '--', label='Case 5', lw=2, ms=6)
250 ln6 = ax.semilogy(tracerData['x067']['Time'],
251                     tracerData['x067']['Numerical'],
252                     '--', label='Case 6', lw=2, ms=6)
253 ln7 = ax.semilogy(tracerData['x115']['Time'],
254                     tracerData['x115']['Numerical'],
255                     '-.', label='Case 7', lw=2, ms=6)
256 ln8 = ax.semilogy(tracerData['x116']['Time'],
257                     tracerData['x116']['Numerical'],
258                     ':', label='Case 8', lw=2, ms=6)
259 ln9 = ax.semilogy(tracerData['x117']['Time'],
260                     tracerData['x117']['Numerical'],
261                     '-.', label='Case 9', lw=2, ms=6)
262 ln10 = ax.semilogy(tracerData['x115']['Time'],
263                      tracerData['x115']['Numerical'],
264                      '--', label='Case 10', lw=2, ms=6)

265
266 # Primary Axis
267 ax.set_xlabel('Time [s]', fontsize='x-large')
268 ax.set_ylabel('Concentration', fontsize='x-large')
269
270 plt.autoscale(enable=True, tight=True)
271 plt.grid()
272
273 # Legend
274 lns = ln0+ln1+ln2+ln3+ln4+ln5+ln6+ln7+ln8+ln9+ln10
275 labs = [l.get_label() for l in lns]
276 ax.legend(lns, labs)
277
278 # Save the image in memory in JPG format
279 figName = 'tracerDecaySemiLogY.jpg'
280 figName = os.path.join(selfFigFolder, figName)
281 plt.savefig(figName, box_inches='tight')
282 plt.close()
283

```

```

284 def lns, ax, fig, labs
285
286 # =====
287 # X-Velocity at XY PLANE versus (y-y0)/H Unique
288 # =====
289 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
290
291 ln0 = ax.plot(data['UMean']['x068']['p17']['y'][('y-y0)/H'],
292                 data['UMean']['x068']['p17']['y'][('u')/U,
293                 '-.', label='Case 0', lw=2, ms=6)
294 ln1 = ax.plot(data['UMean']['x062']['p17']['y'][('y')/H],
295                 data['UMean']['x062']['p17']['y'][('u')/U,
296                 '--', label='Case 1', lw=2, ms=6)
297 ln4 = ax.plot(data['UMean']['x065']['p17']['y'][('y')/H],
298                 data['UMean']['x065']['p17']['y'][('u')/U,
299                 '-.', label='Case 4', lw=2, ms=6)
300 ln7 = ax.plot(data['UMean']['x115']['p17']['y'][('y')/H],
301                 data['UMean']['x115']['p17']['y'][('u')/U,
302                 '-.', label='Case 7', lw=2, ms=6)
303 ln10 = ax.plot(data['UMean']['x115']['p17']['y'][('y')/H],
304                  data['UMean']['x115']['p17']['y'][('u')/U,
305                  ':', label='Case 10', lw=2, ms=6)
306
307 # Primary Axis
308 ax.set_xlabel(r'(y-$y_0$)/H', fontsize='x-large')
309 ax.set_ylabel('u/U', fontsize='x-large')
310
311 plt.autoscale(enable=True, tight=True)
312 plt.grid()
313
314 # Legend
315 lns = ln0+ln1+ln4+ln7+ln10
316 labs = [l.get_label() for l in lns]
317 ax.legend(lns, labs, loc=7)
318
319 # Save the image in memory in JPG format
320 figName = 'velXYPlane.jpg'
321 figName = os.path.join(selfFigFolder, figName)
322 plt.savefig(figName, box_inches='tight')
323 plt.close()
324

```

```

325 del lns, ax, fig, labs
326
327 # =====
328 # Y-Velocity at Interface versus z/H Unique
329 # =====
330 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
331
332 ln0 = ax.plot(data['UMean']['x068']['p00']['z']['v']/U,
333                 data['UMean']['x068']['p00']['z']['z/H'],
334                 '-.', label='Case 0', lw=2, ms=6)
335 ln1 = ax.plot(data['UMean']['x062']['p00']['z']['v']/U,
336                 data['UMean']['x062']['p00']['z']['z/H'],
337                 '--', label='Case 1', lw=2, ms=6)
338 #ln2 = ax.plot(data['UMean']['x063']['p00']['z']['v']/U,
339 #                 data['UMean']['x063']['p00']['z']['z/H'],
340 #                 '-.', label='Case 2', lw=2, ms=6)
341 #ln3 = ax.plot(data['UMean']['x064']['p00']['z']['v']/U,
342 #                 data['UMean']['x064']['p00']['z']['z/H'],
343 #                 ':', label='Case 3', lw=2, ms=6)
344 ln4 = ax.plot(data['UMean']['x065']['p00']['z']['v']/U,
345                 data['UMean']['x065']['p00']['z']['z/H'],
346                 '-.', label='Case 4', lw=2, ms=6)
347 #ln5 = ax.plot(data['UMean']['x066']['p00']['z']['v']/U,
348 #                 data['UMean']['x066']['p00']['z']['z/H'],
349 #                 '--', label='Case 5', lw=2, ms=6)
350 #ln6 = ax.plot(data['UMean']['x067']['p00']['z']['v']/U,
351 #                 data['UMean']['x067']['p00']['z']['z/H'],
352 #                 '--', label='Case 6', lw=2, ms=6)
353 ln7 = ax.plot(data['UMean']['x115']['p00']['z']['v']/U,
354                 data['UMean']['x115']['p00']['z']['z/H'],
355                 '-.', label='Case 7', lw=2, ms=6)
356 #ln8 = ax.plot(data['UMean']['x116']['p00']['z']['v']/U,
357 #                 data['UMean']['x116']['p00']['z']['z/H'],
358 #                 ':', label='Case 8', lw=2, ms=6)
359 #ln9 = ax.plot(data['UMean']['x117']['p00']['z']['v']/U,
360 #                 data['UMean']['x117']['p00']['z']['z/H'],
361 #                 '-.', label='Case 9', lw=2, ms=6)
362 ln10 = ax.plot(data['UMean']['x115']['p00']['z']['v']/U,
363                 data['UMean']['x115']['p00']['z']['z/H'],
364                 ':', label='Case 10', lw=2, ms=6)
365

```

```

366 # Primary Axis
367 ax.set_xlabel('v/U', fontsize='x-large')
368 ax.set_ylabel('z/H', fontsize='x-large')
369
370 plt.autoscale(enable=True, tight=True)
371 plt.grid()
372
373 # Legend
374 lns = ln0+ln1+ln4+ln7+ln10
375 labs = [l.get_label() for l in lns]
376 ax.legend(lns, labs, loc=7)
377
378 # Save the image in memory in JPG format
379 figName = 'yVelatInterfaceZAxis.jpg'
380 figName = os.path.join(selFigFolder, figName)
381 plt.savefig(figName, box_inches='tight')
382 plt.close()
383
384 del lns, ax, fig, labs
385
386 # =====
387 # Y-Velocity at Interface versus z/H 1
388 # =====
389 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
390
391 ln0 = ax.plot(data['UMean']['x068']['p00']['z']['v']/U,
392                 data['UMean']['x068']['p00']['z']['z/H'],
393                 '–', label='Case 0', lw=2, ms=6)
394 ln1 = ax.plot(data['UMean']['x062']['p00']['z']['v']/U,
395                 data['UMean']['x062']['p00']['z']['z/H'],
396                 '–', label='Case 1', lw=2, ms=6)
397 ln2 = ax.plot(data['UMean']['x063']['p00']['z']['v']/U,
398                 data['UMean']['x063']['p00']['z']['z/H'],
399                 '–', label='Case 2', lw=2, ms=6)
400 ln3 = ax.plot(data['UMean']['x064']['p00']['z']['v']/U,
401                 data['UMean']['x064']['p00']['z']['z/H'],
402                 '–', label='Case 3', lw=2, ms=6)
403 ln4 = ax.plot(data['UMean']['x065']['p00']['z']['v']/U,
404                 data['UMean']['x065']['p00']['z']['z/H'],
405                 '–', label='Case 4', lw=2, ms=6)
406 ln5 = ax.plot(data['UMean']['x066']['p00']['z']['v']/U,

```

```

407         data['UMean']['x066']['p00']['z']['z/H'],
408         '--', label='Case 5', lw=2, ms=6)
409
410 # Primary Axis
411 ax.set_xlabel('v/U', fontsize='x-large')
412 ax.set_ylabel('z/H', fontsize='x-large')
413
414 plt.autoscale(enable=True, tight=True)
415 plt.grid()
416
417 # Legend
418 lns = ln0+ln1+ln2+ln3+ln4+ln5
419 labs = [l.get_label() for l in lns]
420 ax.legend(lns, labs, loc=7)
421
422 # Title
423 plt.title('a)', loc='left', fontweight='bold')
424
425 # Save the image in memory in JPG format
426 figName = 'yVelatInterfaceZAxis1.jpg'
427 figName = os.path.join(selFigFolder, figName)
428 plt.savefig(figName, box_inches='tight')
429 plt.close()
430
431 del lns, ax, fig, labs
432
433 # =====
434 # Y-Velocity at Interface versus z/H 2
435 # =====
436 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
437
438 ln5 = ax.plot(data['UMean']['x066']['p00']['z']['v']/U,
439                 data['UMean']['x066']['p00']['z']['z/H'],
440                 '--', label='Case 5', lw=2, ms=6)
441 ln6 = ax.plot(data['UMean']['x067']['p00']['z']['v']/U,
442                 data['UMean']['x067']['p00']['z']['z/H'],
443                 '--', label='Case 6', lw=2, ms=6)
444 ln7 = ax.plot(data['UMean']['x115']['p00']['z']['v']/U,
445                 data['UMean']['x115']['p00']['z']['z/H'],
446                 '--', label='Case 7', lw=2, ms=6)
447 ln8 = ax.plot(data['UMean']['x116']['p00']['z']['v']/U,

```

```

448         data['UMean']['x116']['p00']['z']['z/H'],
449         ':', label='Case 8', lw=2, ms=6)
450 ln9 = ax.plot(data['UMean']['x117']['p00']['z']['v']/U,
451                 data['UMean']['x117']['p00']['z']['z/H'],
452                 '.', label='Case 9', lw=2, ms=6)
453 ln10 = ax.plot(data['UMean']['x115']['p00']['z']['v']/U,
454                  data['UMean']['x115']['p00']['z']['z/H'],
455                  '--', label='Case 10', lw=2, ms=6)
456
457 # Primary Axis
458 ax.set_xlabel('v/U', fontsize='x-large')
459 ax.set_ylabel('z/H', fontsize='x-large')
460
461 plt.autoscale(enable=True, tight=True)
462 plt.grid()
463
464 # Legend
465 lns = ln5+ln6+ln7+ln8+ln9+ln10
466 labs = [l.get_label() for l in lns]
467 ax.legend(lns, labs, loc=7)
468
469 # Title
470 plt.title('b', loc='left', fontweight='bold')
471
472 # Save the image in memory in JPG format
473 figName = 'yVelatInterfaceZAxis2.jpg'
474 figName = os.path.join(selfFigFolder, figName)
475 plt.savefig(figName, box_inches='tight')
476 plt.close()
477
478 del lns, ax, fig, labs
479
480 # =====
481 # Y-Velocity at Interface versus (x-x0)/L Unique
482 # =====
483 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
484
485 ln0 = ax.plot(data['UMean']['x068']['p00']['x'][ '(x-x0)/L'],
486                 data['UMean']['x068']['p00']['x'][ 'v']/U,
487                 '.', label='Case 0', lw=2, ms=6)
488 ln1 = ax.plot(data['UMean']['x062']['p00']['x'][ '(x-x0)/L'],

```

```

489         data['UMean']['x062']['p00']['x']['v']/U,
490         '--', label='Case 1', lw=2, ms=6)
491 ln2 = ax.plot(data['UMean']['x063']['p00']['x']['v']/U,
492                 data['UMean']['x063']['p00']['x']['v']/U,
493                 '.', label='Case 2', lw=2, ms=6)
494 ln3 = ax.plot(data['UMean']['x064']['p00']['x']['v']/U,
495                 data['UMean']['x064']['p00']['x']['v']/U,
496                 ':', label='Case 3', lw=2, ms=6)
497 ln4 = ax.plot(data['UMean']['x065']['p00']['x']['v']/U,
498                 data['UMean']['x065']['p00']['x']['v']/U,
499                 '.', label='Case 4', lw=2, ms=6)
500 ln5 = ax.plot(data['UMean']['x066']['p00']['x']['v']/U,
501                 data['UMean']['x066']['p00']['x']['v']/U,
502                 '--', label='Case 5', lw=2, ms=6)
503 ln6 = ax.plot(data['UMean']['x067']['p00']['x']['v']/U,
504                 data['UMean']['x067']['p00']['x']['v']/U,
505                 '--', label='Case 6', lw=2, ms=6)
506 ln7 = ax.plot(data['UMean']['x115']['p00']['x']['v']/U,
507                 data['UMean']['x115']['p00']['x']['v']/U,
508                 '.', label='Case 7', lw=2, ms=6)
509 ln8 = ax.plot(data['UMean']['x116']['p00']['x']['v']/U,
510                 data['UMean']['x116']['p00']['x']['v']/U,
511                 ':', label='Case 8', lw=2, ms=6)
512 ln9 = ax.plot(data['UMean']['x117']['p00']['x']['v']/U,
513                 data['UMean']['x117']['p00']['x']['v']/U,
514                 '.', label='Case 9', lw=2, ms=6)
515 ln10 = ax.plot(data['UMean']['x115']['p00']['x']['v']/U,
516                  data['UMean']['x115']['p00']['x']['v']/U,
517                  '--', label='Case 10', lw=2, ms=6)

518
519 # Primary Axis
520 ax.set_xlabel('(x-x0)/L', fontsize='x-large')
521 ax.set_ylabel('v/U', fontsize='x-large')
522
523 plt.autoscale(enable=True, tight=True)
524 plt.grid()
525
526 # Legend
527 lns = ln0+ln1+ln2+ln3+ln4+ln5+ln6+ln7+ln8+ln9+ln10
528 labs = [l.get_label() for l in lns]
529 ax.legend(lns, labs)

```

```

530
531 # Save the image in memory in JPG format
532 figName = 'yVelatInterfaceXAxis.jpg'
533 figName = os.path.join(selfFigFolder, figName)
534 plt.savefig(figName, box_inches='tight')
535 plt.close()

536
537 del lns, ax, fig, labs
538
539 # =====
540 # Y-Velocity at Interface versus (x-x0)/L 1
541 # =====
542 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
543
544 ln0 = ax.plot(data['UMean']['x068']['p00']['x'][ '(x-x0)/L'] ,
545                 data['UMean']['x068']['p00']['x'][ 'v']/U,
546                 ' - ', label='Case 0', lw=2, ms=6)
547 ln1 = ax.plot(data['UMean']['x062']['p00']['x'][ '(x-x0)/L'] ,
548                 data['UMean']['x062']['p00']['x'][ 'v']/U,
549                 ' -- ', label='Case 1', lw=2, ms=6)
550 ln2 = ax.plot(data['UMean']['x063']['p00']['x'][ '(x-x0)/L'] ,
551                 data['UMean']['x063']['p00']['x'][ 'v']/U,
552                 ' -.', label='Case 2', lw=2, ms=6)
553 ln3 = ax.plot(data['UMean']['x064']['p00']['x'][ '(x-x0)/L'] ,
554                 data['UMean']['x064']['p00']['x'][ 'v']/U,
555                 ' : ', label='Case 3', lw=2, ms=6)
556 ln4 = ax.plot(data['UMean']['x065']['p00']['x'][ '(x-x0)/L'] ,
557                 data['UMean']['x065']['p00']['x'][ 'v']/U,
558                 ' -.', label='Case 4', lw=2, ms=6)
559 ln5 = ax.plot(data['UMean']['x066']['p00']['x'][ '(x-x0)/L'] ,
560                 data['UMean']['x066']['p00']['x'][ 'v']/U,
561                 ' -- ', label='Case 5', lw=2, ms=6)

562
563 # Primary Axis
564 ax.set_xlabel(' (x-x0)/L', fontsize='x-large')
565 ax.set_ylabel(' v/U', fontsize='x-large')

566
567 plt.autoscale(enable=True, tight=True)
568 plt.grid()

569
570 # Legend

```

```

571 lns = ln0+ln1+ln2+ln3+ln4+ln5
572 labs = [l.get_label() for l in lns]
573 ax.legend(lns, labs)
574
575 # Title
576 plt.title('a', loc='left', fontweight='bold')
577
578 # Save the image in memory in JPG format
579 figName = 'yVelatInterfaceXAxis1.jpg'
580 figName = os.path.join(selfFigFolder, figName)
581 plt.savefig(figName, box_inches='tight')
582 plt.close()
583
584 del lns, ax, fig, labs
585
586 # =====
587 # Y-Velocity at Interface versus (x-x0)/L 2
588 # =====
589 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
590
591 ln5 = ax.plot(data['UMean']['x066']['p00']['x'][ '(x-x0)/L'],
592                 data['UMean']['x066']['p00']['x'][ 'v']/U,
593                 ' - ', label='Case 5', lw=2, ms=6)
594 ln6 = ax.plot(data['UMean']['x067']['p00']['x'][ '(x-x0)/L'],
595                 data['UMean']['x067']['p00']['x'][ 'v']/U,
596                 ' -- ', label='Case 6', lw=2, ms=6)
597 ln7 = ax.plot(data['UMean']['x115']['p00']['x'][ '(x-x0)/L'],
598                 data['UMean']['x115']['p00']['x'][ 'v']/U,
599                 ' -.', label='Case 7', lw=2, ms=6)
600 ln8 = ax.plot(data['UMean']['x116']['p00']['x'][ '(x-x0)/L'],
601                 data['UMean']['x116']['p00']['x'][ 'v']/U,
602                 ' : ', label='Case 8', lw=2, ms=6)
603 ln9 = ax.plot(data['UMean']['x117']['p00']['x'][ '(x-x0)/L'],
604                 data['UMean']['x117']['p00']['x'][ 'v']/U,
605                 ' -.', label='Case 9', lw=2, ms=6)
606 ln10 = ax.plot(data['UMean']['x115']['p00']['x'][ '(x-x0)/L'],
607                  data['UMean']['x115']['p00']['x'][ 'v']/U,
608                  ' -- ', label='Case 10', lw=2, ms=6)
609
610 # Primary Axis
611 ax.set_xlabel('(x-x0)/L', fontsize='x-large')

```

```

612 ax.set_ylabel('v/U', fontsize='x-large')
613
614 plt.autoscale(enable=True, tight=True)
615 plt.grid()
616
617 # Legend
618 lns = ln5+ln6+ln7+ln8+ln9+ln10
619 labs = [l.get_label() for l in lns]
620 ax.legend(lns, labs)
621
622 # Title
623 plt.title('b)', loc='left', fontweight='bold')
624
625 # Save the image in memory in JPG format
626 figName = 'yVelatInterfaceXAxis2.jpg'
627 figName = os.path.join(selfFigFolder, figName)
628 plt.savefig(figName, box_inches='tight')
629 plt.close()
630
631 del lns, ax, fig, labs
632
633 # =====
634 # Internal thickness versus (x-x0)/L
635 # =====
636 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
637
638 ln0 = ax.plot(thickness['x068'][['(x-x0)/L'],
639                 thickness['x068'][['internalThickness']]/W,
640                 '--', label='Case 0', lw=2, ms=6)
641 ln1 = ax.plot(thickness['x062'][['(x-x0)/L'],
642                 thickness['x062'][['internalThickness']]/W,
643                 '--', label='Case 1', lw=2, ms=6)
644 ln2 = ax.plot(thickness['x063'][['(x-x0)/L'],
645                 thickness['x063'][['internalThickness']]/W,
646                 '-.', label='Case 2', lw=2, ms=6)
647 ln3 = ax.plot(thickness['x064'][['(x-x0)/L'],
648                 thickness['x064'][['internalThickness']]/W,
649                 ':', label='Case 3', lw=2, ms=6)
650 ln4 = ax.plot(thickness['x065'][['(x-x0)/L'],
651                 thickness['x065'][['internalThickness']]/W,
652                 '-.', label='Case 4', lw=2, ms=6)

```

```

653 ln5 = ax.plot(thickness['x066'][‘(x-x0)/L’],
654             thickness['x066'][‘internalThickness’]/W,
655             ‘--’, label=‘Case 5’, lw=2, ms=6)
656 ln6 = ax.plot(thickness['x067'][‘(x-x0)/L’],
657             thickness['x067'][‘internalThickness’]/W,
658             ‘--’, label=‘Case 6’, lw=2, ms=6)
659 ln7 = ax.plot(thickness['x115'][‘(x-x0)/L’],
660             thickness['x115'][‘internalThickness’]/W,
661             ‘-.’, label=‘Case 7’, lw=2, ms=6)
662 ln8 = ax.plot(thickness['x116'][‘(x-x0)/L’],
663             thickness['x116'][‘internalThickness’]/W,
664             ‘:’, label=‘Case 8’, lw=2, ms=6)
665 ln9 = ax.plot(thickness['x117'][‘(x-x0)/L’],
666             thickness['x117'][‘internalThickness’]/W,
667             ‘-.’, label=‘Case 9’, lw=2, ms=6)
668 ln10 = ax.plot(thickness['x115'][‘(x-x0)/L’],
669             thickness['x115'][‘internalThickness’]/W,
670             ‘--’, label=‘Case 10’, lw=2, ms=6)

671
672 # Primary Axis
673 ax.set_xlabel(‘(x-x0)/L’, fontsize=‘x-large’)
674 ax.set_ylabel(r’$\delta_{in}/W$’, fontsize=‘x-large’)

675
676 plt.autoscale(enable=True, tight=True)
677 plt.grid()

678
679 # Legend
680 lns = ln0+ln1+ln2+ln3+ln4+ln5+ln6+ln7+ln8+ln9+ln10
681 labs = [l.get_label() for l in lns]
682 ax.legend(lns, labs)

683
684 # Save the image in memory in JPG format
685 figName = ‘internalThickness.jpg’
686 figName = os.path.join(selFigFolder, figName)
687 plt.savefig(figName, box_inches=‘tight’)
688 plt.close()

689
690 del lns, ax, fig, labs

691
692 # =====
693 # External thickness versus (x-x0)/L

```

```

694 # =====
695 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
696
697 ln0 = ax.plot(thickness['x068'][‘(x-x0)/L’],
698                 thickness['x068'][‘externalThickness’]/W,
699                 ‘-’, label=’Case 0’, lw=2, ms=6)
700 ln1 = ax.plot(thickness['x062'][‘(x-x0)/L’],
701                 thickness['x062'][‘externalThickness’]/W,
702                 ‘--’, label=’Case 1’, lw=2, ms=6)
703 ln2 = ax.plot(thickness['x063'][‘(x-x0)/L’],
704                 thickness['x063'][‘externalThickness’]/W,
705                 ‘-.’, label=’Case 2’, lw=2, ms=6)
706 ln3 = ax.plot(thickness['x064'][‘(x-x0)/L’],
707                 thickness['x064'][‘externalThickness’]/W,
708                 ‘:’, label=’Case 3’, lw=2, ms=6)
709 ln4 = ax.plot(thickness['x065'][‘(x-x0)/L’],
710                 thickness['x065'][‘externalThickness’]/W,
711                 ‘-.’, label=’Case 4’, lw=2, ms=6)
712 ln5 = ax.plot(thickness['x066'][‘(x-x0)/L’],
713                 thickness['x066'][‘externalThickness’]/W,
714                 ‘--’, label=’Case 5’, lw=2, ms=6)
715 ln6 = ax.plot(thickness['x067'][‘(x-x0)/L’],
716                 thickness['x067'][‘externalThickness’]/W,
717                 ‘--’, label=’Case 6’, lw=2, ms=6)
718 ln7 = ax.plot(thickness['x115'][‘(x-x0)/L’],
719                 thickness['x115'][‘externalThickness’]/W,
720                 ‘-.’, label=’Case 7’, lw=2, ms=6)
721 ln8 = ax.plot(thickness['x116'][‘(x-x0)/L’],
722                 thickness['x116'][‘externalThickness’]/W,
723                 ‘:’, label=’Case 8’, lw=2, ms=6)
724 ln9 = ax.plot(thickness['x117'][‘(x-x0)/L’],
725                 thickness['x117'][‘externalThickness’]/W,
726                 ‘-.’, label=’Case 9’, lw=2, ms=6)
727 ln10 = ax.plot(thickness['x115'][‘(x-x0)/L’],
728                  thickness['x115'][‘externalThickness’]/W,
729                  ‘--’, label=’Case 10’, lw=2, ms=6)
730
731 # Primary Axis
732 ax.set_xlabel(‘(x-x0)/L’, fontsize=’x-large’)
733 ax.set_ylabel(r’$\delta_{out}/W$, fontsize=’x-large’)

```

```

735 plt.autoscale(enable=True, tight=True)
736 plt.grid()
737
738 # Legend
739 lns = ln0+ln1+ln2+ln3+ln4+ln5+ln6+ln7+ln8+ln9+ln10
740 labs = [l.get_label() for l in lns]
741 ax.legend(lns, labs)
742
743 # Save the image in memory in JPG format
744 figName = 'externalThickness.jpg'
745 figName = os.path.join(selFigFolder, figName)
746 plt.savefig(figName, box_inches='tight')
747 plt.close()
748
749 del lns, ax, fig, labs
750
751 # =====
752 # Total thickness versus (x-x0)/L
753 # =====
754 fig, ax = plt.subplots(figsize=(9,6), dpi=500)
755
756 ln0 = ax.plot(thickness['x068'][ '(x-x0)/L' ],
757                 thickness['x068'][ 'totalThickness' ]/W,
758                 '-.', label='Case 0', lw=2, ms=6)
759 ln1 = ax.plot(thickness['x062'][ '(x-x0)/L' ],
760                 thickness['x062'][ 'totalThickness' ]/W,
761                 '--', label='Case 1', lw=2, ms=6)
762 ln2 = ax.plot(thickness['x063'][ '(x-x0)/L' ],
763                 thickness['x063'][ 'totalThickness' ]/W,
764                 '-.', label='Case 2', lw=2, ms=6)
765 ln3 = ax.plot(thickness['x064'][ '(x-x0)/L' ],
766                 thickness['x064'][ 'totalThickness' ]/W,
767                 ':', label='Case 3', lw=2, ms=6)
768 ln4 = ax.plot(thickness['x065'][ '(x-x0)/L' ],
769                 thickness['x065'][ 'totalThickness' ]/W,
770                 '-.', label='Case 4', lw=2, ms=6)
771 ln5 = ax.plot(thickness['x066'][ '(x-x0)/L' ],
772                 thickness['x066'][ 'totalThickness' ]/W,
773                 '--', label='Case 5', lw=2, ms=6)
774 ln6 = ax.plot(thickness['x067'][ '(x-x0)/L' ],
775                 thickness['x067'][ 'totalThickness' ]/W,

```

```

776             '--', label='Case 6', lw=2, ms=6)
777 ln7 = ax.plot(thickness['x115'][‘(x-x0)/L’],
778                 thickness['x115'][‘totalThickness’]/W,
779                 ‘-.’, label='Case 7', lw=2, ms=6)
780 ln8 = ax.plot(thickness['x116'][‘(x-x0)/L’],
781                 thickness['x116'][‘totalThickness’]/W,
782                 ‘:’, label='Case 8', lw=2, ms=6)
783 ln9 = ax.plot(thickness['x117'][‘(x-x0)/L’],
784                 thickness['x117'][‘totalThickness’]/W,
785                 ‘-.’, label='Case 9', lw=2, ms=6)
786 ln10 = ax.plot(thickness['x115'][‘(x-x0)/L’],
787                  thickness['x115'][‘totalThickness’]/W,
788                  '--', label='Case 10', lw=2, ms=6)

789
790 # Primary Axis
791 ax.set_xlabel('$(x-x_0)/L$', fontsize='x-large')
792 ax.set_ylabel(r'$\delta/W$', fontsize='x-large')
793
794 plt.autoscale(enable=True, tight=True)
795 plt.grid()

796
797 # Legend
798 lns = ln0+ln1+ln2+ln3+ln4+ln5+ln6+ln7+ln8+ln9+ln10
799 labs = [l.get_label() for l in lns]
800 ax.legend(lns, labs)

801
802 # Save the image in memory in JPG format
803 figName = 'totalThickness.jpg'
804 figName = os.path.join(selFigFolder, figName)
805 plt.savefig(figName, box_inches='tight')
806 plt.close()

807
808 del lns, ax, fig, labs

```

---

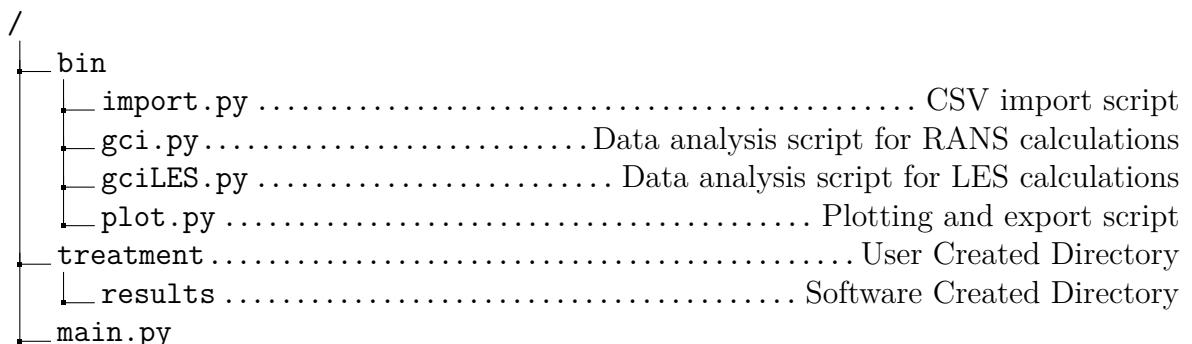
## **Appendix C**

### **Grid Convergence Index (GCI) Python Script**

In this appendix the script used to determine the grid convergence index is presented. This code is an automated script based on Celik et al. (2008) and Dutta e Xing (2018).

## C.1 File Structure

The file structure of the script is shown bellow:



The requirements of the script are:

- Python 3.x
- Scipy
- Numpy
- Pandas
- Matplotlib

## C.2 main.py

The user must provide a folder named treatment where three different *csv* files must be placed. The execution of the code depends only on the main.py file that must be run in a python terminal:

---

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # main.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
```

```

8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24
25 """
26 Main module
27
28 This script analyses numerical and modeling errors in LES simulations and
29 Grid convergence analysis on RANS simulations.
30 The analysis steps are performed by the modules in the bin folder
31 """
32
33 import sys
34 import os
35 import shutil
36 import time
37 start_time = time.time()
38
39 def cls():
40     """
41     Clears the prompt
42     """
43     os.system('cls' if os.name=='nt' else 'clear')
44
45 # Check for necessary directories
46 if not os.path.exists('treatment'):
47     os.makedirs('treatment')
48     print("The directory treatment/ was created, please populate with the "

```

```

49         "desired csv files to be analysed.")
50     sys.exit('The directory treatment/ did not exist.')
51 elif not os.listdir('treatment'):
52     sys.exit('The directory treatment/ is empty.')
53
54 # Clear the previous results directories
55 if os.path.exists('treatment/results'):
56     shutil.rmtree('treatment/results')
57 os.makedirs('treatment/results')
58
59 analysisType = input("Type of Analysis\n[1] RANS\n[2] LES\nChosen Option: ")
60
61 if analysisType == 1:
62     # Import CSV
63     exec(open("bin/import.py").read());
64     # Grid Convergence Analysis (RANS)
65     exec(open("bin/gci.py").read());
66 else:
67     # Grid Convergence Analysis (LES)
68     exec(open("bin/gciLES.py").read());

```

---

### C.3 import.py

The import process occurs in bin/import.py file:

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # Libraries
5 import os
6 import re
7 import pandas as pd
8 from detect_delimiter import detect
9
10 def cls():
11     """
12     Clears the prompt
13     """
14     os.system('cls' if os.name=='nt' else 'clear')

```

```

15
16 # Input delimiter and file names
17 coarserFile = input("Name of coarser mesh file: ")
18 mediumFile = input("Name of medium mesh file: ")
19 finerFile = input("Name of finer mesh file: ")

20
21 coarserFile = "treatment/" + coarserFile
22 mediumFile = "treatment/" + mediumFile
23 finerFile = "treatment/" + finerFile

24
25 with open(coarserFile) as f:
26     for line in f:
27         if re.match(r"^\d+.*$",line):
28             delim = detect(line)
29             break
30     if delim is None:
31         delim = input("""Type of delimiter\n[1] (\\"\\t')\n[2] ('')\n[3] (';')
32                               [4] (',')\n[5] Custom delimiter\nChosen option: """)
33     delim = int(delim)
34     if delim == 1:
35         delim = '\\\\t'
36     elif delim == 2:
37         delim = ' '
38     elif delim == 3:
39         delim = ';'
40     elif delim == 4:
41         delim = ','
42     elif delim == 5:
43         delim = input("Enter custom delimiter: ")

44
45 cls()

46
47 print("Python columns start on zero, please pay attention to this detail.\n")
48 axis = int(input("Axis column number: "))
49 var = int(input("Variable column number: "))

50
51 headerlines = int(input("Number of header lines: "))

52
53 # Import generated data
54 coarser = pd.read_csv(coarserFile,delimiter=delim, skiprows=headerlines,
55                         usecols=[axis,var], header=0,

```

```

56             names=["Axis","Variable_coarser"])
57 medium = pd.read_csv(mediumFile,delimiter=delim, skiprows=headerlines,
58                      usecols=[axis,var], header=0,
59                      names=["Axis","Variable_medium"])
60 finer = pd.read_csv(finerFile,delimiter=delim, skiprows=headerlines,
61                      usecols=[axis,var], header=0,
62                      names=["Axis","Variable_finer"])
63
64 # Reindexing using axis
65 coarser = coarser.set_index('Axis')
66 medium = medium.set_index('Axis')
67 finer = finer.set_index('Axis')
68
69 # Sorting imported data
70 coarser = coarser.sort_values('Axis')
71 medium = medium.sort_values('Axis')
72 finer = finer.sort_values('Axis')
73
74 cls()

```

---

## C.4 gci.py

The processing occurs in bin/gci.py file for RANS calculations:

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5 import pandas as pd
6 import numpy as np
7
8 def cls():
9     """
10     Clears the prompt
11     """
12     os.system('cls' if os.name=='nt' else 'clear')
13
14 cElements = int(input("Number of elements of the coarser mesh: "))
15 mElements = int(input("Number of elements of the medium mesh: "))

```

```

16 fElements = int(input("Number of elements of the finer mesh: "))
17
18 analysisType = input("Type of Analysis\n[1] 2D\n[2] 3D\nChosen Option: ")
19 volume = float(input("Total cell volume [m3]: "))
20
21 if analysisType == '1':
22     h1 = (volume/fElements)**(0.5)
23     h2 = (volume/mElements)**(0.5)
24     h3 = (volume/cElements)**(0.5)
25
26 elif analysisType == '2':
27     h1 = (volume/fElements)**(1/3)
28     h2 = (volume/mElements)**(1/3)
29     h3 = (volume/cElements)**(1/3)
30
31 else:
32     cls()
33     print("Deleting all data...")
34     print("Computer shutting down...")
35
36 # Refinement rate
37
38 r21 = h2/h1
39 r32 = h3/h2
40
41 # Variable absolute error
42 desiredVar = pd.concat([finer, medium, coarser], axis=1)
43 desiredVar = desiredVar.interpolate('index').reindex(medium.index)
44 e21 = desiredVar.Variable_medium - desiredVar.Variable_finer
45 e32 = desiredVar.Variable_coarser - desiredVar.Variable_medium
46 desiredVar['e21'] = e21
47 desiredVar['e32'] = e32
48
49 # Sign
50 sign = np.sign(desiredVar['e32']/desiredVar['e21'])
51 desiredVar['Sign'] = sign.astype(float)
52
53 # Order Error
54 initial = np.repeat(2.0, len(desiredVar.index))
55
56 def apparentOrder(order, df):

```

```

57     order = np.abs(order)
58     q = np.log(((r21**order)-desiredVar.Sign)/((r32**order)-desiredVar.Sign))
59     ap =
60         np.abs(np.log(np.abs(desiredVar['e32'])/desiredVar['e21'])+q))/np.log(r21)
61     error = np.abs(order - ap)
62     error = np.array(error.values.tolist()) #converts to array
63     return np.mean(error)

64 res = optimize.minimize(aparentOrder, args=(desiredVar),
65                         x0=initial, method = 'Nelder-Mead', tol=0.01,
66                         options={'maxiter':1000})

67
68 order = res.x
69 q = np.log((r21**order-desiredVar.Sign)/(r32**order-desiredVar.Sign))
70 ap = np.abs(np.log(np.abs(desiredVar['e32'])/desiredVar['e21'])+q))/np.log(r21)
71 orderError = order - ap

72
73 desiredVar['Aparent Order'] = ap
74 desiredVar['Optimized Order'] = order
75 desiredVar['Order Error'] = orderError

76
77 # Extrapolated values
78 ext21 =
79     ((r21**ap)*desiredVar.Variable_finer-desiredVar.Variable_medium)/((r21**ap)-1)
80 ext32 =
81     ((r32**ap)*desiredVar.Variable_medium-desiredVar.Variable_coarser)/((r32**ap)-1)

82 desiredVar['Extrapolated Value (Finer, Medium)'] = ext21
83 desiredVar['Extrapolated Value (Medium, Coarser)'] = ext32

84 # Calculate and report the error estimatives
85 apxRelErr =
86     np.abs((desiredVar.Variable_finer-desiredVar.Variable_medium)/desiredVar.Variable_fine
87 extRelErr = np.abs((ext21-desiredVar.Variable_finer)/ext21)
88 gci = (1.25*apxRelErr)/((r21**ap)-1)

89 desiredVar['Aproximated Relative Error'] = apxRelErr
90 desiredVar['Extrapolated Relative Error'] = extRelErr
91 desiredVar['Grid Convergence Index'] = gci

92
93 # Export generated table

```

```
94 desiredVar.to_excel("treatment/results/gci.xlsx")
```

---

## C.5 gciLES.py

The processing occurs in bin/gciLES.py file for LES calculations:

---

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 #
4 # gciLES.py
5 #
6 # Copyright 2020 Luiz Oliveira
7 #
8 # This program is free software; you can redistribute it and/or modify
9 # it under the terms of the GNU General Public License as published by
10 # the Free Software Foundation; either version 2 of the License, or
11 # (at your option) any later version.
12 #
13 # This program is distributed in the hope that it will be useful,
14 # but WITHOUT ANY WARRANTY; without even the implied warranty of
15 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 # GNU General Public License for more details.
17 #
18 # You should have received a copy of the GNU General Public License
19 # along with this program; if not, write to the Free Software
20 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
21 # MA 02110-1301, USA.
22 #
23 #
24
25 """
26 Main module
27
28 This script analyses numerical and modeling errors in LES simulations
29 """
30
31 import os
32 import re
33 import openpyxl
```

```

34 import pandas as pd
35 import matplotlib.pyplot as plt
36 from detect_delimiter import detect
37 from scipy.optimize import fsolve
38
39 def cls():
40     """
41     Clears the prompt
42     """
43     os.system('cls' if os.name=='nt' else 'clear')
44
45 def caseInfo(ncases):
46     """
47     Reads the case information for an n number of simulations
48     """
49     d = {'Elements' : [], 'DeltaT' : [], 'Volume' : []}
50     for ii in range(ncases):
51         elmt = int(input("Number of elements of the mesh {0}: ".format(ii)))
52         dt = float(input("Timestep size of the mesh {0}: ".format(ii)))
53         d['Elements'].append(elmt)
54         d['DeltaT'].append(dt)
55         d['Volume'].append(float(input("Total cell volume [m3]: ")))
56     df = pd.DataFrame(dict([ (k,pd.Series(v)) for k,v in d.items() ]))
57     df.index.names = ['Mesh']
58     df.to_csv('treatment/caseInformation.csv', index = False)
59     return df
60
61 def checkDelimiter(filename, directory):
62     """
63     Checks the delimiter of a file or takes the input from the user
64     """
65     with open(os.path.join(directory,filename)) as f:
66         for line in f:
67             if re.match(r"^\d+.*$",line):
68                 delim = detect(line)
69                 break
70     if delim is None:
71         delim = input("""Type of delimiter\n[1] ('\\t')\n[2] (' ')\\n[3] (';')\n[4] (',')\\n [5] Custom delimiter\\nChosen option: """)
72
73     delim = int(delim)
74     if delim == 1:

```

```

75         delim = '\\t'
76     elif delim == 2:
77         delim = ' '
78     elif delim == 3:
79         delim = ';'
80     elif delim == 4:
81         delim = ','
82     elif delim == 5:
83         delim = input("Enter custom delimiter: ")
84     cls()
85     return delim
86
87 def caseImport(ncases):
88 """
89 Imports the data from an n number of simulations
90 """
91 directory = 'treatment'
92 # Get all files.
93 list = os.listdir(directory)
94 filetpl = []
95 for file in list:
96     # Use join to get full file path.
97     location = os.path.join(directory, file)
98     # Get size and add to list of tuples.
99     size = os.path.getsize(location)
100    filetpl.append((size, file))
101 # Sort list of tuples by the first element, size.
102 filetpl.sort(key=lambda s: s[0])
103 filetpl.reverse()
104 df = pd.DataFrame(data=filetpl, columns=["Size", "Filename"])
105 df.drop(df.tail(len(list)-ncases).index, inplace = True)
106 df.index.names = ['Mesh']
107 print("Assuming this file order:")
108 print(df.to_string())
109 order = int(input("Is this correct?\n[1] Yes\n[2] No\nChoice: "))
110 if order == 2:
111     lst = []
112     print("Write the mesh number succeeded by the file name:\n")
113     for ii in range(ncases):
114         file = [int(input()), input()]
115         lst.append(file)

```

```

116     df = pd.DataFrame(data=lst, columns=["Size","Filename"])
117     df.index.names = ['Mesh']
118     cls()
119     print("Files to be imported:\n")
120     print(df.to_string())
121     importedFiles = dict()
122     delim = checkDelimiter(df.Filename[0],directory)
123     for ii in range(ncases):
124         temp = pd.read_csv(os.path.join(directory,df.Filename[ii]),
125                             delimiter=delim)
126         importedFiles['Mesh '+str(ii)] = temp
127     return importedFiles
128
129 def refinementRate(df):
130     """
131     Defines the refinement rate between the meshes
132     """
133     r = list()
134     for n in range(testVersion):
135         if n == testVersion - 1:
136             refRate = 1
137         else:
138             refRate = df.Elements[n+1]/df.Elements[n]
139             r.append(refRate)
140     df['r'] = r
141     return df
142
143 # Import the case structure data (mesh and timestep)
144 testVersion = int(input("""Which test should be performed?
145 [1] Short Version (3 cases)
146 [2] Long Version (5 cases)
147 Choice: """))
148 if testVersion == 1:
149     testVersion = 3
150 else:
151     testVersion = 5
152
153 infoFile = 'treatment/caseInformation.csv'
154 if os.path.exists(infoFile):
155     infoDf = pd.read_csv(infoFile)
156 else:

```

```

157     print ("Please state the meshes from the finer to the coarser")
158     infoDf = caseInfo(testVersion)
159
160 # Import simulation data
161 nVar = int(input("""[1] Single data point
162 [2] Multiple data point (line)
163 Choice: """))
164 cls()
165 var = input("Write the name of the desired variable: ")
166 axis = input("Write the name of the desired plot axis: ")
167 if nVar == 1:
168     cls()
169     print("Please insert the point value for the meshes")
170     simDf = dict()
171     for ii in range(testVersion):
172         jj = str(ii)
173         lst = [float(input("Mesh "+jj+" value: "))]
174         simDf['Mesh '+jj] = pd.DataFrame(data=lst,columns=[var])
175         del ii,jj,lst
176     elif nVar == 2:
177         simDf = caseImport(testVersion)
178     del nVar
179
180 # Starts evaluating the GCI
181 infoDf.eval('h = (@infoDf.Volume[0]/Elements)**(1/3)', inplace=True)
182 infoDf.eval('hstar = (h*DeltaT)**(1/2)', inplace=True)
183 infoDf = refinementRate(infoDf)
184 delta = max(infoDf.h)
185 r = infoDf.r.mean()
186 hstar = infoDf.hstar.mean()
187
188 ## Simulated data
189 s1 = simDf['Mesh 0'][var]
190 s2 = simDf['Mesh 1'][var]
191 s3 = simDf['Mesh 2'][var]
192 if testVersion == 5:
193     s4 = simDf['Mesh 3'][var]
194     s5 = simDf['Mesh 4'][var]
195
196 if testVersion == 3:
197     # Simplified method

```

```

198
199     pn = 1.7
200     pm = 1.5
201     cm = (r**1.7)*(s1-s2)-(s2-s3))/((r**1.7)-r**1.5)-r**3.2+r**3))\
202         *delta**1.5)
203     sc = ((r**1.7)*s1-s2)*(r**3.2)-r**3))-(r**1.7)*s2-s3)*(r**1.7) \
204         -r**1.5))/((r**1.7)-1)*((r**3.2)-r**3))-(r**1.7)-r**1.5)))
205     cn = (s1-sc-cm*delta**1.5)/(hstar**1.7))
206
207     Enum = dict()
208     Enum[0] = cn*(hstar**1.7)
209     Enum[1] = cn*(r**1.7)*(hstar**1.7)
210     Enum[2] = cn*(r**3.4)*(hstar**1.7)
211
212     Emod = dict()
213     Emod[0] = cm*(delta**1.5)
214     Emod[1] = cm*(r**1.5)*(delta**1.5)
215     Emod[2] = cm*(r**3)*(delta**1.5)
216
217     jj=0
218     for ii in simDf:
219         simDf[ii]['Sc'] = sc
220         simDf[ii]['Numerical Error'] = Enum[jj]
221         simDf[ii]['Modelling Error'] = Emod[jj]
222         simDf[ii]['Total Error'] = Enum[jj] + Emod[jj]
223         jj+=1
224     del ii,jj
225
226 elif testVersion == 5:
227     # Full method
228
229     def fullMethod(vars):
230         # =====
231         # Sets the nonlinear system of 5 equations
232         # =====
233         sc, cn, cm, pn, pm = vars
234         eq1 = cn*hstar**pn + cm*delta**pm
235         eq2 = cn*(r*hstar)**pn + cm*(r*delta)**pm
236         eq3 = cn*((r**2)*hstar)**pn + cm*((r**2)*delta)**pm
237         eq4 = cn*((r**3)*hstar)**pn + cm*((r**3)*delta)**pm
238         eq5 = cn*((r**4)*hstar)**pn + cm*((r**4)*delta)**pm

```

```

239         return [eq1, eq2, eq3, eq4, eq5]
240
241     sc, cn, cm, pn, pm = fsolve(fullMethod, (0.007, 1, 1, 1.7, 1.5))
242     Enum = dict()
243     for ii in range(testVersion):
244         if ii == 0:
245             val = cn*hstar**pn
246         else:
247             val = cn*((r**ii)*hstar)**pn
248         Enum[ii]=val
249
250     Emod = dict()
251     for ii in range(testVersion):
252         if ii == 0:
253             val = cm*delta**pm
254         else:
255             val = cm*((r**ii)*delta)**pm
256         Emod[ii]=val
257
258     jj=0
259     for ii in simDf:
260         simDf[ii]['Sc'] = sc
261         simDf[ii]['Numerical Error'] = Enum[jj]
262         simDf[ii]['Modelling Error'] = Emod[jj]
263         simDf[ii]['Total Error'] = Enum[jj] + Emod[jj]
264         jj+=1
265     del ii, jj, val, var
266
267 # Export Results to Excel
268 d = {'Order of Accuracy for the Numerical Error (Pn)': pn,
269       'Order of Accuracy for the Modelled Error (Pm)': pm,
270       'Mean Constant for Numerical Errors (Cn)': cn.mean(),
271       'Mean Constant for Modelled Errors (Cm)': cm.mean(),
272       'Delta': delta,
273       'Hstar': hstar,
274       'Mean Refinement Rate': r
275   }
276 idx = [0]
277
278 summary = pd.DataFrame(data=d, index=idx)
279 xlsxFile = 'treatment/results/dataSummary.xlsx'

```

```

280 if not os.path.isfile(xlsxFile):
281     wb = openpyxl.Workbook()
282     wb.save(xlsxFile)
283
284 with pd.ExcelWriter(xlsxFile, engine="openpyxl", mode='a') as writer:
285     summary.to_excel(writer, sheet_name='Summary', index=False)
286     for df_name, df in simDf.items():
287         df.to_excel(writer, sheet_name=df_name, index=False)
288
289 del d, idx, xlsxFile
290
291 # Plot with error bars
292 fig, ax = plt.subplots(figsize=(9,6), dpi=300)
293 ax.plot(simDf['Mesh 0'][axis], simDf['Mesh 0'][var],
294          label= 'Mesh 0', aa=True)
295 ax.plot(simDf['Mesh 1'][axis], simDf['Mesh 1'][var],
296          label= 'Mesh 1', aa=True)
297 ax.plot(simDf['Mesh 2'][axis], simDf['Mesh 2'][var],
298          label= 'Mesh 2', aa=True)
299 if testVersion == 5:
300     ax.plot(simDf['Mesh 3'][axis], simDf['Mesh 3'][var],
301             label= 'Mesh 3 - Coarser', aa=True)
302     ax.plot(simDf['Mesh 4'][axis], simDf['Mesh 4'][var],
303             label= 'Mesh 4 - Coarser', aa=True)
304
305 ax.legend(loc='best', fontsize='x-large')
306
307 plt.grid()
308 plt.autoscale(enable=True, tight=True)
309 plt.xlabel(axis, fontsize='x-large')
310 plt.ylabel(var, fontsize='x-large')
311 plt.savefig('treatment/results/allMeshes.png', bbox_inches='tight')
312
313 fig, ax = plt.subplots(figsize=(9,6), dpi=300)
314 l, caps, c = plt.errorbar(simDf['Mesh 1'][axis], simDf['Mesh 1'][var],
315                            simDf['Mesh 1']['Total Error'],
316                            elinewidth = 1, capsize = 5, capthick = 1, marker = 'o',
317                            # errorevery = 5,
318                            uplims = True, lolims = True,
319                            lw=1.5, aa = True)
320

```

```

321 for cap in caps:
322     cap.set_marker("_")
323
324 plt.grid()
325 plt.autoscale(enable=True, tight=True)
326 plt.xlabel(axis, fontsize='x-large')
327 plt.ylabel(var, fontsize='x-large')
328 plt.savefig('treatment/results/Mesh1.png', bbox_inches='tight')

```

---

## C.6 plot.py

And finally the plotting and the spreadsheet containing the results are output by bin/plot.py file:

---

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import matplotlib.pyplot as plt
5
6  fig, ax = plt.subplots(figsize=(9,6), dpi=300)
7  ax.plot(desiredVar.index, desiredVar.Variable_coarser,
8          label= 'Coarser', aa=True)
9  ax.plot(desiredVar.index, desiredVar.Variable_medium,
10         label= 'Medium', aa=True)
11 ax.plot(desiredVar.index, desiredVar.Variable_finer,
12         label= 'Finer', aa=True)
13
14 ax.legend(loc='best', fontsize='x-large')
15
16 plt.grid()
17 plt.autoscale(enable=True, tight=True)
18 plt.savefig('treatment/results/allMeshes.png')
19
20 fig, ax = plt.subplots(figsize=(9,6), dpi=300)
21 ax.errorbar(desiredVar.index, desiredVar.Variable_medium,
22             gci*desiredVar.Variable_medium,
23             errorevery = 5, elinewidth = 1,
24             uplims = True, lolims = True,
25             lw=1.5, aa = True)

```

```
26
27 plt.grid()
28 plt.autoscale(enable=True, tight=True)
29 plt.savefig('treatment/results/mediumWithErrorbars.png')
30
31 fig, ax = plt.subplots(figsize=(9,6), dpi=300)
32 ax.errorbar(desiredVar.index, desiredVar.Variable_finer,
33             gci*desiredVar.Variable_finer,
34             errorevery = 5, elinewidth = 1,
35             uplims = True, lolims = True,
36             lw=1.5, aa = True)
37
38 plt.grid()
39 plt.autoscale(enable=True, tight=True)
40 plt.savefig('treatment/results/finerWithErrorbars.png')
```

---

## Appendix D

# OpenFOAM Configuration of The Effects of Vegetation Density Upon Flow and Mass Transport in Lateral Cavities model

In this appendix, the code in the chapter: The Effects of Vegetation Density Upon Flow and Mass Transport in Lateral Cavities. A copy of this configuration will also be available at the Github repository linked in the conclusion.

## D.1 File Structure

The file structure of the script is shown bellow:

```

/
  0.orig ..... Initial Boundary Conditions Folder
    nut ..... Boundary conditions of turbulent viscosity
    p ..... Boundary conditions of pressure
    tracer ..... Boundary conditions of the inert scalar
    0 ..... Boundary conditions of velocity
  constant ..... Mesh and General information about the simulation
    fvOptions ..... Configuration of the porous media
    g ..... Gravity
    transportProperties ..... Fluid characteristics
    turbulenceProperties ..... Turbulence settings
      boundaryData ..... Pre-calculated fields for the inlet
        inlet
          points
            0
              k
              L
              nut
              nuTilda
              omega
              p
              R
              U
  system ..... Main configuration folder
    blockMeshDict ..... Mesh configuration
    controlDict ..... Simulation Control
    decomposeParDict ..... Configuration of the parallelisation of the grid
    fvSchemes ..... Configuration of the used numerical schemes
    fvSolution ..... Configuration of the solver
    setFieldsDict ..... Set of the initial tracer fields
    topoSetDict ..... Mesh manipulation
    totalTKE
  allClear
  mesh ..... Bash script to aid the creation of the mesh
  ramCache ..... Bash script to clear the memory cache
  reconstructParParallel ..... Union of the parallel cases into a single directory
  x###.foam ..... Header file for the visualisation software (Paraview)

```

## D.2 0.orig/nut

```
1  /*-----* C++ -----*/
2  | ====== |
3  | \\\ / F ield      | OpenFOAM: The Open Source CFD Toolbox |
4  | \\\ / O peration   | Version: v1912 |
5  | \\\ / A nd        | Website: www.openfoam.com |
6  | \\\/ M anipulation |
7  \*-----*/
8  FoamFile
9  {
10    version    2.0;
11    format     ascii;
12    class      volScalarField;
13    location   "0";
14    object     nut;
15 }
16 // * * * * *
17
18 dimensions [0 2 -1 0 0 0 0];
19
20 internalField uniform 0;
21
22 boundaryField
23 {
24   inlet
25   {
26     type          timeVaryingMappedFixedValue;
27     setAverage   false;
28     perturb      0;
29   }
30   outlet
31   {
32     type          calculated;
33     value         uniform 0;
34   }
35   bottom
36   {
37     type          nutUSpaldingWallFunction;
38     value         uniform 0;
39     maxIter      100;
```

```

40         tolerance      1e-07;
41     }
42     lateralWall
43     {
44         type          nutUSpaldingWallFunction;
45         value         uniform 0;
46         maxIter      100;
47         tolerance    1e-07;
48     }
49     freeSurface
50     {
51         type          zeroGradient;
52     }
53     farField
54     {
55         type          zeroGradient;
56     }
57 }
58
59
60 // ****

```

---

### D.3 0.orig/p

```

1  /*-----* C++ -----*/
2  | ====== |
3  | \\ / Field | OpenFOAM: The Open Source CFD Toolbox |
4  | \\ / Operation | Version: v1912 |
5  | \\ / And | Website: www.openfoam.com |
6  | \\ / Manipulation |
7  \*-----*/
8 FoamFile
9 {
10    version   2.0;
11    format    ascii;
12    class     volScalarField;
13    location  "0";
14    object    p;
15 }

```



## D.4 0.orig/tracer

```
1  /*-----* C++ -----*/
2  | ====== |
3  | \ \ / F ield      | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O peration   | Version: v1912 |
5  | \ \ / A nd        | Website: www.openfoam.com |
6  | \ \ / M anipulation |
7  \*-----*/
8  FoamFile
9  {
10    version    2.0;
11    format     ascii;
12    class      volScalarField;
13    location   "0";
14    object     tracer;
15 }
// * * * * *
16
17
18 dimensions [0 0 0 0 0 0];
19
20 internalField uniform 0;
21
22 boundaryField
23 {
24   inlet
25   {
26     type      zeroGradient;
27   }
28   outlet
29   {
30     type      zeroGradient;
31   }
32   bottom
33   {
34     type      zeroGradient;
35   }
36   lateralWall
37   {
38     type      zeroGradient;
39 }
```

```

40     farField
41     {
42         type          zeroGradient;
43     }
44     freeSurface
45     {
46         type          zeroGradient;
47     }
48 }
49
50
51 // ****

```

---

## D.5 0.orig/U

```

1  /*-----*-- C++ --*-----*/
2  | ====== |
3  | \\\    / F ield      | OpenFOAM: The Open Source CFD Toolbox |
4  | \\\    / O peration   | Version: v1912 |
5  | \\\  / A nd          | Website: www.openfoam.com |
6  |   \\\/ M anipulation |
7  \*-----*/
8 FoamFile
9 {
10    version    2.0;
11    format     ascii;
12    class      volVectorField;
13    location   "0";
14    object     U;
15 }
16 // * * * * *
17
18 dimensions      [0 1 -1 0 0 0 0];
19
20 internalField uniform (0.101 0 0);
21
22 boundaryField
23 {
24     inlet

```

```

25     {
26         type          turbulentDFSEMinlet;
27         delta         0.021;
28         interpolateU true;
29         interpolateL true;
30         interpolateR true;
31         value         uniform (0.101 0 0);
32     }
33     outlet
34     {
35         type          zeroGradient;
36     }
37     bottom
38     {
39         type          noSlip;
40     }
41     lateralWall
42     {
43         type          noSlip;
44     }
45     freeSurface
46     {
47         type          slip;
48     }
49     farField
50     {
51         type          slip;
52     }
53 }
54
55
56 // ****

```

---

## D.6 constant/fvOptions

```

1  /*-----* C++ -----*/
2  | ====== | |
3  | \\ / Field | OpenFOAM: The Open Source CFD Toolbox | |
4  | \\ / Operation | Version: v1912 | |

```

```

5 | \\ / A nd | Website: www.openfoam.com |
6 | \\\ Manipulation |
7 /*-----*/
8 FoamFile
9 {
10     version    2.0;
11     format      ascii;
12     class       dictionary;
13     location    "constant";
14     object      fvOptions;
15 }
16 // * * * * *
17
18 embayment
19 {
20     type          explicitPorositySource;
21     active        true;
22     selectionMode cellZone;
23     cellZone     embayment;
24
25     explicitPorositySourceCoeffs
26     {
27         selectionMode cellZone;
28         cellZone     embayment;
29
30         type          DarcyForchheimer;
31
32         mu  mu;
33         d   (116.62 116.62 4.51E-04); //Original values d (116.62 116.62
34             4.51E-04);
35         f   (3.09 3.09 6.08E-03);    //Original values f (3.09 3.09 6.08E-03);
36
37         coordinateSystem
38         {
39             origin (0.25 0.30 0);
40             e1    (1 0 0);
41             e2    (0 1 0);
42         }
43     }
44

```

```
45
46 //*****
```

---

## D.7 constant/g

```
1 /*-----* C++ -----*/
2 | ====== |
3 | \\\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
4 | \\\ / O peration | Version: v1912 |
5 | \\\ / A nd | Website: www.openfoam.com |
6 | \\\/ M anipulation |
7 */
8 FoamFile
9 {
10     version    2.0;
11     format      ascii;
12     class       uniformDimensionedVectorField;
13     location    "constant";
14     object      g;
15 }
16 // * * * * *
17
18 dimensions [0 1 -2 0 0 0 0];
19 value (0 0 -9.81);
20
21
22 // *****
```

---

## D.8 constant/transportProperties

```
1 /*-----* C++ -----*/
2 | ====== |
3 | \\\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
4 | \\\ / O peration | Version: v1912 |
5 | \\\ / A nd | Website: www.openfoam.com |
6 | \\\/ M anipulation |
7 */
```

```

8   FoamFile
9   {
10      version 2.0;
11      format ascii;
12      class dictionary;
13      location constant;
14      object transportProperties;
15  }
16 // * * * * *
17
18      transportModel Newtonian;
19      nu          [ 0 2 -1 0 0 0 0 ] 1E-6;
20      mu          [ 1 -1 -1 0 0 0 0 ] 1E-03
21      rho         [ 1 -3 0 0 0 0 0 ] 1000;
22
23 // ****

```

---

## D.9 constant/turbulenceProperties

```

1  /*-----*-- C++ --*-----*/
2  | ====== |
3  | \\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
4  | \\ / O peration | Version: v1912 |
5  | \\ / A nd | Website: www.openfoam.com |
6  | \\/ M anipulation |
7  \*-----*/
8   FoamFile
9   {
10      version    2.0;
11      format     ascii;
12      class      dictionary;
13      location   "constant";
14      object     turbulenceProperties;
15  }
16 // * * * * *
17
18 simulationType LES;
19
20 LES

```

```

21  {
22      turbulence      on;
23      LESModel        WALE;
24      printCoeffs    on;
25
26      delta          cubeRootVol; //since the WALE model does not require damping
27          close to the wall
28
29  // ****

```

---

## D.10 system/blockMeshDict

```

1  /*-----* C++ -----*/
2  | ====== | |
3  | \\\ / Field | OpenFOAM: The Open Source CFD Toolbox | |
4  | \\\ / Operation | Version: v1912 | |
5  | \\\ / And | Website: www.openfoam.com | |
6  | \\\/ Manipulation | |
7  \*-----*/
8  FoamFile
9  {
10     version 2.0;
11     format ascii;
12     class dictionary;
13     location system;
14     object blockMeshDict;
15 }
16 // * * * * *
17
18 // Geometry Parameters
19 inletX      0.25;
20 channelY    0.30;
21 embX        #calc "$inletX + 0.25";
22 embY        #calc "$channelY + 0.15";
23 outletX     #calc "2*$embX + $inletX";
24 depth       0.1;
25
26 // Mesh Parameters

```

```

27     z           40;
28     embx        80;
29     emby        80;
30
31     ioX         40;
32     outX        120;
33     ioY         120;
34
35     gradingX    1;
36     gradingXinv 1;
37     gradingY    2;
38     gradingYinv 0.5;
39
40     embGradingY 2;
41     embGradingYinv 0.5;
42
43     gradingZ    41;
44
45     scale 1;
46     vertices
47     (
48         // Bottom Vertices
49         (0.00 0.00 0.000)          //0
50         ($inletX 0.00 0.000)       //1
51         ($embX 0.00 0.000)        //2
52         ($outletX 0.00 0.000)      //3
53         ($outletX $channelY 0.000) //4
54         ($embX $channelY 0.000)    //5
55         ($inletX $channelY 0.000)  //6
56         (0.00 $channelY 0.000)    //7
57         ($embX $embY 0.000)       //8
58         ($inletX $embY 0.000)     //9
59
60         // Upper Vertices
61         (0.00 0.00 $depth)        //10
62         ($inletX 0.00 $depth)      //11
63         ($embX 0.00 $depth)        //12
64         ($outletX 0.00 $depth)      //13
65         ($outletX $channelY $depth) //14
66         ($embX $channelY $depth)    //15
67         ($inletX $channelY $depth)  //16

```

```

68      (0.00 $channelY $depth)           //17
69      ($embX $embY $depth)             //18
70      ($inletX $embY $depth)          //19
71  );
72
73  blocks
74  (
75      hex
76      ( 6 5 8 9 16 15 18 19)
77      embayment
78      ( $embx $emby $z)
79      simpleGrading
80      (
81          (
82              (0.1 0.2 $embGradingY)
83              (0.8 0.6 1)
84              (0.1 0.2 $embGradingYinv)
85          )
86          (
87              (0.1 0.2 $embGradingY)
88              (0.8 0.6 1)
89              (0.1 0.2 $embGradingYinv)
90          )
91          $gradingZ
92      )
93
94      hex
95      ( 0 1 6 7 10 11 16 17)
96      inlet_channel
97      ( $ioX $ioY $z)
98      simpleGrading
99      (
100         1
101         //(
102             // (0.25 0.3 $gradingX)
103             // (0.50 0.4 1)
104             // (0.25 0.3 $gradingXinv)
105             //)
106         (
107             (0.1 0.2 $gradingY)
108             (0.8 0.6 1)

```

```

109          (0.1 0.2 $gradingYinv)
110      )
111      $gradingZ
112  )
113
114
115
116      hex
117      ( 1 2 5 6 11 12 15 16)
118      middle_channel
119      ( $embx $ioY $z)
120      simpleGrading
121      (
122      (
123          (0.1 0.2 $embGradingY)
124          (0.8 0.6 1)
125          (0.1 0.2 $embGradingYinv)
126      )
127      (
128          (0.1 0.2 $gradingY)
129          (0.8 0.6 1)
130          (0.1 0.2 $gradingYinv)
131      )
132      $gradingZ
133  )
134
135      hex
136      ( 2 3 4 5 12 13 14 15)
137      outlet_channel
138      ( $outX $ioY $z)
139      simpleGrading
140      (
141          1
142          //(
143          // (0.25 0.3 $gradingX)
144          // (0.50 0.4 1)
145          // (0.25 0.3 $gradingXinv)
146          //)
147      (
148          (0.1 0.2 $gradingY)
149          (0.8 0.6 1)

```

```

150          (0.1 0.2 $gradingYinv)
151      )
152      $gradingZ
153  )
154 );
155
156     edges
157  (
158 );
159
160     boundary
161  (
162     inlet
163  {
164         type    patch;
165         faces
166         (
167             ( 0 7 17 10)
168         );
169     }
170     outlet
171  {
172         type    patch;
173         faces
174         (
175             ( 3 4 14 13)
176         );
177     }
178     bottom
179  {
180         type    wall;
181         faces
182         (
183             ( 0 1 6 7)
184             ( 1 2 5 6)
185             ( 2 3 4 5)
186             ( 6 5 8 9)
187         );
188     }
189     lateralWall
190  {

```

```

191     type    wall;
192     faces
193     (
194         ( 7 6 16 17)
195         ( 6 9 19 16)
196         ( 9 8 18 19)
197         ( 5 15 18 8)
198         ( 5 4 14 15)
199     );
200 }
201     farField
202 {
203     type    wall;
204     faces
205     (
206         ( 0 10 11 1)
207         ( 1 11 12 2)
208         ( 2 12 13 3)
209     );
210 }
211     freeSurface
212 {
213     type    wall;
214     faces
215     (
216         ( 10 11 16 17)
217         ( 11 12 15 16)
218         ( 12 13 14 15)
219         ( 16 15 18 19)
220     );
221 }
222 );
223     mergePatchPairs
224 (
225 );
226
227 // ****

```

---

## D.11 system/controlDict

```
1  /*-----* C++ -----*/\n2  | ======\n3  | \\\\ / F ield      | OpenFOAM: The Open Source CFD Toolbox\n4  | \\\\ / O peration   | Version: v1912\n5  | \\\\ / A nd         | Website: www.openfoam.com\n6  | \\\\/ M anipulation |\n7  /*-----*/\n8  FoamFile\n9 {\n10    version 2.0;\n11    format ascii;\n12    class dictionary;\n13    location system;\n14    object controlDict;\n15 }\n// * * * * *\n\n18    application          pimpleFoam;\n19    startFrom            latestTime;\n20    startTime             0;\n21    stopAt                endTime;\n22    endTime               1000;\n23    deltaT                1.0E-3;\n24    writeControl           adjustableRunTime;\n25    writeInterval          10;\n26    purgeWrite             0;\n27    writeFormat             ascii;\n28    writePrecision          6;\n29    writeCompression        yes;\n30    timeFormat              general;\n31    timePrecision           6;\n32    graphFormat             raw;\n33    runTimeModifiable      yes;\n34    adjustTimeStep          true;\n35    maxCo                  0.90;\n36    maxDeltaT              0.05;\n\n37\n38 functions\n39 {
```

```

40     turbulenceFields1
41     {
42         type          turbulenceFields;
43         libs          ("libfieldFunctionObjects.so");
44         writeControl  writeTime;
45         timeStart    150;
46         fields        (R nuTilda L k I);
47     }
48
49     Q1 //second invariant of the velocity gradient tensor
50     {
51         type          Q;
52         libs          ("libfieldFunctionObjects.so");
53         timeStart    150;
54         writeControl  writeTime;
55     }
56
57     yPlus1
58     {
59         type          yPlus;
60         libs          ("libfieldFunctionObjects.so");
61         timeStart    150;
62         writeControl  writeTime;
63     }
64
65     Co1
66     {
67         type          CourantNo;
68         libs          ("libfieldFunctionObjects.so");
69         timeStart    150;
70         writeControl  writeTime;
71     }
72
73     vorticity1
74     {
75         type          vorticity;
76         libs          ("libfieldFunctionObjects.so");
77         timeStart    150;
78         writeControl  writeTime;
79     }
80

```

```

81     wallShearStress1
82     {
83         type          wallShearStress;
84         libs          ("libfieldFunctionObjects.so");
85         timeStart    150;
86         writeControl writeTime;
87     }
88 }
89
90 LambVector1 //cross product of a velocity vector [m/s] and vorticity
91     vector [1/s]
92 {
93     type          lambVector;
94     libs          ("libfieldFunctionObjects.so");
95     libs          ("libfieldFunctionObjects.so");
96     timeStart    150;
97     writeControl writeTime;
98 }
99 //">#includeFunc absUy
100
101 UyExtract
102 {
103     type          components;
104     libs          (fieldFunctionObjects);
105     field         U;
106     timeStart    150;
107     writeControl none;
108 }
109
110 absUy
111 {
112     type          mag;
113     libs          (fieldFunctionObjects);
114     field         Uy;
115     result        absUy;
116     timeStart    150;
117     writeControl none;
118 }
119
120 surfaceInterpolate1

```

```

121 {
122     type           surfaceInterpolate;
123     libs          (fieldFunctionObjects);
124     fields         ((absUy absUySurface));
125     timeStart      150;
126     writeControl   none;
127 }
128
129 velocityInterface
130 {
131     type           surfaceFieldValue;
132     libs          (fieldFunctionObjects);
133     fields         (absUySurface);
134     operation      areaIntegrate;
135     regionType    faceZone;
136     name          interface;
137     timeStart      150;
138     executeControl timeStep;
139     executeInterval 1;
140     writeControl   timeStep;
141     writeInterval  1;
142     writeFields    false;
143 }
144
145 tracer
146 {
147     type           scalarTransport;
148     libs          ("libsolverFunctionObjects.so");
149     enabled        true;
150     timeStart      150;
151     writeControl   writeTime;
152     log            yes;
153
154     nCorr          1;
155
156     // Turbulent diffusivity;
157     alphaD          0.001;      // Molecular diffusivity
158     alphaDt         1.111;     // Turbulent diffusivity (alphaDt = 1
159             / Sct)
160
161     // Bounds the transported scalar within 0 and 1

```

```

161     bounded01           true;
162
163     //name of field
164     field                tracer;
165 }
166
167     tracerVolAverage
168 {
169     type                 volFieldValue;
170     libs                 ("libfieldFunctionObjects.so");
171
172     log                  true;
173     timeStart            150;
174     writeControl         timeStep;
175     writeInterval        1;
176     writeFields          true;
177
178     regionType          cellZone;
179     name                 porousZone;
180     operation            volAverage;
181
182     fields
183     (
184         tracer
185     );
186 }

187
188     surfaceInterpolateTracer
189 {
190     type                 surfaceInterpolate;
191     libs                 (fieldFunctionObjects);
192     fields               ((tracer tracerSurface));
193     timeStart            150;
194     writeControl         none;
195 }

196
197     tracerBottom
198 {
199     type                 surfaceFieldValue;
200     libs                 (fieldFunctionObjects);
201     fields               (tracerSurface);

```

```

202     operation           average;
203     regionType        faceZone;
204     name              interfaceBottom;
205     timeStart         150;
206     executeControl    timeStep;
207     executeInterval   1;
208     writeControl      timeStep;
209     writeInterval     1;
210     writeFields       false;
211 }
212
213 tracerMiddle
214 {
215     type               surfaceFieldValue;
216     libs               (fieldFunctionObjects);
217     fields             (tracerSurface);
218     operation          average;
219     regionType        faceZone;
220     name              interfaceMiddle;
221     timeStart         150;
222     executeControl    timeStep;
223     executeInterval   1;
224     writeControl      timeStep;
225     writeInterval     1;
226     writeFields       false;
227 }
228
229 tracerTop
230 {
231     type               surfaceFieldValue;
232     libs               (fieldFunctionObjects);
233     fields             (tracerSurface);
234     operation          average;
235     regionType        faceZone;
236     name              interfaceTop;
237     timeStart         150;
238     executeControl    timeStep;
239     executeInterval   1;
240     writeControl      timeStep;
241     writeInterval     1;
242     writeFields       false;

```

```

243     }
244
245     generalVariablesAveraging
246     {
247         type           fieldAverage;
248         libs          ("libfieldFunctionObjects.so");
249         enabled        true;
250         writeControl   writeTime;
251         timeStart      150;
252         restartOnRestart false;
253         resetOnOutput   false;
254
255         fields
256         (
257             U
258             {
259                 mean       on;
260                 prime2Mean on;
261                 base       time;
262             }
263
264             P
265             {
266                 mean       on;
267                 prime2Mean on;
268                 base       time;
269             }
270
271             Co
272             {
273                 mean       on;
274                 prime2Mean on;
275                 base       time;
276             }
277
278             yPlus
279             {
280                 mean       on;
281                 prime2Mean on;
282                 base       time;
283             }

```

```

284
285     turbulenceProperties:R
286     {
287         mean          on;
288         prime2Mean   on;
289         base          time;
290     }
291
292     vorticity
293     {
294         mean          on;
295         prime2Mean   on;
296         base          time;
297     }
298
299     lambVector
300     {
301         mean          on;
302         prime2Mean   on;
303         base          time;
304     }
305 );
306 }
307
308 #includeFunc totalTKE
309
310 totalTKEAveraging
311 {
312     type          fieldAverage;
313     libs          ("libfieldFunctionObjects.so");
314     enabled        true;
315     writeControl  writeTime;
316     timeStart     160;
317     restartOnRestart false;
318     resetOnOutput false;
319
320     fields
321     (
322         totalTKE
323         {
324             mean          on;

```

```

325             prime2Mean      on;
326             base          time;
327         }
328     );
329 }
330
331     probes
332 {
333     type          probes;
334     libs          ("libsampling.so");
335     writeControl  timeStep;
336     writeInterval 1;
337     setFormat    csv;
338
339     fields
340     (
341         p U
342     );
343
344     probeLocations
345     (
346         (0.25 0.30 0.05)    //0
347         (0.30 0.30 0.05)    //1
348         (0.35 0.30 0.05)    //2
349         (0.40 0.30 0.05)    //3
350         (0.45 0.30 0.05)    //4
351         (0.50 0.30 0.05)    //5
352     );
353 }
354
355     meanProbes
356 {
357     type          probes;
358     libs          ("libsampling.so");
359     writeControl  timeStep;
360     writeInterval 1;
361     setFormat    csv;
362     timeStart    150;
363
364     fields
365     (

```

```

366             pMean UMean pPrime2Mean UPrime2Mean
367         );
368
369     probeLocations
370     (
371         (0.25 0.30 0.05)      //0
372         (0.30 0.30 0.05)      //1
373         (0.35 0.30 0.05)      //2
374         (0.40 0.30 0.05)      //3
375         (0.45 0.30 0.05)      //4
376         (0.50 0.30 0.05)      //5
377     );
378 }
379
380 genericalPlanes
381 {
382     type           surfaces;
383     libs          ("libsampling.so");
384     writeControl   onEnd;
385
386     interpolationScheme cell;
387     surfaceFormat    raw;
388
389     surfaces
390     (
391         p00
392         {
393             type           cuttingPlane;
394             planeType     pointAndNormal;
395
396             pointAndNormalDict
397             {
398                 point        (0 0.30 0);
399                 normal       (0 1 0);
400                 zone        porousZone;
401             }
402         }
403         p01
404         {
405             type           cuttingPlane;
406             planeType     pointAndNormal;

```

```

407
408     pointAndNormalDict
409     {
410         point      (0 0.33 0);
411         normal     (0 1 0);
412         zone       porousZone;
413     }
414 }
415 p02
416 {
417     type          cuttingPlane;
418     planeType    pointAndNormal;
419
420     pointAndNormalDict
421     {
422         point      (0 0.36 0);
423         normal     (0 1 0);
424         zone       porousZone;
425     }
426 }
427 p03
428 {
429     type          cuttingPlane;
430     planeType    pointAndNormal;
431
432     pointAndNormalDict
433     {
434         point      (0 0.39 0);
435         normal     (0 1 0);
436         zone       porousZone;
437     }
438 }
439 p04
440 {
441     type          cuttingPlane;
442     planeType    pointAndNormal;
443
444     pointAndNormalDict
445     {
446         point      (0 0.42 0);
447         normal     (0 1 0);

```

```

448           zone      porousZone;
449       }
450   }
451   p05
452   {
453     type      cuttingPlane;
454     planeType pointAndNormal;
455
456     pointAndNormalDict
457   {
458     point      (0.28 0 0);
459     normal     (1 0 0);
460     zone       porousZone;
461   }
462 }
463 p06
464 {
465   type      cuttingPlane;
466   planeType pointAndNormal;
467
468   pointAndNormalDict
469   {
470     point      (0.32 0 0);
471     normal     (1 0 0);
472     zone       porousZone;
473   }
474 }
475 p07
476 {
477   type      cuttingPlane;
478   planeType pointAndNormal;
479
480   pointAndNormalDict
481   {
482     point      (0.35 0 0);
483     normal     (1 0 0);
484     zone       porousZone;
485   }
486 }
487 p08
488 {

```

```

489         type      cuttingPlane;
490         planeType   pointAndNormal;
491
492         pointAndNormalDict
493     {
494         point      (0.38 0 0);
495         normal     (1 0 0);
496         zone       porousZone;
497     }
498 }
499 p09
500 {
501         type      cuttingPlane;
502         planeType   pointAndNormal;
503
504         pointAndNormalDict
505     {
506         point      (0.42 0 0);
507         normal     (1 0 0);
508         zone       porousZone;
509     }
510 }
511 p10
512 {
513         type      cuttingPlane;
514         planeType   pointAndNormal;
515
516         pointAndNormalDict
517     {
518         point      (0.45 0 0);
519         normal     (1 0 0);
520         zone       porousZone;
521     }
522 }
523 p11
524 {
525         type      cuttingPlane;
526         planeType   pointAndNormal;
527
528         pointAndNormalDict
529     {

```

```

530             point      (0.48 0 0);
531             normal     (1 0 0);
532             zone       porousZone;
533         }
534     }
535     p12
536     {
537         type      cuttingPlane;
538         planeType pointAndNormal;
539
540         pointAndNormalDict
541         {
542             point      (0 0 0.01);
543             normal     (0 0 1);
544             zone       porousZone;
545         }
546     }
547     p13
548     {
549         type      cuttingPlane;
550         planeType pointAndNormal;
551
552         pointAndNormalDict
553         {
554             point      (0 0 0.02);
555             normal     (0 0 1);
556             zone       porousZone;
557         }
558     }
559     p14
560     {
561         type      cuttingPlane;
562         planeType pointAndNormal;
563
564         pointAndNormalDict
565         {
566             point      (0 0 0.03);
567             normal     (0 0 1);
568             zone       porousZone;
569         }
570     }

```

```

571     p15
572     {
573         type          cuttingPlane;
574         planeType    pointAndNormal;
575
576         pointAndNormalDict
577         {
578             point        (0 0 0.04);
579             normal       (0 0 1);
580             zone         porousZone;
581         }
582     }
583     p16
584     {
585         type          cuttingPlane;
586         planeType    pointAndNormal;
587
588         pointAndNormalDict
589         {
590             point        (0 0 0.05);
591             normal       (0 0 1);
592             zone         porousZone;
593         }
594     }
595     p17
596     {
597         type          cuttingPlane;
598         planeType    pointAndNormal;
599
600         pointAndNormalDict
601         {
602             point        (0 0 0.06);
603             normal       (0 0 1);
604             zone         porousZone;
605         }
606     }
607     p18
608     {
609         type          cuttingPlane;
610         planeType    pointAndNormal;
611

```

```

612         pointAndNormalDict
613     {
614         point      (0 0 0.07);
615         normal     (0 0 1);
616         zone       porousZone;
617     }
618 }
619 p19
620 {
621     type      cuttingPlane;
622     planeType pointAndNormal;
623
624     pointAndNormalDict
625     {
626         point      (0 0 0.08);
627         normal     (0 0 1);
628         zone       porousZone;
629     }
630 }
631 p20
632 {
633     type      cuttingPlane;
634     planeType pointAndNormal;
635
636     pointAndNormalDict
637     {
638         point      (0 0 0.09);
639         normal     (0 0 1);
640         zone       porousZone;
641     }
642 }
643 p21
644 {
645     type      cuttingPlane;
646     planeType pointAndNormal;
647
648     pointAndNormalDict
649     {
650         point      (0 0 0.10);
651         normal     (0 0 1);
652         zone       porousZone;

```

```

653         }
654     }
655 );
656
657     fields
658 (
659         UMean
660         pMean
661         turbulenceProperties:RMean
662         vorticityMean
663         lambVectorMean
664     );
665 }
666
667     runTimeControl1
668 {
669         type          runTimeControl;
670         libs          ("libutilityFunctionObjects.so");
671         timeStart    350;
672         writeControl  onEnd;
673         conditions
674 {
675             tracer
676 {
677                 type          minMax;
678                 functionObject tracerVolAverage;
679                 fields        (volAverage(porousZone,tracer));
680                 value         0.05;
681                 mode          minimum;
682             }
683         }
684     }
685
686     #includeFunc residuals
687 }
688
689 // ****

```

---

## D.12 system/decomposeParDict

```
1  /*-----* C++ -----*/\n2  | ====== |\n3  | \\\\ / F ield | OpenFOAM: The Open Source CFD Toolbox |\n4  | \\\\ / O peration | Version: v1912 |\n5  | \\\\ / A nd | Website: www.openfoam.com |\n6  | \\\\ / M anipulation |\n7  /*-----*/\n8\n9  FoamFile\n10 {\n11     version    2.0;\n12     format      ascii;\n13     class       dictionary;\n14     object      decomposeParDict;\n15 }\n16 // * * * * *\n17\n18\n19     method      scotch;\n20\n21\n22     scotchCoeffs\n23 {\n24 }\n25\n26     constraints\n27 {\n28         // Keep owner and neighbour on same processor for faces in zones\n29         faces\n30         {\n31             type    preserveFaceZones;\n32             zones   (interface interfaceBottom interfaceMiddle interfaceTop);\n33             enabled true;\n34         }\n35     }\n36 // ****\n
```

## D.13 system/fvSchemes

```
1  /*-----*-- C++ --*-----*/
2  | ====== |
3  | \\\    / F ield      | OpenFOAM: The Open Source CFD Toolbox |
4  | \\   / O peration   | Version: v1912 |
5  | \\ / A nd          | Website: www.openfoam.com |
6  | \\\/ M anipulation |
7  \*-----*/
8
9  FoamFile
10 {
11     version 2.0;
12     format ascii;
13     class dictionary;
14     location system;
15     object fvSchemes;
16 }
17 // * * * * *
18
19 ddtSchemes
20 {
21     default backward;
22 }
23
24 gradSchemes
25 {
26     default Gauss linear;
27 }
28
29 divSchemes
30 {
31     default none;
32     div(phi,U) Gauss LUST grad(U);
33     div(phi,nuTilda) Gauss limitedLinear 0.1;
34     div((nuEff*dev2(T(grad(U))))) Gauss linear;
35
36     div(phi,tracer) Gauss limitedLinear01 1;
37 }
38
39 interpolationSchemes
```

```

40     {
41         default      linear;
42     }
43
44     laplacianSchemes
45     {
46         default      Gauss linear orthogonal;
47     }
48
49     snGradSchemes
50     {
51         default      orthogonal;
52     }
53
54     wallDist
55     {
56         method meshWave;
57     }
58
59     fluxRequired
60     {
61         default no;
62         p ;
63         Phi ;
64     }
65
66 // ****

```

---

## D.14 system/fvSolution

```

1  /*-----* C++ -----*/
2  | ====== |
3  | \\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
4  | \\ / O peration | Version: v1912 |
5  | \\ / A nd | Website: www.openfoam.com |
6  | \\ / M anipulation |
7  \*-----*/
8
9  FoamFile

```

```

10  {
11      version 2.0;
12      format ascii;
13      class dictionary;
14      location system;
15      object fvSolution;
16  }
17 // * * * * *
18
19 PIMPLE
20 {
21     nOuterCorrectors 3;
22     nCorrectors      3;
23     nNonOrthogonalCorrectors 0;
24     pRefPoint (0.15 0.15 0.1);
25     pRefValue 0;
26
27     residualControl
28     {
29         "(p|U)"
30         {
31             tolerance    1e-04;
32             relTol       0;
33         }
34     }
35
36     relaxationFactors
37     {
38         fields
39         {
40             p           0.4;
41             pFinal      1;
42         }
43
44         equations
45         {
46             U           0.7;
47             UFinal      1;
48             nuTilda     1;
49             nuTildaFinal 1;
50         }

```

```

51         }
52     }
53
54     solvers
55     {
56         p
57         {
58             solver      GAMG;
59             smoother   GaussSeidel;
60             tolerance  1e-04;
61             relTol     0.01;
62             minIter    1;
63             maxIter    200;
64         }
65
66         pFinal
67         {
68             $p;
69             smoother   GaussSeidel;
70             tolerance  1e-04;
71             relTol     0.01;
72         }
73
74         U
75         {
76             solver      PBiCGStab;
77             preconditioner diagonal;
78             tolerance  1e-04;
79             relTol     0.01;
80             minIter    1;
81             maxIter    100;
82         }
83
84         UFinal
85         {
86             $U;
87             tolerance  1e-04;
88             relTol     0.01;
89         }
90
91     tracer

```

```

92     {
93         solver          PBiCGStab;
94         preconditioner diagonal;
95         tolerance      1e-04;
96         relTol         0.01;
97         minIter        1;
98     }
99
100    Phi
101    {
102        solver          GAMG;
103        smoother       GaussSeidel;
104        tolerance      1e-06;
105        relTol         0.01;
106        maxIter        20;
107    }
108 }
109
110    relaxationFactors
111    {
112        fields
113        {
114            p              0.4;
115            pFinal         1;
116        }
117
118        equations
119        {
120            U              0.7;
121            UFinal         1;
122            nuTilda        1;
123            nuTildaFinal   1;
124        }
125
126    }
127
128    potentialFlow
129    {
130        nNonOrthogonalCorrectors 10;
131    }
132

```

```
133 // ****
```

---

## D.15 system/setFieldsDict

```
1 /*-----* C++ -----*\ \
2 | ====== | |
3 | \ \ / F ield | OpenFOAM: The Open Source CFD Toolbox | |
4 | \ \ / O peration | Version: v1912 | |
5 | \ \ / A nd | Website: www.openfoam.com | |
6 | \ \ / M anipulation | |
7 /*-----*/
```

```
8 FoamFile
9 {
10     version    2.0;
11     format      ascii;
12     class       dictionary;
13     object      setFieldsDict;
14 }
15 // * * * * *
16
17 defaultFieldValues
18 (
19     volScalarFieldValue tracer 0
20 );
21
22 regions
23 (
24     // Setting values inside a box
25     boxToCell
26     {
27         box      (0.25 0.30 0) (0.50 0.45 0.10);
28         fieldValues
29         (
30             volScalarFieldValue tracer 1
31         );
32     }
33 );
34
35
```

```
36 // ****//
```

---

## D.16 system/topoSetDict

```
1 /*-----* C++ -----*/\n2 | ====== |\n3 | \\" / F ield | OpenFOAM: The Open Source CFD Toolbox |\n4 | \\" / O peration | Version: v1912 |\n5 | \\" / A nd | Website: www.openfoam.com |\n6 | \\"/ M anipulation |\n7 /*-----*/\n8\n9 FoamFile\n10 {\n11     version    2.0;\n12     format      ascii;\n13     class       dictionary;\n14     object      topoSetDict;\n15 }\n16 // * * * * *\n17\n18 actions\n19 (\n20 {\n21     name      porousZone;\n22     type      cellZoneSet;\n23     action    new;\n24     source    boxToCell;\n25     sourceInfo\n26 {\n27         box (0.25 0.30 0) (0.50 0.45 0.1);\n28     }\n29 }\n30 }\n31 {\n32     name      interfaceSelection;\n33     type      faceSet;\n34     action    new;\n35     source    boxToFace;
```

```

36         sourceInfo
37     {
38         box (0.25 0.2999 0) (0.50 0.3001 0.1);
39     }
40 }
41
42 {
43     name    interfaceSelection;
44     type    faceSet;
45     action   subtract;
46     source   normalToFace;
47     normal   (0 1 0);
48     cos      0.01;
49 }
50
51 {
52     name    interfaceSelection;
53     type    faceSet;
54     action   subtract;
55     source   normalToFace;
56     normal   (0 0 1);
57     cos      0.01;
58 }
59
60 {
61     name    interface;
62     type    faceZoneSet;
63     action   new;
64     source   setToFaceZone;
65     faceSet interfaceSelection;
66 }
67
68 {
69     name    interfaceBottom;
70     type    faceSet;
71     action   new;
72     source   boxToFace;
73     sourceInfo
74     {
75         box (0.25 0.2999 0) (0.50 0.3001 0.033);
76     }

```

```

77     }
78
79     {
80         name    interfaceMiddle;
81         type    faceSet;
82         action   new;
83         source   boxToFace;
84         sourceInfo
85         {
86             box (0.25 0.2999 0.033) (0.50 0.3001 0.066);
87         }
88     }
89
90     {
91         name    interfaceTop;
92         type    faceSet;
93         action   new;
94         source   boxToFace;
95         sourceInfo
96         {
97             box (0.25 0.2999 0.066) (0.50 0.3001 0.1);
98         }
99     }
100
101    {
102        name    interfaceBottom;
103        type    faceSet;
104        action   subtract;
105        source   normalToFace;
106        normal   (0 1 0);
107        cos      0.01;
108    }
109
110    {
111        name    interfaceBottom;
112        type    faceSet;
113        action   subtract;
114        source   normalToFace;
115        normal   (0 0 1);
116        cos      0.01;
117    }

```

```

118
119    {
120        name      interfaceMiddle;
121        type      faceSet;
122        action    subtract;
123        source   normalToFace;
124        normal   (0 1 0);
125        cos      0.01;
126    }
127
128    {
129        name      interfaceMiddle;
130        type      faceSet;
131        action    subtract;
132        source   normalToFace;
133        normal   (0 0 1);
134        cos      0.01;
135    }
136
137    {
138        name      interfaceTop;
139        type      faceSet;
140        action    subtract;
141        source   normalToFace;
142        normal   (0 1 0);
143        cos      0.01;
144    }
145
146    {
147        name      interfaceTop;
148        type      faceSet;
149        action    subtract;
150        source   normalToFace;
151        normal   (0 0 1);
152        cos      0.01;
153    }
154
155    {
156        name      interfaceBottom;
157        type      faceZoneSet;
158        action   new;

```

```

159         source setToFaceZone;
160         faceSet interfaceBottom;
161     }
162
163     {
164         name    interfaceMiddle;
165         type    faceZoneSet;
166         action   new;
167         source  setToFaceZone;
168         faceSet interfaceMiddle;
169     }
170
171     {
172         name    interfaceTop;
173         type    faceZoneSet;
174         action   new;
175         source  setToFaceZone;
176         faceSet interfaceTop;
177     }
178 );

```

---

## D.17 system/totalTKE

```

1  /*-----* C++ -----*/
2  | ====== |
3  | \ \ / F ield      | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O peration   | Version: v1912 |
5  | \ \ / A nd        | Website: www.openfoam.com |
6  | \ \ / M anipulation |
7  /*-----*/
8 totalTKE
9 {
10    type          coded;
11    libs          ("libutilityFunctionObjects.so");
12    name          totalTKE;
13    executeControl timeStep;
14    writeControl  writeTime;
15    timeStart     155;
16    // timeEnd      0;

```

```

17     enabled      true;
18
19 /*-----*/
20
21     Total Turbulent Kinetic Energy Evaluation
22     ** Requires fieldAverage Function to Obtain UPrime2Mean**
23     ** Resolved Reynolds Stress Tensor
24     ** Requires turbulenceFields Function to Obtain R**
25     ** Subgrid Reynolds Stress Tensor
26
27 /*-----*/
28
29     codeExecute
30     #{
31         static autoPtr<volScalarField> totalTKE;
32
33         if
34         (
35             mesh().foundObject<volSymmTensorField>("UPrime2Mean")
36             &&
37             mesh().foundObject<volSymmTensorField>("turbulenceProperties:R")
38             &&
39             mesh().foundObject<volScalarField>("totalTKE") == 0
40         )
41         {
42             Info << "Turbulent Kinetic Energy:" << endl;
43             Info << "  Initialising" << endl;
44             Info << "  Calculating" << nl << endl;
45
46             totalTKE.set
47             (
48                 new volScalarField
49                 (
50                     IOobject
51                     (
52                         "totalTKE",
53                         mesh().time().timeName(),
54                         mesh(),
55                         IOobject::NO_READ,
56                         IOobject::AUTO_WRITE
57                     ),

```

```

58         mesh(),
59         dimensionedScalar
60         (
61             "totalTKE",
62             dimensionSet(0,2,-2,0,0,0,0),
63             0
64         )
65     )
66 );
67
68 const volSymmTensorField& R =
69     mesh().lookupObjectRef<volSymmTensorField>("turbulenceProperties:R");
70
71 const volSymmTensorField& UPrime2Mean =
72     mesh().lookupObjectRef<volSymmTensorField>("UPrime2Mean");
73
74 volScalarField& totalTKE =
75     mesh().lookupObjectRef<volScalarField>("totalTKE");
76 totalTKE = (0.5 * tr(R)) + (0.5 * tr(UPrime2Mean));
77 }
78
79 else if
80 (
81     mesh().foundObject<volSymmTensorField>("UPrime2Mean")
82     &&
83     mesh().foundObject<volSymmTensorField>("turbulenceProperties:R")
84     &&
85     mesh().foundObject<volScalarField>("totalTKE")
86 )
87 {
88     Info << "Turbulent Kinetic Energy:" << endl;
89     Info << " Calculating" << nl << endl;
90
91     const volSymmTensorField& R =
92         mesh().lookupObjectRef<volSymmTensorField>("turbulenceProperties:R");
93     const volSymmTensorField& UPrime2Mean =
94         mesh().lookupObjectRef<volSymmTensorField>("UPrime2Mean");
95
96     volScalarField& totalTKE =
97         mesh().lookupObjectRef<volScalarField>("totalTKE");
98     totalTKE = (0.5 * tr(R)) + (0.5 * tr(UPrime2Mean));
99 }

```

```

93
94     else
95     {
96         Info << "Turbulent Kinect Energy:" << endl;
97         Warning << endl
98             << "    Unable to Calculate Turbulent Kinect Energy" << endl
99             << "    UPrime2Mean and/or R Unavailable" << endl
100            << "    Enable fieldAverage and turbulenceFields Functions"
101            << nl << endl;
102     }
103 }

```

---

## D.18 allClear

---

```

1  #!/bin/bash
2
3  # Saves 0.orig from being deleted
4  mv 0.orig foo
5
6  # Deletes Files
7  rm -r constant/polyMesh
8  rm -r processor*/
9  rm -r dynamicCode
10 rm -r log
11 rm -r 0.* [1-9]*
12
13 # Restores 0.orig
14 mv foo 0.orig
15
16 # Creates file for paraview
17 CASE=${PWD##*/}
18 touch $CASE.foam

```

---

## D.19 mesh

---

```
1  #!/bin/sh
```

```

2
3 case=${PWD##*/}
4
5 rm -rf log p* 0
6 mkdir log
7 cp -r 0.orig 0
8
9 { # try
10     echo -e "Compiled variables:\n"
11     blockMesh > log/blockMesh.log &&
12     printf '%*s' "${COLUMNS:-$(tput cols)}" , , | tr , , -
13     echo -e "blockMesh completed without errors"
14     #save your output
15
16 } || { # catch
17     # save log for exception
18     echo -e "An error occurred on blockMesh"
19     exit 1
20 }
21 {
22     topoSet >log/topoSet.log &&
23     echo -e "topoSet completed without errors"
24 } || {
25     echo -e "An error occurred on topoSet"
26     exit 1
27 }
28 {
29     checkMesh -allGeometry -allTopology -writeAllFields -writeSets vtk >
30         log/checkMesh.log &&
31     echo -e "checkMesh completed without errors"
32 } || {
33     echo -e "An error occurred on checkMesh"
34     exit 1
35 }
36 rm -rf dynamicCode
37
38 {
39     setFields > log/setFields.log &&
40     echo -e "setFields completed without errors"
41 } || {

```

```

42     echo -e "An error occurred on setFields"
43     exit 1
44 }
45
46 echo -e "Mesh constructed and checked."
47 echo -e "Tracer fields set."

```

---

## D.20 ramCache

```

1 #!/bin/bash
2
3 free && sync && echo 3 > /proc/sys/vm/drop_caches && free

```

---

## D.21 reconstructParParallel

```

1 #!/bin/bash
2 echo "
3     K. Wardle 6/22/09, modified by H. Stadler Dec. 2013, minor fix Will
4         Bateman Sep 2014.
5     bash script to run reconstructPar in pseudo-parallel mode
6     by breaking time directories into multiple ranges
7
8 USAGE="
9
10    USAGE: $0 -n <NP> -f fields -o <OUTPUTFILE>
11    -f (fields) is optional, fields given in the form T,U,p; option is
12        passed on to reconstructPar
13    -t (times) is optional, times given in the form tstart,tstop
14        -o (output) is optional
15
16 #TODO: add flag to trigger deletion of original processorX directories after
17     successful reconstruction
18 # At first check whether any flag is set at all, if not exit with error message
19 if [ $# == 0 ]; then
20     echo "$USAGE"
21     exit 1

```

```

20   fi
21
22 #Use getopt to pass the flags to variables
23 while getopt "f:n:o:t:" opt; do
24   case $opt in
25     f) if [ -n $OPTARG ]; then
26       FIELDS=$(echo $OPTARG | sed 's/,/ /g')
27     fi
28     ;;
29     n) if [ -n $OPTARG ]; then
30       NJOBS=$OPTARG
31     fi
32     ;;
33     o) if [ -n $OPTARG ]; then
34       OUTPUTFILE=$OPTARG
35     fi
36     ;;
37     t) if [ -n $OPTARG ]; then
38       TLOW=$(echo $OPTARG | cut -d ',' -f1)
39       THIGH=$(echo $OPTARG | cut -d ',' -f2)
40     fi
41     ;;
42   \?) 
43     echo "$USAGE" >&2
44     exit 1
45     ;;
46   :)
47     echo "Option -$OPTARG requires an argument." >&2
48     exit 1
49     ;;
50   esac
51 done
52
53 # check whether the number of jobs has been passed over, if not exit with
54 # error message
54 if [[ -z $NJOBS ]]
55 then
56   echo "
57     the flag -n <NP> is required!
58   "
59   echo "$USAGE"

```

```

60     exit 1
61 fi
62
63 APPNAME="reconstructPar"
64
65 echo "running $APPNAME in pseudo-parallel mode on $NJOBS processors"
66
67 #count the number of time directories
68 NSTEPS=$(( $(ls -d processor0/[0-9]*/ | wc -l)-1))
69 NINITAL=$(ls -d [0-9]*/ | wc -l) ##count time directories in case root dir,
    this will include 0
70
71 P=p
72 #find min and max time
73 TMIN=$(ls processor0 -1v | sed '/constant/d' | sort -g | sed -n 2$P) #
    modified to omit constant and first time directory
74 #TMIN='ls processor0 | sort -nr | tail -1'
75 TMAX=$(ls processor0 -1v | sed '/constant/d' | sort -gr | head -1) # modified
    to omit constant directory
76 #TMAX='ls processor0 | sort -nr | head -1'
77
78 #Adjust min and max time according to the parameters passed over
79 if [ -n "$TLOW" ]
80 then
81     TMIN=$(ls processor0 -1v | sed '/constant/d' | sort -g | sed -n 1$P) # now
        allow the first directory
82     NLOW=2
83     NHIGH=$NSTEPS
84     # At first check whether the times are given are within the times in the
        directory
85     if [ $(echo "$TLOW > $TMAX" | bc) == 1 ]; then
86         echo "
87             TSTART ($TLOW) > TMAX ($TMAX)
88             Adjust times to be reconstructed!
89             "
90         echo "$USAGE"
91         exit 1
92     fi
93     if [ $(echo "$THIGH < $TMIN" | bc) == 1 ]; then
94         echo "
95             TSTOP ($THIGH) < TMIN ($TMIN)

```

```

96     Adjust times to be reconstructed!
97     "
98     echo "$USAGE"
99     exit 1
100    fi
101
102    # Then set Min-Time
103    until [ $(echo "$TMIN >= $TLOW" | bc) == 1 ]; do
104        TMIN=$(ls processor0 -1v | sed -n $NLOW$P)
105        NSTART=$((NLOW))
106        let NLOW=NLOW+1
107    done
108
109    # And then set Max-Time
110    until [ $(echo "$TMAX <= $THIGH" | bc) == 1 ]; do
111        TMAX=$(ls processor0 -1v | sed -n $NHIGH$P)
112        let NHIGH=NHIGH-1
113    done
114
115    # Finally adjust the number of directories to be reconstructed
116    NSTEPS=$((NHIGH-NLOW+3))
117
118    else
119
120        NSTART=2
121
122    fi
123
124    echo "reconstructing $NSTEPS time directories"
125
126    NCHUNK=$((NSTEPS/$NJOBS))
127    NREST=$((NSTEPS%$NJOBS))
128    TSTART=$TMIN
129
130    echo "making temp dir"
131    TEMPDIR="temp.parReconstructPar"
132    mkdir $TEMPDIR
133
134    PIDS=""
135    for i in $(seq $NJOBS)
136    do

```

```

137 if [ $NREST -ge 1 ]
138 then
139   NSTOP=$((NSTART+$NCHUNK))
140   let NREST=$NREST-1
141 else
142   NSTOP=$((NSTART+$NCHUNK-1))
143 fi
144 TSTOP=$(ls processor0 -1v | sed -n $NSTOP$P)
145
146
147 if [ $i == $NJOBS ]
148 then
149   TSTOP=$TMAX
150 fi
151
152 if [ $NSTOP -ge $NSTART ]
153 then
154   echo "Starting Job $i - reconstructing time = $TSTART through $TSTOP"
155   if [ -n "$FIELDS" ]
156     then
157       $($APPNAME -fields "($FIELDS)" -time $TSTART:$TSTOP >
158           $TEMPDIR/output-$TSTOP &)
159   echo "Job started with PID $(pgrep -n -x $APPNAME)"
160   PIDS="$PIDS $(pgrep -n -x $APPNAME)" # get the PID of the latest (-n) job
161   exactly matching (-x) $APPNAME
162   else
163     $($APPNAME -time $TSTART:$TSTOP > $TEMPDIR/output-$TSTOP &)
164   echo "Job started with PID $(pgrep -n -x $APPNAME)"
165   PIDS="$PIDS $(pgrep -n -x $APPNAME)"
166   fi
167   fi
168
169 let NSTART=$NSTOP+1
170 TSTART=$(ls processor0 -1v | sed -n $NSTART$P)
171 done
172
173 #sleep until jobs finish
174 #if number of jobs > NJOBS, hold loop until job finishes
175 NMORE_OLD=$(echo 0)
176 until [ $(ps -p $PIDS | wc -l) -eq 1 ]; # check for PIDS instead of $APPNAME
177   because other instances might also be running

```

```

175 do
176   sleep 10
177   NNOW=$(ls -d [0-9]*/ | wc -l) ##count time directories in case root dir,
178     this will include 0
179   NMORE=$(echo $NSTEPS-$NNOW+$NINITAL | bc) ##calculate number left to
180     reconstruct and subtract 0 dir
181   if [ $NMORE != $NMORE_OLD ]
182     then
183       echo "$NMORE directories remaining..."
184   fi
185   NMORE_OLD=$NMORE
186 done
187
188 #combine and cleanup
189 if [ -n "$OUTPUTFILE" ]
190   then
191     #check if output file already exists
192     if [ -e "$OUTPUTFILE" ]
193       then
194         echo "output file $OUTPUTFILE exists, moving to $OUTPUTFILE.bak"
195         mv $OUTPUTFILE $OUTPUTFILE.bak
196       fi
197
198     echo "cleaning up temp files"
199     for i in $(ls $TEMPDIR)
200       do
201         cat $TEMPDIR/$i >> $OUTPUTFILE
202       done
203     fi
204
205 rm -rf $TEMPDIR
206
207 echo "finished"

```

---