Thesis Consent Form

THE UNIVERSITY OF AUCKLAND LIBRARY

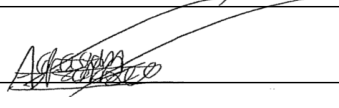
This thesis may be consulted for the purposes of research or private study provided that due acknowledgement is made where appropriate and that permission is obtained before any material from the thesis is published. Students who do not wish their work to be available for reasons such as pending patents, copyright agreements, future publication or to protect confidential information should seek advice from the Graduate Centre as to restricted use or embargo.

Author of thesis	Alfonso Gastélum Strozzi
Title of thesis	Smoothing Particle Hydrodynamics parallel implementation for numerical modelling of solid-fluid interactions
Name of degree	Doctor of Philosophy
Date Submitted	01/09/2011

Print Format ((Tick the boxes that apply)	
✔	I agree that the University of Auckland Library may make a copy of this thesis available for the collection of another library on request from that library.
✔	I agree to this thesis being copied for supply to any person in accordance with the provisions of Section 56 of the Copyright Act 1994.

Digital Format - PhD theses
I certify that a digital copy of my thesis deposited with the University will be the same as the final officially approved version of my thesis. Except in the circumstances set out below, no emendation of content has occurred and I recognise that minor variations in formatting may occur as a result of the conversion to digital format. Access to my thesis may be limited for a period of time specified by me at the time of deposit. I understand that if my thesis is available online for public access it can be used for criticism, review, news reporting, research and private study.
Digital Format– Masters theses
I certify that a digital copy of my thesis deposited with the University will be the same as the print version submitted for examination. Except in the circumstances set out below, no emendation of content has occurred and I recognise that minor variations in formatting may occur as a result of the conversion to digital format. Access will normally only be available to authenticated members of the University of Auckland, but I may choose to allow public access under special circumstances. I understand that if my thesis is available online for public access it can be used for criticism, review, news reporting, research and private study.

Copyright (Digital Format Theses) (Tick **ONE box only**)	
✔	I confirm that my thesis does not contain material for which the copyright belongs to a third party, ***(or)*** that the amounts copied fall within the limits permitted under the Copyright Act 1994.
☐	I confirm that for all third party copyright material in my thesis, I have obtained written permission to use the material and attach copies of each permission, ***(or)*** I have removed the material from the digital copy of the thesis, fully referenced the deleted materials and, where possible, provided links to electronic sources of the material.

Signature  Date 01/09/2011

Comments on access conditions

Graduate Centre only: Digital copy accepted ☐ Signature Date

Smoothing Particle Hydrodynamics parallel implementation for numerical modelling of solid-fluid interactions

by

Alfonso Gastélum Strozzi

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy, The University of Auckland, 2011.

Supervisor: Patrice Delmas
Supervisor: Georgy Gimel'farb
Supervisor: Jorge Marquez

Abstract

In this work is presented an efficient parallel implementation of a Lagrangian mesh-free method based on the Smoothing Particle Hydrodynamics technique to analyse the behaviour of solids and fluids.

The proposed solution makes use of different techniques to improve the computational complexity of the simulation. To improve the performance, a parallel solution of the SPH numerical solution is implemented on the GPU using the NVIDIA CUDA architecture

Both the serial and the parallel implementation of SPH used benchmark experiments to compare effectiveness under different conditions. Results are provided with a set of discriminants that allow the program to select which of the implementation (serial or parallel) is best suited to solve the model studied.

The SPH modelling implementation is used to solve fluid flows simulation in a soil core's pore structure. The study of this problem is important to understand how heterogeneous soil structure affects water and solute transfer. This helps to obtain accurate predictions of contaminants residence time (average amount of time that a contaminant particle spends inside the pore network) and the quantity leached in the zone represented by the structures.

In order to have a method to quantify the accuracy of the results provided by the SPH implementation the commercial software COMSOL was used to obtain the numerical results on the same models solved by the SPH implementation. Finally both results are compared against the expected theoretical results.

The second application of the developed computational implementation is the study of the deformation of solid objects under stress deformation. The SPH model in conjunction with a stereo technique provides a method capable of defining the solid material variables. The obtained variables can be used in solid deformation modelling using SPH.

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Co-Authorship Form

Graduate Centre
ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Abstract.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 6, section 5. Publication: Y.H. Chan, A. Gastelum, A. Lau, et al. (2009). Modelling of elastic deformation using stereo vision and smooth particles hydrodynamics. 24th International Conference Image and Vision Computing New Zealand, IVCNZ 2009. 23/11/2009 - 25/11/2009, Wellington, New Zealand, IEEE.

Nature of contribution by PhD candidate	The candidate made use of the SPH numerical solution to solve the solid deformation model using the data obtained from the stereo system.
Extent of contribution by PhD candidate (%)	50

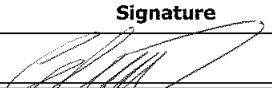
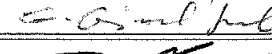
CO-AUTHORS

Name	Nature of Contribution
Yuk Hin Chan	Development of the stereo vision system used to obtained the experimental data. Principal author of the manuscript.
Georgy Gimel´farb	Theoretical background on the numerical solution of stereo-vision problems and revised the manuscript.
Patrice Delmas	Development of the equipment to obtained the range data and revised the manuscript.

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

Name	Signature	Date
Yuk Hin Chan		01/09/2011
Georgy Gimel´farb		01/09/2011
Patrice Delmas		01/09/2011

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Co-Authorship Form

Graduate Centre
ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Abstract.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 7, pages 109 to 177. Publication: C. Duwing, P. Delmas, A. Gastelum, et al. (2010). Modelling of water flow through soil pores observed by x-ray tomography. 1st International Conference and Exploratory Workshop on Soil Architecture and Physico-Chemical Functions, CESAR 2010. 30/11/2010 - 02/12/2010, Blichers Alle, Denmark.

Nature of contribution by PhD candidate	The candidate made use of his SPH implementation to obtained the numerical results of the flow inside soil pore models. He also made the necessary image processing on the database to build such models and compare the results against the ones obtained by the commercial software COMSOL.
Extent of contribution by PhD candidate (%)	60

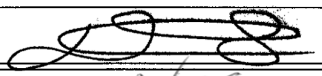
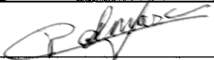
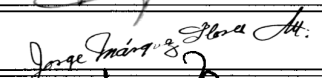
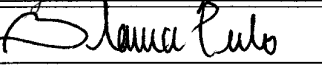
CO-AUTHORS

Name	Nature of Contribution
Celine Duwing	Soil science specialist. Verified the equations used and the consistency of the results obtained. Wrote the conclusions on the soil science side. Edited and revised the manuscript.
Patrice Delmas	Supervisor. Acquired the data at 3S lab CTscan facility. Verified the Image Processing applied by the candidate and revised the manuscript.
Jorge Marquez	Co-supervisor. Verified the mathematical morphology equations and revised the manuscript.
Blanca Prado	Provided the soil core samples.
P. Charrier	Managed the CT scan facilities during the imaging experiment. Supervised the imaging experiment.

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

Name	Signature	Date
Celine Duwing		31/08/2011
Patrice Delmas		31/08/2011
Jorge Marquez		31/08/2011
Blanca Prado		30/08/2011

Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisor, Dr. Patrice Delmas, whose expertise, understanding, and support, added considerably to my graduate experience. I appreciate his invitation to be part of the Intelligent Vision Systems New Zealand laboratory at the University of Auckland were I was able to develop my research.

I would also like to thank the other members of my committee, Dr. Jorge Marquez for being an active part of my academic life since my master degree, for helping me along the way with his advises and knowledge to improve my scientific understanding. Finally a sincere thank you to Associate-Prof Georgy Gimel'farb for the assistance he providedme with my research and for all the occasions when he took time from his busy life to share a good talk with me.

A very special thanks to my fellow PhD students Edwin Chan and Minh Nguyen for the company during the long days of works in the laboratory and their ideas and suggestions that helped me improve the level of my research.

I would also like to thank the students in the Intelligent Vision Systems New Zealand laboratory, Anthony Lau, Rui Gong and Sina Masoud Ansari, you made my days in the laboratory an interesting experience. Even more I want to thank you all for enduring my demands in our daily research.

And I am deeply grateful to the National Council of Science and Technology in Mxico (CONACYT) for the trust and support that they gave me in order to study in New Zealand and to the University of Auckland and the Computer Sciences Department for the financial support that made my research possible.

I would also like to thank my family for the support they provided me through my entire life and in particular during my years in the doctoral degree. Without they love, encouragement and emotional support, I would not have finished this thesis.

Finally I would like to thank Claudia Dominguez Garcia for been an important part of this experiences and for sharing the long hours of work supporting me and giving me encouragement to continue.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols and Abbreviations

Abbreviation	Description	Definition
BC	Boundary Conditions	page 8
BM	Biomechanical Modelling	page 16
CFD	Computational Fluid Dynamics	page 7
CG	Computer Graphics	page 16
CM	Computational Mechanics	page 7
CP	Computational Physics	page 5
CS	Computer Sciences	page 6
CSM	Computational solid mechanics	page 7
CUDA	Compute Unified Device Architecture	page 2
FEM	Finite Element Method	page 10
GPU	Graphics processing Unit	page 1
IC	Initial Conditions	page 8
IVS	Intelligent Vision Systems New Zealand	page 5
ODE	Ordinary Differential Equations	page 8
PDE	Partial Differential Equations	page 8
REV	Representative Elemental Volume	page 123
SPH	Smoothing Particle Hydrodynamic	page 8

Physics Symbols

Symbol	Units	Description	Definition
$\mathbf{b}$	$[N]$	Body forces	page 30
e_1, e_2, e_3	Dimensionless	Cartesian axis of references	page 38
S	$[m^2]$	Element surface	page 27
V	$[m^3]$	Element volume	page 26
v	$[m/s]$	Darcy flow	page 149
$\delta(\boldsymbol{r} - \boldsymbol{r'})$	Dimensionless	Dirac delta function	page 18
u	$[m]$	displacement vector	page 39
e	$[J]$	Energy	page 31
a	$[m/s^2]$	Eulerian acceleration	page 18
ν	$[m/s]$	Element velocity	page 18

Physics Symbols

Symbol	Units	Description	Definition
P	Dimensionless	Fluid particle	page 16
ρ	$[kg/m^3]$	Fluid particle density	page 16
Δm	$[kg]$	Fluid particle mass	page 16
ΔV	$[m^3]$	Fluid particle volume	page 16
$\mathbf{Pk_i}$	Dimensionless	Histogram peak	page 120
K	$[mm/s]$	Hydraulic conductivity	page 119
p	$[Pa]$	Isotropic pressure	page 32
$\rho\frac{D\nu}{Dt}$	$[kg \cdot \frac{m}{s}]$	Momentum	page 31
$\mathbf{F}$	$[N]$	Net forces	page 30
$\overline{\mathbf{p}}$	Dimensionless	Normalised particle position with respect to the tree volume	page 57
$\mathbf{p}$	$[m]$	Particle position on the Three-dimensional space	page 16
Q_p	$[m^3/s]$	Poiseuille law flow	page 174
Po_ℓ	$[m]$	Pore effective length	page 128
R_h	Dimensionless	Pore hydraulic radius	page 128
Po_α	$[m^2]$	Pore inlet area	page 128
κ	$[m^2]$	Pore permeability	page 86
Po_ς	$[m]$	Pore skeleton	page 127
Po_β	$[m^2]$	Pore outlet area	page 128
Po_τ	Dimensionless	Pore tortuosity	page 128
Q	$[m^3/s]$	Rate of flow	page 26
$\mathbf{fs}$	$[N]$	Surface forces	page 30
W	Dimensionless	Smoothing kernel function	page 19
h	$[m]$	Smoothing length	page 19
e_i	$[J]$	SPH particle energy	page 34
$\langle\rangle$	Dimensionless	SPH numerical approximation operand	page 20
ρ_i	$[kg/m^3]$	SPH particle density	page 20
m_i	$[kg]$	SPH particle mass	page 20
$\mathbf{p_i}$	$[m]$	SPH particle position	page 57
r_i	$[m]$	SPH particle position from system origin	page 20
ν_i	$[m/s]$	SPH particle velocity	page 33
μ_i	$[Pa \cdot s]$	SPH viscous coefficient	page 33
$\epsilon_i^{\alpha\beta}$	Dimensionless	SPH strain	page 33
ϵ	Dimensionless	Strain	page 31
τ	$[Pa]$	Stress	page 31
σ	$[Pa]$	Stress tensor	page 32
Q	$[m^3/s]$	Volumetric flow	page 149

Computational Symbols

Symbol	Description	Definition
gn	Virtual representation of a node using the CUDA index structure	page 62
pi	Integer representation of the space position of particles	page 58

Computational Symbols

Symbol	Description	Definition
n	Structure containing the node information	page 56
pointd	Three double values structure	page 56
T	Structure containing the tree structure	page 56
Tl	Top tree level needed to solve the problem on the CUDA architecture.	page 60
uint32	Unsigned 32 bits integer	page 56

Chapter 1

INTRODUCTION

1.1 Thesis objectives

The purpose of this work was to build a computationally efficient mesh-free Smoothing Particle Hydrodynamics (SPH) implementation that can be used to simulate the behaviour of solid and fluid materials and their interactions.

In this thesis the SPH implementation is used to obtain the numerical solution of problem such as fluid-solid interaction, stereo-vision solid deformation measurement and fluid flow estimation in porous media.

Since the computational solution will be applied to different problems, the final set of computational tools that integrate the SPH implementation should be easily adaptable to analyse different physical problems and materials (i.e. solids, fluids, gases) interaction.

The selection of the numerical solution method and the problems modelled were made takes into account the dataset, equipments and processing capabilities available in the IVS laboratory:

- **Data acquisition techniques:** The experimental data available at the beginning of the research came from two imaging sources:

  - Range (i.e. depth) and image data acquired using the stereo vision systems developed in the IVS laboratory. This data is used for solid calibration computational implementation (see Chapter 6.4.4) to obtain the material studied characteristic constants (e.g. Elastic modulus, Viscosity coefficient and Young modulus).
  - Image data obtained by Computer Tomography of porous media materials (CT-images). This type of data provides with the geo-morphological characteristics of soil core pore structures where the fluid flow SPH simulation take place.

- **Computer equipment:** The IVS laboratory main area of study is on the computer vision field and its main specialization is in the computer implementation of stereo-vision techniques.

  In the IVS laboratory the computational implementation of problems that require a real time solution makes use of a Graphics Processing Unit (GPU) hardware. GPUs are used to obtain parallel implementations of the solutions that can, in some cases, improve the computational time of the applications.

For the SPH implementation it was also decided to use the GPUs available at that time in the IVS laboratory (GPUs models GTX 480 and 570) and design the necessary algorithms to obtain a parallel implementation.

- **GPU Language:** In todays market there are three popular architecture options to perform GPU computing.

Test set-up		
Architecture	Company	Beta release date
CUDA	NVIDIA	06/07
DirectCompute	Microsoft	03/09
OpenCL	Khronos Group	NVIDIA API 09/09 ATI API 08/09

Table 1.1: Different direct computing architectures that can be use on GPUs to obtain parallel numerical solutions.

At the time the parallel development of the SPH was started, the Compute Unified Device Architecture (CUDA) [1] was the only stable (not alpha or beta) release option for direct computing on GPUs. With this in mind It was decided to invest in NVIDIA GPUs and use CUDA for the parallel implementation of this thesisŕesearch.

The thesis also accounts for the requirements needed to model the different materials physical properties. The computational solution needs to provide for the numerical solutions for the different mathematical equations that describe the behaviour of solid or fluid materials under the different conditions studied in this thesis.

The continuous experimental data obtained at the IVS laboratory (e.g. depth maps from stereo-videos and CT-images database) is easily represented by moving elements (tracking points) inside the continuum. With this in mind a Lagrangian approach was selected to build the computational solution for the phenomena studied.

For the model representation a mesh-free method was selected. The mesh-free approach is used to represent both solids and fluids.

The original hypothesis and premises of the work are:

- Boundary condition approximation: After reviewing the literature it was decided to use **mesh-free particle elements to describe and analyse fluid-solid interaction. The mesh-free individual elements do not make use of a pre-defined topological structure (mesh) and are free to adapt to changes of the boundary without the need to redefine the topological structure of the problem. In table 1.2 a list of the differences between mesh and mesh-free methods is presented.**

- Experimental stereo vision data: Data from the stereo vision system is easily interpreted as individual spatial points moving over time (three dimensional pixel position). It was assumed **that the moving points obtained from stereo and optical flow [2, 3] could serve as control data on a Lagrangian solution system** (see Chapter 2.2 for a description on the Lagrangian references frame).

- Parallel implementation: **The initial idea was that each spatial element (discrete representation of the material continuum) represented by a particle on the SPH computational**

solution could be defined by a set of neighbourhood particles with a neighbour size small enough to define all the interacting particles whitin the memory restrictions on the GPU architecture.

- CT-scan data resolution: The CT-scanner used to image the soil samples is a X-ray CT scanner from RX solutions with resolution of 87 m, time and sample size constraints the resolution and limits the data obtained to the micrometre level. It is assumed that for the flow on porous medium model **it is possible to build the soil macro pore network and simulate realistic preferential flow even when the micro-pore structures are absent**.

This thesis presents both the CPU serial solution and the CUDA parallel SPH solution. Parallel octree elements are used to solve problems specific to fluid modelling with solid boundary interaction focusing on the fluid flow inside pore structures obtained from the CT scans. An extension of the method to solid dynamics and the use of an inverse SPH method to explore experimental data provided by a stereo vision system are also presented.

A review of the methodology is followed by a discussion of computational physics and its implementation with mesh and mesh-free methods.

1.2 Contributions

The main contributions of this thesis are:

- **Parallel SPH solution:** The solid-fluid final computational solution using the CUDA architecture to provide a solution faster than its serial counterpart. All the methods presented in the thesis have both a serial and parallel version of our own SPH implementation.
- **Stereo system for morphological model parameters calibration:** A method is presented to apply a stereo system range data of real objects deformation on a Lagrangian computational solution to calibrate the mechanical properties of the object.
- **Parallel tree implementation:** The use of a GPU multi-core approach requires that the algorithms are implement considering the number of cores available. A new parallel technique to solve the tree structure creation, update and destruction is introduced in Chapter 5.3.3.
- **Fluid - Soil study implementation:** A new SPH implementation for the study of porous medium preferential flow on macro pores is presented, the database for the soil model walls are based on a 3D model obtained from a set of CT-scan images acquired using the X-ray CT scanner from RX solutions available at the 35-R laboratory.
- **Viscoelastic solid implementation:** The study of viscoelastic materials needs a different set of constitutive equations in order to build the SPH solution. In this thesis a new approach that uses a video-stereo system to obtain the displacement field on the surface of the object under stress is introduce. The experimental data is use to solve the computational solid model on the SPH implementation and the material characteristics of the solid under stress are obtain.

1.3 PhD Publications

The material and research resulting from the studies of my doctoral research have been published in the peer-reviewed conference papers listed below:

- C. Duwin, P. Delmas, A. Gastelum, et al. (2010). Modelling of water flow through soil pores observed by x-ray tomography. 1st International Conference and Exploratory Workshop on Soil Architecture and Physico-Chemical Functions, CESAR 2010. 30/11/2010 - 02/12/2010, Blichers Alle, Denmark.

- A. Gastelum, P. Delmas, M. Lefrancq, et al. (2010). Visualising 3D porous media fluid interaction using x-ray ct data and smooth particles hydrodynamics modelling. 25th international conference of image and vision computing new zealand, IVCNZ 2010. 8/11/2010 - 9/11/2010, Queenstown, New Zealand, IEEE.

- Y. H. Chan, A. Gastelum, A. Lau, et al. (2009). Modelling of elastic deformation using stereo vision and smooth particles hydrodynamics. 24th International Conference Image and Vision Computing New Zealand, IVCNZ 2009. 23/11/2009 - 25/11/2009, Wellington, New Zealand, IEEE online proceedings.

- C. Duwig, A. Gastelum, P. Delmas, et al. (2009). Modelling of water transport through an andosol: use of ct scan images and smooth particles hydrodynamics approach. Neuvième édition des journes d'etude sur les milieux poreux, JEMP2009. 21/10/2009 - 22/10/2009, Paris, France.

- A. Gastelum, J. Marquez, F. Trujillo, et al. (2009). 3d porous media liquid-solid interaction simulation using sph modeling and tomographic images, MVA2009. IAPR conference on machine vision applications, MVA2009. 20/05/2009-22/05/2009, Tokio, Japan, IAPR.

- A. Gastelum, J. Marquez, P. Delmas, et al. (2008). A mesh free mechanic model of the upper gastrointestinal system. 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC2008. 20/08/2008 - 24/08/2008, Vancouver, Canada, IEEE.

- A. Gastelum, M. Krueger, J. Marquez, et al. (2008). Automatic 3d lip shape segmentation and modelling. 23th International Conference Image and Vision Computing New Zealand, IVCNZ 2008. 26/11/2008 - 28/11/2008, Christchurch, New Zealand, IEEE.

- A. Gastelum, J. Marquez, G. Gimel'farb, et al. (2008). 3d lip shape sph-based evolution. Simposio mexicano en cirugía asistida por computadora y procesamiento de imágenes médicas, MEXCAS 2008. 17/11/2009 - 18/11/2009, Mexico City, Mexico.

- J. Marquez, M. Padilla and A. Gastelum (2007). Image-fusion operators for 3d anatomical and functional analysis of the brain. 29rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC2007. 23/08/2007 - 26/08/2007, Lyon, France, IEEE.

- A. Gastelum and J. Marquez (2007). Modelling interactions with a computer model of a soft tissue organ using a smooth particles system. Joint meeting of the Southwest Regional Chapter of the American Association of Physicists in Medicine (sw-aapm) and the Federación Mexicana de Organizaciones de Física Médica (fmofm), an international medical physics conference. 16/03/2007 - 18/03/2007 Queretaro, Mexico, AAPM.

- A. Gastelum, P. Delmas and J. Marquez (2006). Texture generation for the computer representation of the upper gastrointestinal system. 22nd International Conference Image and Vision Computing New Zealand, IVCNZ 2006. 27/11/2006 - 29/11/2006, Great Barrier Island, New Zealand.

A book chapter on SPH applied to viscoelastic modelling of the lips was published in:

- A. Gastelum, P. Delmas, J. Marquez, A. Woodward, J. James, M. Lievin and G. Gimelfarb (2009). 3D lip shape SPH based evolution using prior 2D dynamic lip features extraction and static 3D lip measurements. Editors Alan Wee and Chung Liew, IGI Global.

1.4 Thesis outline

This thesis describes the development of an SPH algorithm for the numerical solution of fluid-solid interaction in various applications.

Chapter 1: Introduces the use of Computational Physics (CP) to obtain numerical solution to physics problems using a discrete representation of continuous volume. The discrete representation in some of the solutions to CP problems use mesh-centred methods, while other focuses on mesh-free methods. This chapter also discuss our mesh-free methods to solve fluid-solid dynamic problems.

Chapter 2: Introduces the Smoothing Particle Hydrodynamic method using standard notation and describes its application to continuous CP problems. Requirements for a computational system to describe the problem of interest with a set of nodes and obtain a numerical solution are outlined.

Chapter 3: Considers an SPH implementation for fluid flow dynamics problems and presents the basic SPH method for fluid based on the technique presented in [4]. The mathematics and physics notation equations in this chapter follow the standard introduced notations in [5, 6]. The computational solution follows my own interpretation and implementation. The programming solution was developed with the aim to obtain a serial and parallel CUDA implementation.

Chapter 4: Presents an SPH implementation for solid dynamics problems. The basic SPH theory present is applied to the solid dynamics as in accord with [7]. Some changes to the definitions introduced in [7] are presented. The most important changes relate to: the viscoelastic implementation; the octree neighbourhood solver; and the CUDA implementation of the computational solution. The solid-case implementation is my own development. A collaboration with the students of the Intelligent Vision System laboratory (IVS) was establish, to use a stereo-vision system which measure small surface deformations of solids. I introduced the numerical solution and parallel implementation of the stereo algorithm and optical stereo flow implementation while the students provided with the original serial stereo algorithm and real time video systems.

Chapter 5: This chapter outlines and details computational requirements to find all the neighbourhoods and collisions of the mesh-free elements (particles). This is the core of my algorithm set to solve the most expensive problems in the numerical simulation. The fast space search [8] using octree and Morton keys is used as the tools for developing the SPH search engine. The chapter presents a new CUDA implementation I developed and the tools to solve the problem in parallel. The Morton key defines when a mesh-free element requires a physical calculation due to change on space. The same mesh-free data management structure is used to solve spatial problems in the SPH model such as the soil analysis presented in Chapter 7.

Chapter 6: Presents the implementation of the SPH algorithm improvements along with different computational and experimental results. Computational errors and solutions are discussed. Different numerical initial conditions and solutions are compared with respect to the computational time and solution accuracy.

Chapter 7: Presents the application of the implemented SPH for fluid-solid interaction to the problem of the simulation of fluid flow through macropore structures with micro level details when the resolution of the CT allows it. Techniques for constructing the boundary walls from CT-scan data are introduced. The necessary computational constraints and initial physical conditions are described. Comparisons

against a mesh method based on a commercial software "COMSOL" are presented. The advantages and disadvantages of the mesh-free methods are emphasised.

Chapter 8: Concludes on the numerical and physical solutions for the pore structures. Current limitations of the implementation and future work are discussed.

1.5 Computational physics

Computational Science implements mathematical models in computer systems to solve scientific problems and to analyse experimental data [9]. Computers are used to explore functions having analytical as well as non-analytical expressions, evaluate numerically complex functions or approximate a continuous solution from a set of discrete points. When applying to solutions and models of problems in physics or experimental data analysis on this field, the discipline receives the name of Computational Physics (CP) [10].

The CP applications originally span from the need of numerical solutions to problems based on known theories and models and the abilities to solve complex numerical representations with the aid of computers. The field grow with the introduction of new computational models base on the analysis experimental data, the numerical simulation of the computational models can then be used to derive new theories, Figure 1.1 shows a diagram of the idea behind the use of CP.

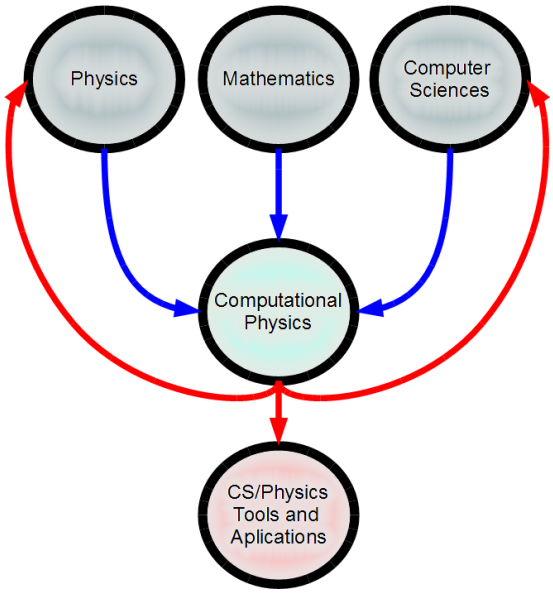


Figure 1.1: **Multidisciplinary structure of CP** - CP is not only the overlapping of different disciplines, but also comprises a set of tools that can be applied in other fields of science and technology.

CP is in most cases defined as the integration of physics, computer sciences (CS) and mathematics having been developed and applied to other fields. As the field expands, CP embraces new methods and techniques from other fields of science and technology such as computer graphics [11, 12], computer animation [13, 14], image processing [15, 16], data mining [17, 18], signal processing [19, 20, 21], etc.

The CP solutions are obtained by advancing the numerical representations of physical objects in space and/or time [22]. The numerical solutions implementation use different programming language. Historically one of the most popular language used on the programming of computational solutions on CP and other areas of sciences and engineering was FORTRAN and up to this date it is one of the most common language for scientific programming [23].

The numerical solutions presented on this thesis are coded using the C++ language. The reason why C++ was chosen due to its easier portability to parallel programming as the original CUDA architecture was only available for the C language.

1.5.1 Computational fluid dynamics

Obtaining a description of a substance's dynamics flow falls into the field of Computational Fluid Dynamics (CFD) [24]. One of its first applications was the study to the supersonic blunt body problem,of the flow behaviour during and after it interacts with the blunt body during supersonic events was studied. Moretti and Abbet [25] proposed a numerical finite-difference solution and a corresponding computational implementation for the problem. The physical aspects of fluid flow simulation follows three physical principles [26]:

1. Mass conservation
2. The Newton's second law: $f = ma$
3. Energy conservation

These principles expressed in terms of a mathematical formulation are know as the Navier-Stokes differential equations [27]. This set of equations may be used to describe fluid behaviour under different conditions. The NavierStokes equations are define any type of single-phase fluid flow. If the fluid viscosity is not taken into account in the Navier-Stokes equations the Euler equations are obtained. If the vorticity terms are removed, the full potential equations are obtained and finally the linearisation of these equations provided with the potential equations [28].

The numerical solution of the Navier-Stokes equations with a computational implementation can be used to solve the flow of behaviour different fluid models, under different boundary conditions and external forces. The CFD can be used to solve space and time dependent flow fields [29, 30, 31].

1.5.2 Computational solid mechanics

As defined in [32]: "Computational Mechanics (CM) is concerned with the use of computational methods to study and simulate physical events governed by the laws of mechanics".

One of the areas of CM is the computational solid mechanics (CSM). CSM studies the model and the numerical analysis of solid materials deformation.

The analysis of the solid material on CSM requires the use of a mathematical model that describes how the physical properties of such material are distributed in the material volume. The model used to define the solid materials presented in this thesis uses the continuous mass approach introduced by the French mathematician Augustin Louis Cauchy [33].

In the continuous mass approach a solid has a smooth distribution of its constitutive material and it can be infinitely divisible into tinier components[34]. The behaviour of solid materials is governed by elasticity, viscoelasticity, and plasticity constitutive equations [35].

Figure 1.2 presents a CSM-base solution applied to lips biomechanical modelling [36].

1.6 Computational simulation

CP problems are solved by numerical simulation which circumvents expensive, time consuming, or difficult to control experiments. Numerical simulation for problems in physics often follows similar steps (Figure

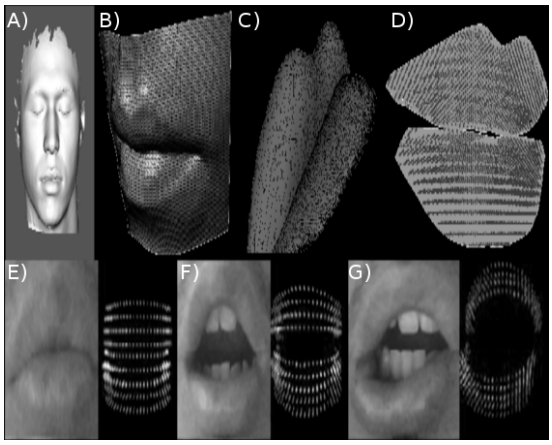


Figure 1.2: **Lips 3D modelling** - A mesh-free method is used to simulated the behaviour of the lips during movement using a 3D SPH model. Figure 1.2 shows the process of obtaining a deformable lip model using a 3D face as model. Data is obtained using a structured light scanner. From left to right: a) 3D face acquired using the Solutionix Rexcan 400 3D scanner; b) corresponding triangle representation of the mouth region surface; c) close-up of the simplified lip mesh; d) mesh-free representation; e,f,g) Picture of the lips on different positions with corresponding mesh-free model.

1.3), starting from observed physical phenomenon to final numerical simulation solution. Different physical phenomenons can be described numerically with the aid of a set of ordinary differential equations (ODE), partial differential equations (PDE), integral equations, or any other mathematical relationships [37] indicating time, space or energy changes.

The process of modelling and simulation can be detailed as follows:

1. Observation of a physical phenomenons. The data used by the simulation and models are build with the aid of experimental data obtained by different means when studied the phenomenons.

2. Mathematical model of the phenomenon using a set of simplifications and assumptions. The numerical descriptions on the computer is build using a set of approximations and simplifications of the phenomenon that are necessary due to either the current computational capabilities or the theoretical description of the phenomenons been studied. Finally the accuracy of the model and simulation can varied depending on the mathematical models used to represent the phenomenon been solved.

3. Problem domain discretisation for numerical solution. The continuum material is represented with a number of discrete components, one of the most popular method is the finite element method [38] that used a mesh based approach to obtain the discrete model. The work presented on this thesis make used of the Smoothing Particle Hydrodynamic (SPH) approach to obtain the numerical solution.

4. Boundary conditions (BC) and initial conditions (IC):

   - BC confines the solution and study of the phenomenon to a volume and defines the type of interaction between the phenomenon of interest and its environment.
   - IC specifies initial stages of the phenomenon behaviour.

5. Computational stage, the actual implementation and the coding of the solution. This stage focuses is on robustness, numerical accuracy, methods to increase the speed of computation and data management.

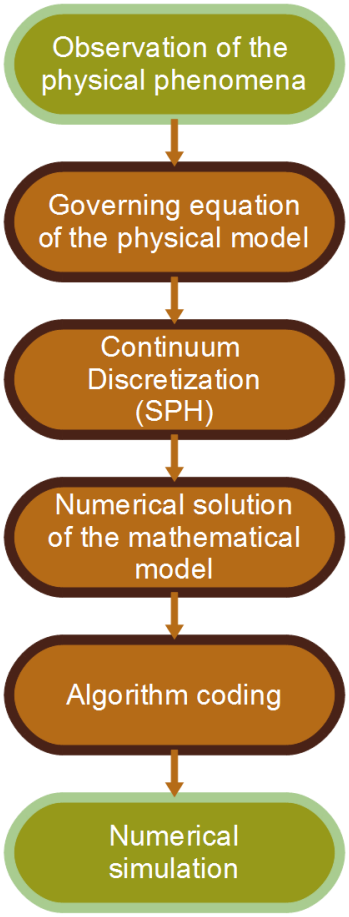


Figure 1.3: **Numerical simulation diagram** - The diagram shows the steps taken by the numerical simulation of physical observations.

Analytical solutions for most of the system of equations describing different physical problems usually have non direct solutions. Different CP techniques are used to find approximate solutions to the spatial or temporal partial differential or integral equations by representing these equations through algebraic representations. These algebraic solutions are used to calculate numeric values of the goal functions at different discrete time and/or space points (Figure 1.4).

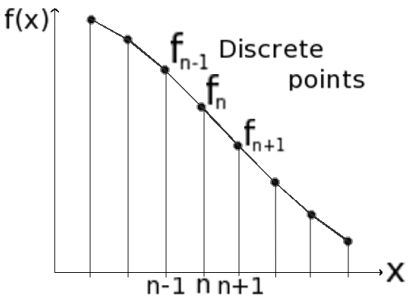


Figure 1.4: **Discretized points of a function** - Uniform discretisation of a one-dimensional function $f(x)$, the points represented the function behaviour at the specific point position

1.7 Mesh-free methods

1.7.1 Mesh methods and their limitations

One of the most common tools for obtaining numerical solutions to physical problems is a mesh-like discretisation of the continuum problem space that can be solved using a mesh based numerical model.

One of the most common numerical mesh-centre methods is the finite element method (FEM) [39]. The continuum is divided into discrete elements, each element or cell having a simple geometrical shape (e.g. tetrahedra). The elements have a connectivity function and the collection of all the elements forms the mesh which provides topological representation of the physical problem [40] [41].

A finite element analysis can be describe with the following steps [42]:

1. Domain discretization. It consists on the mesh generation. The type of geometrical representation the mesh will use is selected and the nodes and their relations (geometric elements forming the mesh) are defined.

2. Interpolation function. When choosing the interpolation method the function should satisfy prescribed conditions and have a solution at a finite number of points. In the case of FEM, the finite number of points are the nodes on the mesh and the conditions are obtained from the problem model. Polynomial interpolation are usually chose to solve FEM problems.

3. Derivation of stiffness matrix per model element. The stiffness matrix is symmetric with positive values. It provide a discrete representation of the stiffness of Hooke's law. Each element on the model (for example each triangle on a tetrahedral mesh) has its own stiffness matrix.

4. Global stiffness matrix. The global matrix defines the geometric and material properties of the whole object been model and it depends on the elements stiffness matrix.

5. Defining boundary conditions. The boundary conditions defined the behaviour between the model and its surroundings.

6. Numerical solution using a FEM method.

The use of a mesh structure has some inherent limitations [43]:

1. Mesh generation: cannot always be achieved by automatic computational techniques. Some problems require the user to invest time and, in some instances, use special computer-aided design tools to build a required mesh structure.

2. Stress accuracy: Stress values obtained through the use of a mesh are often discontinuous at boundaries of the mesh cells due to piecewise nature of the displacement field.

3. Re-meshing: When the problems are not static and I changed on the objects volume and boundaries need to be updated a re-meshing of the structure is needed to maintain the accuracy of the solution over time, adaptive analysis must be performed. Adaptive analysis usually requires re-meshing of the problem space in order to maintain the connectivity. Automatic re-meshing of three-dimensional 3D volumes is still a persistent problem in the mesh-based solutions field [39].

4. Complex mesh deformations: are limited in the analysis of some problems:

    - Large deformations of the mesh cause distortions of the elements (cells), resulting in a loss of accuracy.

- When the need to describe and explore cracks arises, the cracks could propagate along different paths than those defined by the mesh.
- Material breaking into multiple components is a problem due to the nature of the continuum mechanics solution. The mesh elements cannot be broken without redefining and re-meshing the problem.

To escape these drawbacks, the discrete numerical solution has to be represented without a mesh or a proper adaptive mesh system needs to be implemented.

1.7.2 Mesh-free method definition

As defined in [44]:

"A meshfree method is a method used to establish a system of algebraic equations for the whole problem domain without the use of a predefined mesh for the domain discretisation."

The discrete solution is obtained with the aid of a set of nodes scattered on and inside the problem boundary. The nodes do not form a geometric mesh. Prior information on the topological relations between the points is not required, and the solution focus on identifying which nodes belong to the same material.

The density (quantity) of nodes depend on the accuracy required for the solutions, but the definition of this density can vary during the implementation of the numerical analysis. Adaptive methods are easy to implement in mesh-free applications. The initial distribution of the nodes depend only on their necessary density (accuracy) and need not be predefined.

Mesh-free applications should be able to work with any distribution that provides enough nodes per volume, and do not required additional conditions on the initial distribution. Figure 1.5 shows the same 3D spatial volume with different initial representations required for the mesh-centred and mesh-free method. The latter have the following limitations and requirements that must be fulfilled in order to obtain a correct numerical solution:

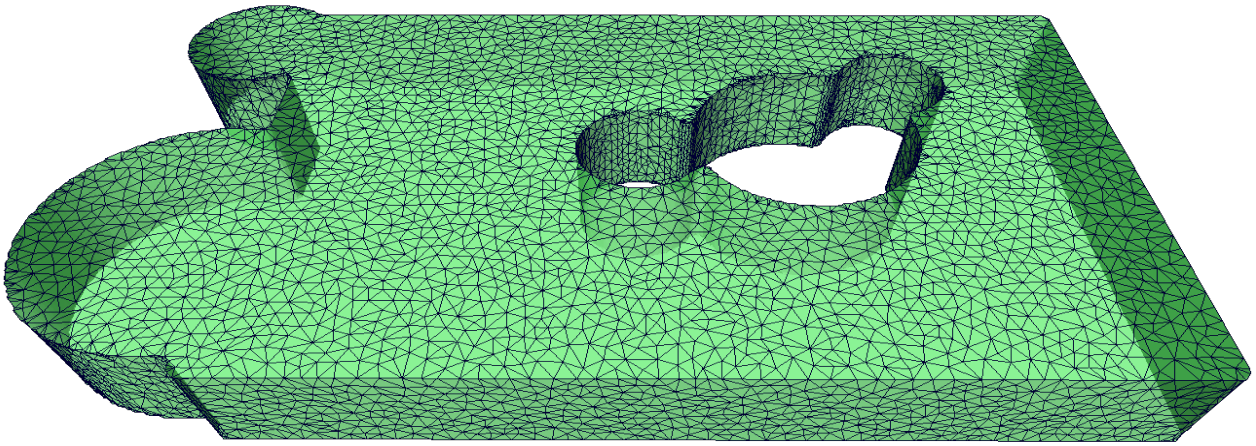
1. Boundary particles: When simulating a computational problem, the number of particles on the frontier is less than the regular number in the solution and this reduces the simulation accuracy. There exist solutions to this problem, for example, the use of so called ghost particles on the boundaries of the problem domain.

2. Neighbourhoods of particles: The number of particles in some areas may decrease causing interpolation problems. To maintain the accuracy the defined action volume of particles has to be updated over time.

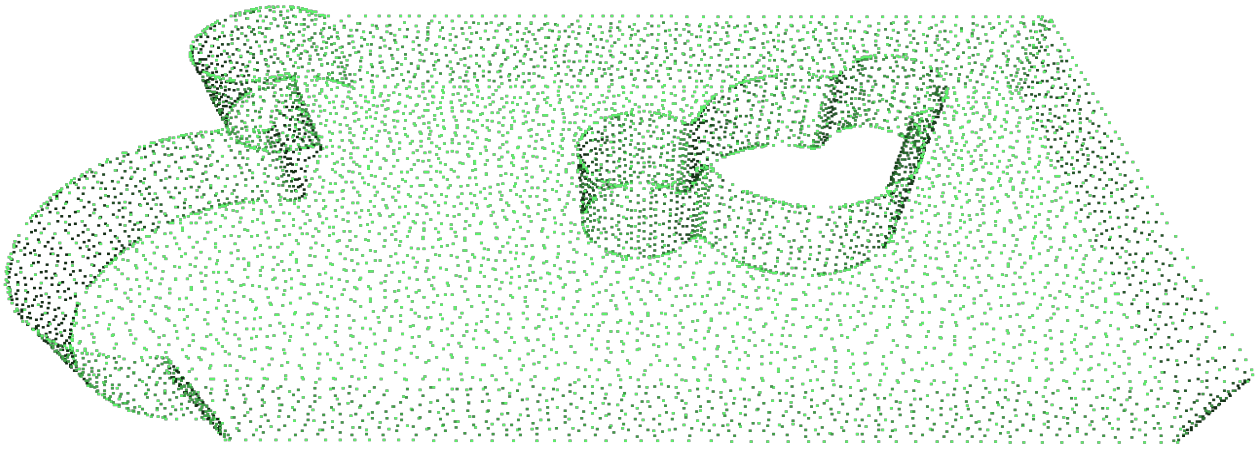
3. Time step: The time steps have to be updated in account with the changing maximum speed of the system. If the time step is too large, then most of the physical values will be inaccurate, while if it is too small the computational time will be too large.

In [45] is presented a table comparing FEM (most representative mesh method) and the approach taken by the mesh methods, on Table 1.2 is shown a summary of the differences.

The shape function provides with the rules of behaviour of the nodes and the links between nodes when solving with FEM or the node relations on the mesh free methods, using the shape function is possible to defined the material behaviour on the problem volume. Different shape functions will produced different element matrices.



(a) Tetrahedral representation for the mesh-centre method



(b) Nodal representation for the mesh-free method

Figure 1.5: **3D volume representations** - The image shows the different volume representation for mesh-centre and a mesh-free method. The mesh-free method does not require any predefined distributions of elements

Differences between FEM and mesh free Methods			
	Items	FEM	Mesh free
1	Mesh geometry elements	Yes	No
2	Mesh generation	Element connectivity property require. In some cases can be a difficult task.	No connectivity is required
3	Adaptive analysis	Difficult for 3D cases	The analysis can always be done
4	Shape function creation	Geometric element based	Node element based
5	Shape function property	Satisfy the Kronecker delta conditions; is valid for all the elements of the same type	Depending on the method may or may not satisfy Kronecker delta conditions, the function is different for each point.
6	Stiffness matrix (discretized element system)	Symmetrical	The symmetry depends on the method used
7	Boundary conditions	Standard method with easy implementation	Special methods may be required depending on the mesh free method
8	Computational speed	Fast computational solution due to optimized implementation	1.1 to 50 times slower (depends on the implementation and method used)
9	Results retrieval	Special techniques are required	Standard routine (node element based)
10	Accuracy	Accurate compared to Finite Different Method	Depending on node resolution can be more accurate than FEM
11	Stage of development	Very well developed	Infancy of development
12	Commercial applications	Many	few available

Table 1.2: Summary of the differences between the FEM and mesh free method implementation as presented in [45].

Chapter 2

SMOOTHING PARTICLE HYDRODYNAMICS (SPH)

2.1 Introduction

This chapter presents a general introduction to the mesh-free SPH method and its use for computational solutions describing the behaviour of material quantities in a physical volume. The mesh-free SPH method was first introduced and described in detail by Gingold and Monaghan in [46].

Advances in the field included the addition of the viscous flow numerical solution [47], the redefinition of the particle resolution on problems with self-gravity [48], and the generalized boundary conditions for fracture propagation problems [49]. These additions to the SPH theory are now considered standard and will be in this chapter formulated.

SPH was used originally to model the behaviour of non-spherical stars, where the model consisted of approximately 80 particles. The SPH method can be used to obtain solutions of a wide range of hydrodynamic problems. [5] discusses several applications and techniques their problems and the basis for constructing one, two, and three dimensional systems and solutions.

SPH provides tools to solve numerical PDEs. By extending the technique presented in [46], the numerical SPH approach has been modified and applied to different types of fluid conditions. One example of using the SPH formulations to define different fluid problems is presented in [50] to simulate a weakly compressible particle system.

Recently the SPH technique has been extended to include solid materials and the interaction between fluids and solids. A technique in [7] simulates the interaction between a solid object and a fluid poured over it.

The fluid objects are solved using the SPH method, the method obtains the particle behaviour of the fluid state, enabling the fluid part of the problem to be solved using a mesh-free problem representation.

For the solids a mesh-centred method is used, the interaction between fluid particles and the solid object is coupled over the triangle mesh representation of the solid.

Similarly in [51] mesh and mesh-free methods are combined to animate burning and melting solids. The theory presented used particle systems without introducing the SPH operator.

in [52] a full mesh free method describes solid deformation using SPH defining the solid - fluid interaction as particle - particle collisions. This method restricts the solid behaviour to elastic deformations.

The SPH method has been applied also in areas outside of CP; Computer Graphics (CG)uses mesh-free methods as a powerful tool for solid-fluid animation. In the CG applications the correctness of the computational solutions is less important, and some errors and simplifications assumptions can be accepted if the application has restrictions such as real-time solutions or user interactivity. In [53], the SPH is used to animate the fluid-solid interaction and animated melting solids.

Biomechanical Modelling (BM) is another area where the mesh-free methods can be implemented in place of mesh-centre methods, particular in [54] a simplified version of SPH is implemented to model solid deformations and simulate the behaviour of human organs (e.g. the liver) under stress. The SPH animates organ behaviour, the solution follows a simple viscoelastic approach for the deformation.

With the extension of the SPH to solid materials, new complementary laws have been implemented to obtain better animations of the behaviour of solids. In [55] a new corotated technique is added to the simulation to allow rotations of rigid solids.

In the field of CM where more rigorous solutions are required, in most cases the use of meshes has been the golden standard. One important area of the mechanics is the study of deformation of solids under stress using the strain fields. A technique to obtain the strain field described in [56] uses undeformed sheets as references for the initial state, and a mechanical system deforms the reference sheets using the object of study. The final strain field is introduced to a mesh system to model the solid deformation.

The strain field deformation in [57] uses a stereo vision system and a SPH computational solution. Here, the SPH obtains mechanical properties of an object under stress.

In the sections below the basics of fluid dynamics and the kinematics of flux described using the Eulerian references frames is discussed first. Then the final equation to solve the fluid dynamics on a Lagrangian references frame using the Eulerian description will be presented.

After the introduction of the necessary basics to describe a quantity on the Lagrangian frame, the solution to this equation using the computational SPH method is presented. Finally, some features of the SPH operation and the smoothing kernel are introduced.

2.2 Fluid dynamics

A fluid does not have a specific shape and deforms continuously under an applied surface force, while resisting external forces acting on individual elements [58]. Fluids are discrete at the microscopic level, and their molecular properties have large fluctuations. The fluid needs to be considered as a continuum when obtaining models to describe its macroscopic behaviour [59]. Macroscopic models of fluids takes the characteristics related to the molecular properties as parameters to the motion equations, one of these parameters being the fluid viscosity.

When analysing the motion of fluids, the use of a discrete volume is useful to define the model [60]. In fluid dynamics the discrete volume is a fluid particle P representing a mass Δm portion of the total fluid volume V_m. The particles P is defined by a small spherical volume ΔV, the size of the fluid particle being small enough to consider the volume uniform. From the microscopic point of view it is sufficiently large in comparison to the intermolecular distances and provides a sufficient number of molecules to average statistical molecular fluctuations. The density ρ of the fluid particle is given by the obvious relationship:

$$\rho = \frac{m}{V} \tag{2.1}$$

The fluid flow takes place in the three-dimensional ($\mathbb{R}^3$) Cartesian coordinate system (x, y, z), the particle position $\mathbf{p}$ having the components $\mathbf{p} = (p_x, p_y, p_z)$. The laws that describe the fluid motion can be sufficiently represented in Euclidean space [61]. When using a hypothetical continuum frame the

components of the material (fluid particles) are distributed continuously. To describe the movement of the discrete volumes of the continuum, a fixed Cartesian coordinate frame is used to represent the motion of all the fluid particles. Figure 2.1 shows spatial locations of two example fluid particles in the Cartesian reference frame.

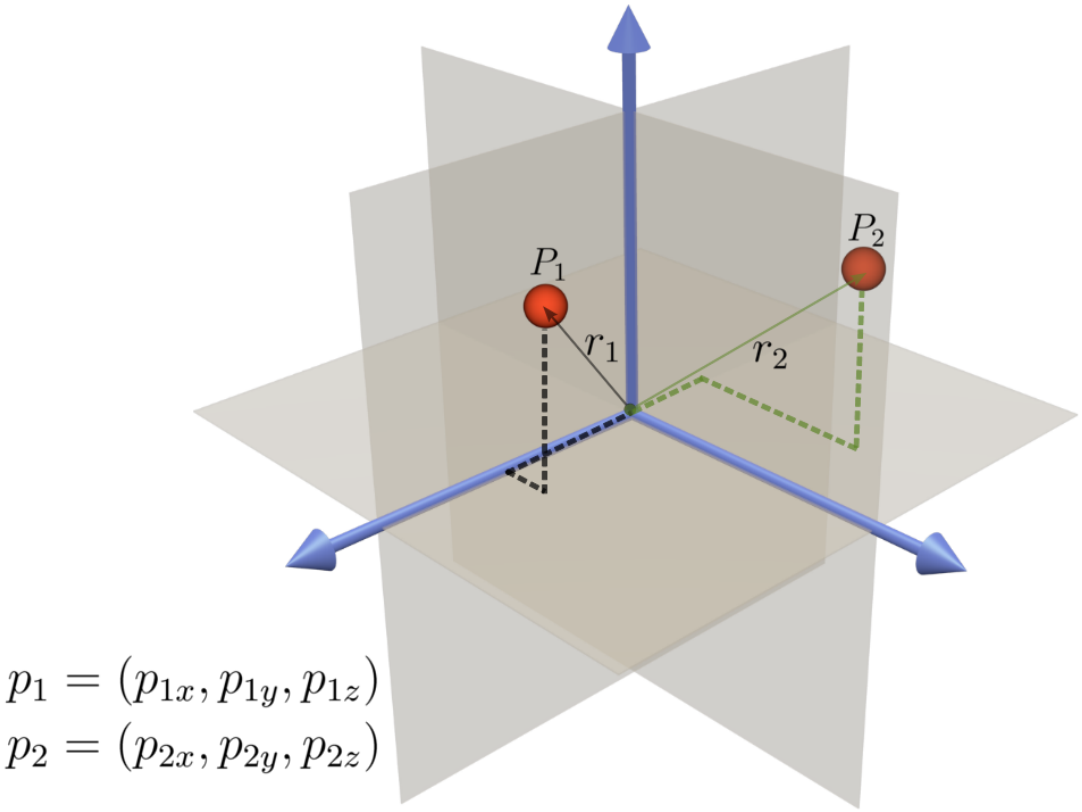


Figure 2.1: **3D coordinate system** - The 3D inertial Cartesian coordinate system is used as a reference frame for the spatial location of a particle. In the figure is shown the position of two particles p_1 and p_2 with respect to the origin defined by the vectors r_1 and r_2

Using the reference frame, the location of a fluid particle can be determined at each time step using a vector $\mathbf{r}$ where $\mathbf{r} = (x, y, z)$. The location of a discrete volume (fluid particle) haves a reference configuration position $\mathbf{R} = (X, Y, Z)$ at the initial reference time t_0.

The laws of fluid motion consider the position of the fluid particle as a function of the original position and time as shown in Equation 2.2.

$$\mathbf{r} = \chi(\mathbf{R}, t) \tag{2.2}$$

The function χ for the vector values of $\mathbf{r}$ at the time $t = t_0$ has to have values equal to $\mathbf{R}$:

$$\mathbf{r}(\mathbf{R}, t_0) = \chi(\mathbf{R}, t_0) = \mathbf{R} \tag{2.3}$$

The flow can be defined using either a Lagrangian description or an Eulerian description:

Eulerian description: The flow properties are obtained at fixed positions in the volume space as the time varies, using a fixed reference frame. The value of a variable A depends on time and space:

$$A(\mathbf{r}, t) \tag{2.4}$$

Lagrangian description: The flow quantities are provided by moving flow particles, any quantity A is defined by its initial conditions and the time t:

$$A(t) \tag{2.5}$$

The velocity $\boldsymbol{\nu}(\mathbf{r}, t)$ of the fluid in the Eulerian frame depends on the time t and the spatial position $\mathbf{r}$ as shown on Equation 2.6

$$\boldsymbol{\nu}(\mathbf{r}, t) = \frac{d\mathbf{r}}{dt} \tag{2.6}$$

The Eulerian acceleration a of the fluid is as follows:

$$\mathbf{a}(\mathbf{r}, t) = \frac{d}{dt}\boldsymbol{\nu}(\mathbf{r}, t) \tag{2.7}$$

2.2.1 Material derivative

When studying the movement of fluid particles, the material derivative [62] is the rate of property change of a fluid particle p. The material derivative is a Lagrangian concept but its definition is constructed using an Eulerian reference frame. Figure 2.2 shows the change in the value of a function $f(\mathbf{r}, t)$ due to the change in space and time.

The material derivative is defined as the Lagrangian time derivative of an Eulerian quantity $f(\mathbf{r}, t)$:

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + \boldsymbol{\nu} \bullet \nabla f \tag{2.8}$$

In a generalised notation the material derivative operation is defined as:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \boldsymbol{\nu} \bullet \nabla \tag{2.9}$$

The equations of the Lagrangian time derivative of any function A depends on the Eulerian reference frame for the gradient, time, and position of the fluid particle p:

$$\frac{DA}{Dt} = f(A, \nabla A, \boldsymbol{r}) \tag{2.10}$$

2.3 Integral interpolation of the Smoothing Particle Hydrodynamics

The value of a quantity A of a fluid particle inside the fluid volume depends on the spatial location with respect to the reference frame. The integral representation of a quantity A with respect to a differential volume $d\boldsymbol{r'}$ is defined in [6]. If the system consists of an infinite number of particles the quantity A can be defined as:

$$A(\boldsymbol{r}) = \int A(\boldsymbol{r'})\delta(\boldsymbol{r} - \boldsymbol{r'})d\boldsymbol{r'} \tag{2.11}$$

Where $\delta(\boldsymbol{r} - \boldsymbol{r'})$ is the Dirac delta function :

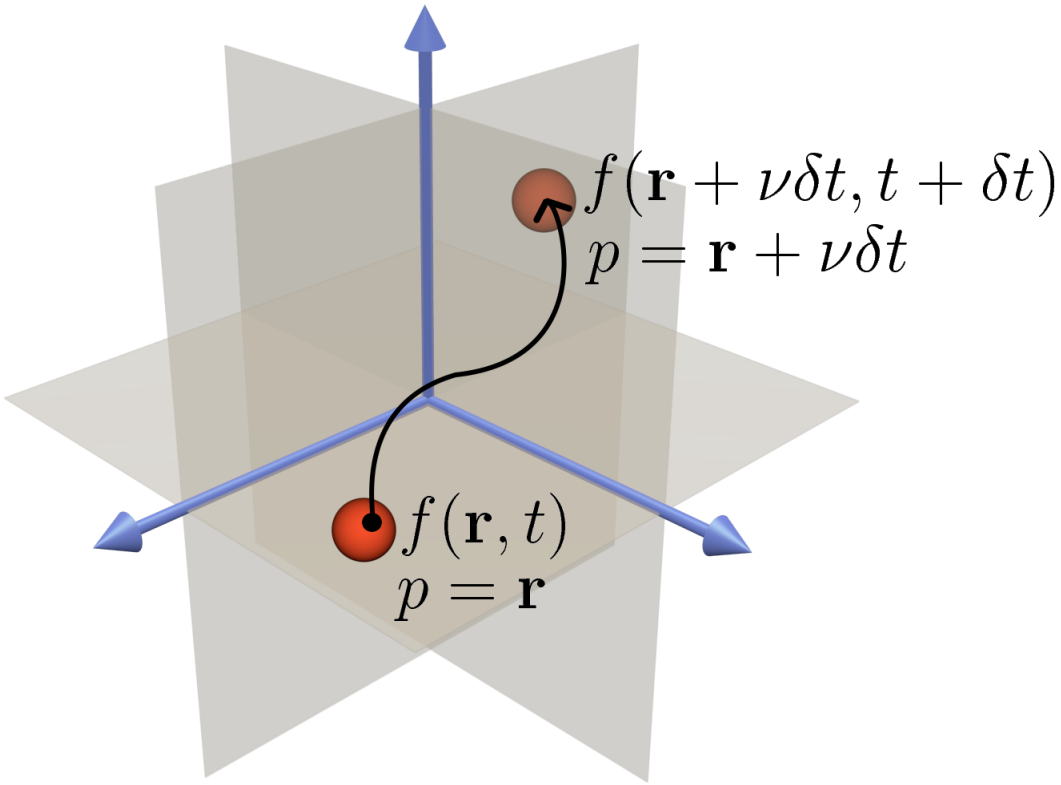


Figure 2.2: **Particle movement on the Eulerian reference frame** - The change in the particle physical quantity value depends on the change in space and time.

$$\delta(\boldsymbol{r}-\boldsymbol{r'}) = \begin{cases} \infty & r = r' \\ 0 & r \neq r' \end{cases} \tag{2.12}$$

Assuming the continuous function $A(\boldsymbol{r})$, the integral representation given by Equation 2.11 is exact. When solving the discrete representation of the function A the quantity value must be obtained by smoothing a finite set of particles.

$$A(\boldsymbol{r}) \doteq \int A(\boldsymbol{r'})W(\boldsymbol{r}-\boldsymbol{r'},h)d\boldsymbol{r'} \tag{2.13}$$

W is the smoothing kernel function, the value of W depends on the distance relation $(\boldsymbol{r}-\boldsymbol{r'})$ and the smoothing length h. The value h defines the influence volume of the smoothing kernel function W. The kernel function W tends to the delta function when h tends to zero.

$$\lim_{h\to 0} W(\boldsymbol{r}-\boldsymbol{r'},h) = \delta(\boldsymbol{r}-\boldsymbol{r'}) \tag{2.14}$$

The smoothing kernel functions are normalised to 1 so the constant values are interpolated exactly [63], as shown in Equation 2.15.

$$\int W(\boldsymbol{r}-\boldsymbol{r'},h)d\boldsymbol{r'} = 1 \tag{2.15}$$

The kernel W should also follow the compact condition. This condition handles the effect from particles with positions outside the volume of effect defined by h.

$$W(\boldsymbol{r}-\boldsymbol{r'},h)=0 \text{ when } \left|x-x'\right|>kh \tag{2.16}$$

k is a constant that together with the smoothing length defines the effective non-zero volume of W.

In [6] the operations needed to approximate the integral using a summation over mass elements are explained. The fluid volume is divided into small particles elements i. Each particle has a mass m_i, density ρ_i and position r_i. The interpolation integral value of the quantity A at the position of the particle i, A_i is given by:

$$\int \frac{A(\boldsymbol{r'})}{\rho(\boldsymbol{r'})}\rho(\boldsymbol{r'})d\boldsymbol{r'} \tag{2.17}$$

Where $\rho(\boldsymbol{r'})d\boldsymbol{r'}$ is the element mass.

2.4 Smoothing Particle Hydrodynamics particle approximation summation

In Equation 2.13 the function $A(\boldsymbol{r'})$ value is integrated over the infinitesimal volume $d\boldsymbol{r'}$. For the numerical approximation summation the mass elements are defined as [6]:

$$m=\rho d\boldsymbol{r'} \tag{2.18}$$

The mass element m is represented by a set of particles i. The total number of particles N forms the particles system of the problem. The mass element of each particle can then be defined as:

$$m_i=\Delta V_i\rho_i \tag{2.19}$$

Where ΔV_i and ρ_i are the individual volume and density of the particles and $i=(1,2,...,N)$.

As described in the last section, the total fluid volume is represented by a set of particles. The continuous integral representation (Equation 2.13) is converted to a discrete summation over a set of particles j inside the support domain of a focus particle i (e.g. see Figure 2.3).

The smoothing function W, is used to approximate the continuum function. W determines if and how a particle j will affect the focus particle i. Figure 2.4 shows a schematic representation of the particles interactions.

The summation approximation of the integral solution for $A(\boldsymbol{r})$ [64] is given by Equation 2.20:

$$\langle A_S(\boldsymbol{r}_i)\rangle=\sum_{j=1}^{N}\frac{m_j}{\rho_j}A(\boldsymbol{r}_j)W_{ij} \tag{2.20}$$

Where the operation symbol $\langle\rangle$ is used to defined an SPH operation over the function $A_S(\boldsymbol{r}_i)$ and W_{ij} is defined as:

$$W_{ij}=W(\boldsymbol{r_i}-\boldsymbol{r_j},h) \tag{2.21}$$

The value of the quantity A_i is approximated using an average over all the particles j that belong to the support domain of i. The approximation for the first spatial derivative is given by:

$$\frac{\partial A_s}{\partial x}=\langle\nabla\cdot A(\boldsymbol{r}_i)\rangle=\sum_{j=1}^{N}\frac{m_j}{\rho_j}A(\boldsymbol{r}_j)\cdot\nabla_i W_{ij} \tag{2.22}$$

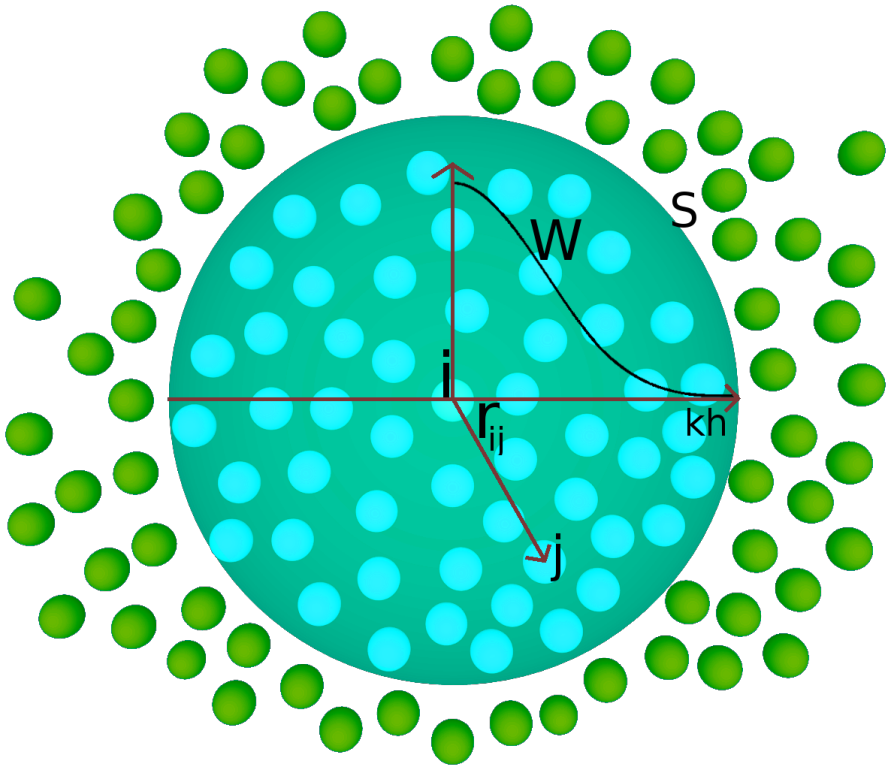


Figure 2.3: **Particle summation approximation** - The value of a quantity A_i with position defined by the particle i using a set of particles j that belongs to the support domain of W_i, is defined by the summation of all the particles j. The support domain is defined by the quantity kh. The particles outside the interaction volume of particle i will not have a direct effect on particle i.

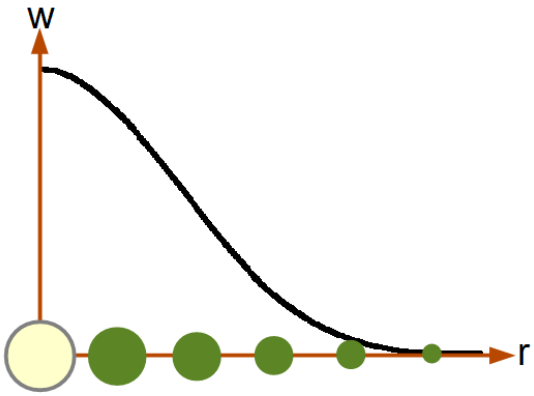


Figure 2.4: **Particles smoothing function** - The value of W is related to the distances between the focus particle i and each j particle on its vicinity. The figure shows a schematic representation of the interaction, where the area of each element is use to represent the decrease on the W factor as the distances increases.

In the literature there are different definitions for the smoothing function W, different smoothing functions are used in the search for the most efficient approximation to the continuum function A. This thesis uses the B-spline function defined in [4] as the smoothing kernel function (Equation 2.23),

$$W(R,h) = \alpha_{dim} \begin{cases} \frac{2}{3} - R^2 + \frac{1}{2}R^3 & 0 \leq R < 1 \\ \frac{1}{6}(2-R)^3 & 1 \leq R < 2 \\ 0 & R \geq 2 \end{cases} \tag{2.23}$$

Where the constant α_{dim} depends on the dimension of the problem (for the three-dimensional problems $\alpha_{dim} = \frac{3}{2\pi h^3}$) and R is the relative distance between two particles:

$$R = \frac{|r_i - r_j|}{h} \tag{2.24}$$

Figure 2.5 shows the behaviour of the B-spline kernel function and its derivative.

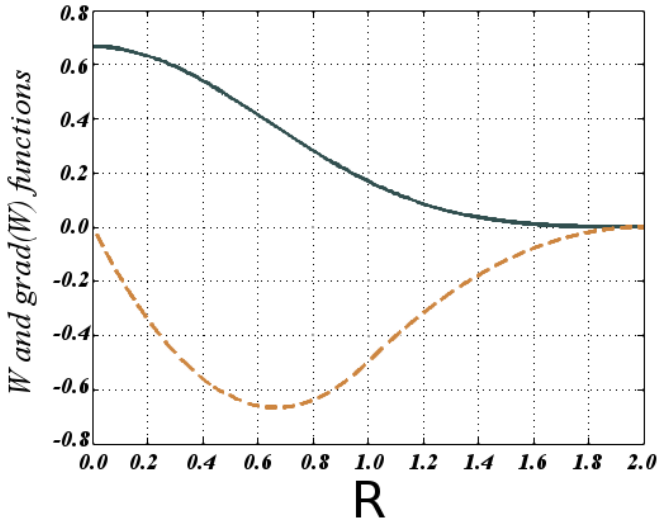


Figure 2.5: **Smoothing kernel function** - In the SPH implementation the B-spline smoothing kernel function is used, the figure shows the B-spline function on black and its derivative in orange.

A technique to obtain the PDEs of the SPH formulations presented in [63] replaces the derivative by the derivative of the approximation function. As described in [63] this type of derivative will not vanish if A is a constant. The two identities used in the literature to form the derivative introduce the density inside the gradient and use a relation between the values of the quantity A for different particles i and j.

$$\nabla \cdot A(\boldsymbol{r}) = \frac{1}{\rho}\left(\nabla \cdot (\rho A(\boldsymbol{r})) - A(\boldsymbol{r}) \cdot \nabla \rho\right) \tag{2.25}$$

$$\nabla \cdot A(\boldsymbol{r}) = \rho\left(\nabla \cdot \left(\frac{A(\boldsymbol{r})}{\rho}\right) + \left(\frac{A(\boldsymbol{r})}{\rho^2}\right) \cdot \nabla \rho\right) \tag{2.26}$$

Applying Equation 2.25 and Equation 2.26 to the particle system the gradient of a quantity A_i of a particle i is defined as:

$$\nabla \cdot A(\boldsymbol{r}_i) = \frac{1}{\rho_i}\left(\sum_{j=1}^{N} m_j \left(A(\boldsymbol{r}_j) - A(\boldsymbol{r}_i)\right) \cdot \nabla_i W_{ij}\right) \tag{2.27}$$

$$\nabla \cdot A(\boldsymbol{r}_i) = \rho_i \left(\sum_{j=1}^{N} m_j \left(\left(\frac{A(\boldsymbol{r}_j)}{\rho_j^2} \right) + \left(\frac{A(\boldsymbol{r}_i)}{\rho_i^2} \right) \right) \cdot \nabla_i W_{ij} \right) \tag{2.28}$$

Besides these two identities the SPH operation on quantities follow the rules:

$$\langle A_1 + A_2 \rangle = \langle A_1 \rangle + \langle A_2 \rangle \tag{2.29}$$

$$\langle A_1 A_2 \rangle = \langle A_1 \rangle \langle A_2 \rangle \tag{2.30}$$

If a constant is operating over a quantity A the constant can be taken out from the operand:

$$\langle cA_1 \rangle = c \langle A_1 \rangle \tag{2.31}$$

The SPH operator is commutative:

$$\langle A_1 + A_2 \rangle = \langle A_2 + A_1 \rangle \tag{2.32}$$

$$\langle A_1 A_2 \rangle = \langle A_2 A_1 \rangle \tag{2.33}$$

2.5 Conclusions

The introduction of a mesh free numerical method based on the distribution of volume elements (particles) allows for a representation of the fluid dynamics equations using the approximation operand $\langle\rangle$ of the SPH method.

SPH uses the Lagrangian definition of the laws that describe fluid motion, the quantities were obtained with the use of a particle system. Each particle in the system depends on the position inside the volume with respect to the reference frame and the current time step, The simulation obtains the smoothed average value per particle in the volume during a period of time.

The introduction of the operand $\langle\rangle$ and its derivative provides a tool to approximate integral equation and PDEs through the use of the summatory of the individual values of the particles inside the support domain of the focus particle. The effect of the neighbour particles over the focus particle depends on the distances between particles and the smoothing kernel function.

Chapter 3

SMOOTHING PARTICLE HYDRODYNAMICS IMPLEMENTATION ON FLUID FLOW

3.1 Introduction

The SPH implementation can be used to solve different types of fluid flows and flow interactions. The simplest problems on fluid flow only needs to solved the PDEs describing ideal fluids, without the consideration of viscosity or shear stress and using a simplified isotropic normal stress.

Fluid viscosity is the internal friction caused when the fluid under the effect of a force undergoes a shape change [65][66]. All fluids are subject to viscosity. The fluid friction can be expressed as a force that causes a resistance to the flow. In Section 3.3.3 a description of the relation between the viscosity and the shear tensor is presented.

The basic Eulerian approach to the description of fluid flow supposes ideal fluid conditions, which neglect shear stress and isotropic normal stress. With these conditions the pressure p depends only on the fluid velocity ν and its rate of change [67].

When fluid viscosity is taken into account it is necessary to obtain the formulation that relates the gradient of the velocity $\nabla\nu$ and the stress. The mathematical formulation of the flow of fluids with viscosity can be obtained by applying the Navier-Stokes equations [68].

The inclusion of viscosity is one of the most important parts of the simulation since all fluids have viscosity and the solution of the problems presented in this thesis requires the use of non ideal fluids.

3.2 Fluid characteristics

Depending on the focus of the simulation different types of fluid definitions will be used. The SPH implementation presented in this thesis makes use of a modular approach where depending on the requirements, different modules (each module is in charge of solving a physical property) can be activated.

Another fluid property described by the model is fluid compressibility. Compressibility determines the capacity of the fluid density to vary significantly when there is a pressure change. The study of compressible flow necessitates the introduction of the conservation of energy principle to the simulation. The formulation will be discussed in Section 3.3.4.

In the SPH implementation presented in this thesis the following type of fluids can be solved: ideal fluids (fluid without viscosity or compressibility), perfect fluids (fluid with compressibility and without viscosity) and Newtonian fluids.

In a Newtonian fluid the stress is proportional to the time rate of changed of the strain; the proportionality factor is the viscosity coefficient. In Section 3.3.3 is discussed the relation between the fluid momentum and the stress.

3.3 Viscous fluids dynamics: Navier-Stokes equation

The solution of the flow of viscous fluids is given by the governing equations of the laws of conservation [69]:

1 Conservation of mass

2 Conservation of momentum

3 Conservation of energy

As seen in Section 2.2, the SPH solution is based on the Lagrangian references frame. The mathematical representation of the equations in the Lagrangian frame is obtained using the control volume descriptor, discussed on Section 3.3.1.

3.3.1 Control volume

The control volume V is an arbitrary mathematical description of a cut in space of the total fluid volume [68], the cut allows for the obtention of a small component of the whole problem, the control volume is used to obtain the mathematical relations of the governing equation of the fluid continuous medium inside of it, instead of trying to defined the whole fluid volume problem. The surfaces surrounding the defined control volume receives the name of the control surface S and it is used as the frontier of the control volume on the mathematical description of the governing equation.

In the Lagrangian references frame the control volume moves along with the rest of the fluid, Figure 3.1, the defined control volume V changed its position in space.

In the Lagrangian scheme the movement of the control volume V changes the fluid properties inside of it due to the changed on the surroundings of V. A change in the control surface S (control volume deformation) will result in a change in V and a new control volume definition is obtained.

Figure 3.2 shows a differential volume ΔV inside the original control volume. The definition of ΔV changes due to the movement of the differential control surface dS over a time interval Δt, ΔV is given by Equation 3.1

$$\Delta V = \nu \Delta t \cdot \mathbf{n} dS \tag{3.1}$$

Where ν is the differential control volume velocity and $\mathbf{n}$ is the normal vector to the velocity.

The total volume change of the entire control volume is then the integration over the control surface S

$$\Delta V = \int_S \nu \Delta t \cdot \mathbf{n} dS \tag{3.2}$$

The rate of flow Q through S is defined as the change of volume with respect to time $\Delta V / \Delta t$:

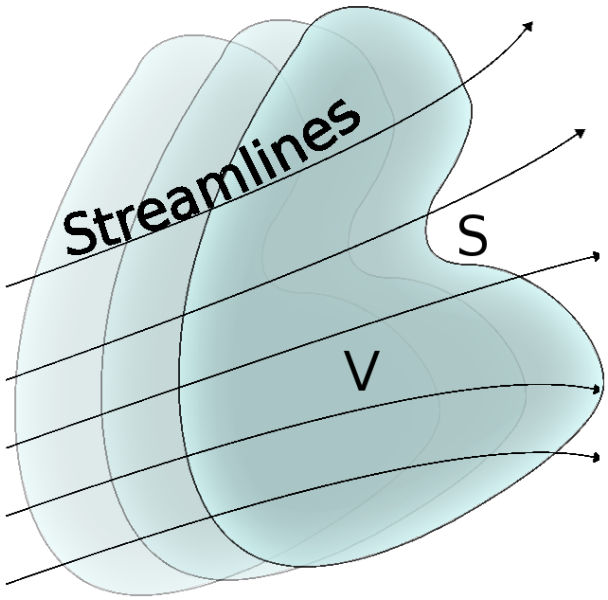


Figure 3.1: **Lagrangian control volume** - In the Lagrangian frame of reference the control volume moves with the fluid and changes its position in space. V is the control volume and S is the control surface surrounding the volume.

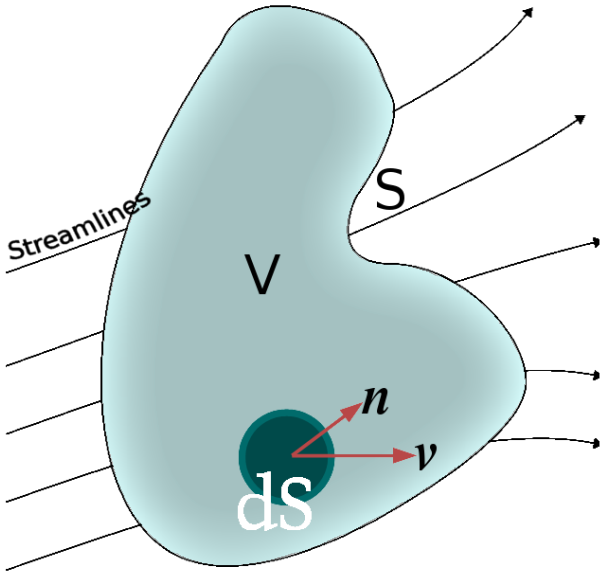


Figure 3.2: **Time differential Lagrangian control volume** - The change of dS produces a Volume ΔV change in the Lagrangian infinitesimal volume cell inside the original control volume.

$$Q = \Delta V/\Delta t = \int_V \nabla \cdot \nu dV \tag{3.3}$$

To obtain a better description of the fluid volume inside of the control volume, an infinitesimal control volume δV with volume value equal to $\delta V = dx * dy * dz$ is defined. The fluid volume inside of the infinitesimal control volume is considered to be homogeneous and all the molecules that form the infinitesimal volume have equal values on all its physical properties. The new infinitesimal rate of flow δQ is defined as

$$\delta Q = \Delta\left(\delta V\right)/\Delta t = \left(\nabla \cdot \nu\right) \int_V d\left(\delta V\right) = \left(\nabla \cdot \nu\right)\left(\delta V\right) \tag{3.4}$$

Finally the differential rate of flow is defined as:

$$D\left(\delta V\right)/Dt = \left(\nabla \cdot \nu\right)\left(\delta V\right) \tag{3.5}$$

The velocity divergence that defines the time rate of volume change per unit volume is given by

$$\nabla \cdot \nu = \frac{1}{\delta V}\frac{D\left(\delta V\right)}{Dt} \tag{3.6}$$

The velocity divergence is used to describe the control volume physical properties changes over time.

3.3.2 Continuity equation

The accuracy of the solution depends on how the continuum is represented when using discrete elements. The conservation of mass needs to be maintained on the discrete solution. In the Lagrangian representation the infinitesimal control volumes (infinitesimal fluid cell) are used as the elements where the mass of the continuum is represented.

The fluid cells are defined by their: volume δV, mass m and density ρ. Each fluid cell follows the conservation of mass definition. The amount of mass that fluxes into the cell is equal to the amount that fluxes out of the cell [67], so the change on volume produces a change on the cell density that can be expressed with the aid of the material derivative (Section 2.2.1) as:

$$\frac{D\rho}{Dt} = -\rho\frac{1}{\delta V}\frac{D\left(\delta V\right)}{Dt} = -\rho\nabla \cdot \nu \tag{3.7}$$

Where Equation 3.7 follows the conservation of mass on each cell.

3.3.3 Momentum equation

In this section a brief description of the forces acting on the volume element will be presented (for a more in-depth description see [26]). The momentum equation is derived from Newton's second law of continuum mechanics (conservation of momentum):

$$\mathbf{f} = m\mathbf{a} \tag{3.8}$$

Where $\mathbf{f}$ is the resulting force, m is the object mass and $\mathbf{a}$ its acceleration. The infinitesimal fluid cell has a position defined by $\mathbf{x} = (x, y, z)$ and an acceleration given by $\mathbf{a} = (\frac{D\nu_x}{Dt}, \frac{D\nu_y}{Dt}, \frac{D\nu_z}{Dt})$, where ν is the cell velocity. Figure 3.3 shows the forces in the x direction over a fluid element (fluid cell).

The total net forces (resultant force) over the fluid elements is the result of the addition of the body forces and the surface forces.

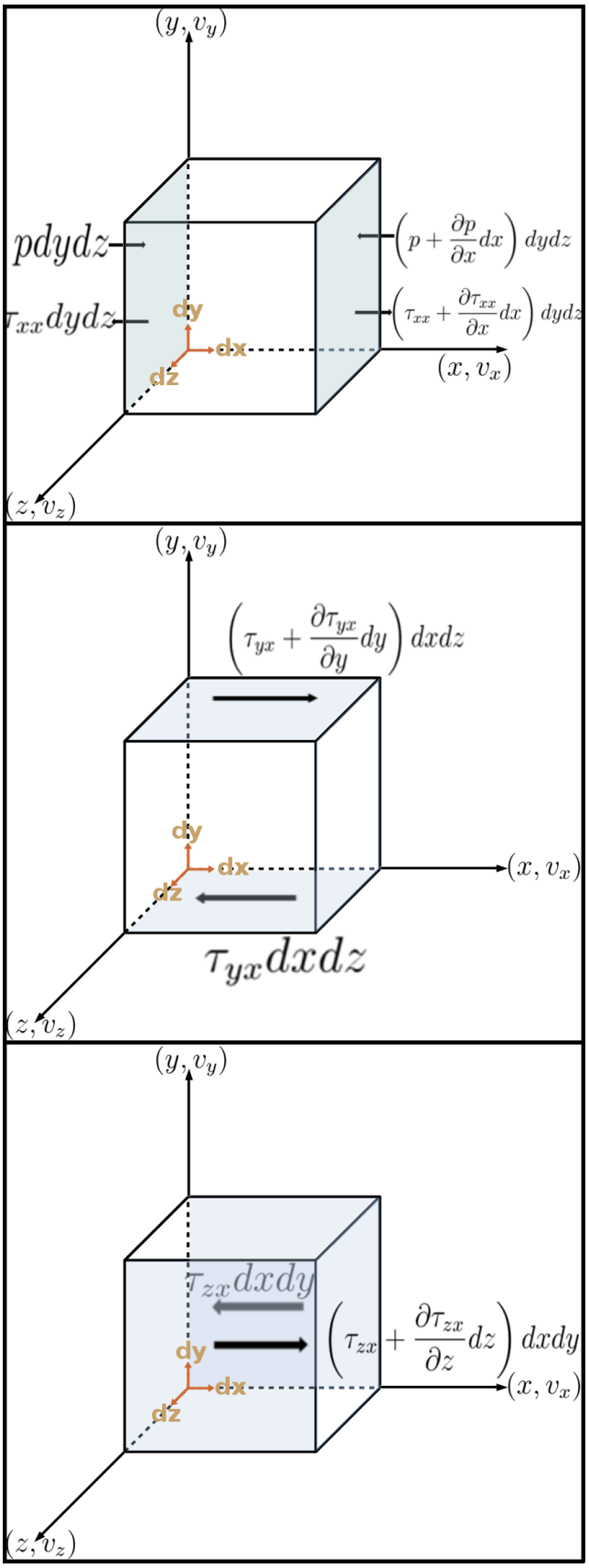


Figure 3.3: **Forces over a fluid cell** - The figure shows the x direction forces over a Lagrangian infinitesimal fluid cell. The stress τ_{ij} uses a convention where the index i denotes the perpendicular plane the force has been applied to and j is the direction of the force.

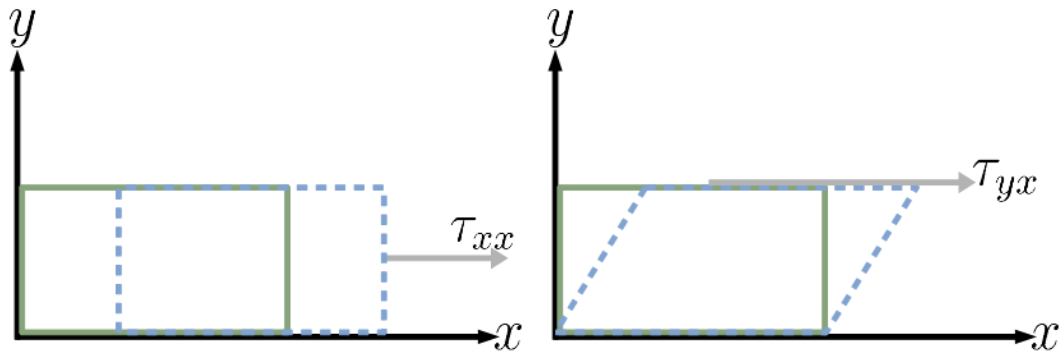
1. ***Body forces:*** These forces act over the complete fluid element (the control volumetric mass). They include: gravitational, electric, and magnetic forces.

2. ***Surface forces:*** These forces act on the control surface directly, the sources of the forces are:

   (a) Pressure force generated on the surface due to the surrounding fluid elements.

   (b) Viscous force, these forces are the shear and the normal stress. The force is also imposed by the tubing (force around the control surface) and pushing of the control surface due to friction.

The body forces on a fluid element will be defined as $\rho\mathbf{f}(dxdydz)$. The shear and normal stresses τ are related to the rate of change of deformation of the fluid element. In Figure 3.4 a simplified 2D representation of the normal and the shear stresses is shown.



(a) Normal stress (b) Shear stress

Figure 3.4: **Normal and shear stresses components** - The figure shows the 2D representation of the plane xy showing the stress effect over a volume element. a) shows the normal stress τ_{xx}. b) shows the shear stress τ_{yx}

The shear and normal stresses are related to the velocity gradient of the flow. One common type of normal stress is the pressure that acts as a compressive stress. In the case of the shear stress the effect is a change in the volume geometry as presented in Figure 3.4b.

Using Figure 3.3 the net surface force in the x direction fs_x acting on the fluid cell can been calculated using Equation 3.9:

$$\begin{aligned} fs_x = &\left[p - \left(p + \frac{\partial p}{\partial x}dx\right)\right] dydz \\ &+ \left[\left(\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x}dx\right) - \tau_{xx}\right] dydz \\ &+ \left[\left(\tau_{yx} + \frac{\partial \tau_{yx}}{\partial y}dy\right) - \tau_{yx}\right] dxdz \\ &+ \left[\left(\tau_{zx} + \frac{\partial \tau_{zx}}{\partial z}dz\right) - \tau_{zx}\right] dxdy \end{aligned} \tag{3.9}$$

Reducing and regrouping the terms and adding the body force to the surfaces forces gives the final expression for the net force on the x direction F_x presented on Equation 3.10

$$F_x = \left(-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z}\right) dxdydz + \rho fs_x dxdydz \tag{3.10}$$

Rewriting Newton's second law using the net force is expressed on Equation 3.11.

$$m\frac{d\nu_x}{dt} = \rho dxdydz\frac{d\nu_x}{dt} \quad = \left(-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z}\right) dxdydz + \rho f_x dxdydz \tag{3.11}$$

Finally the momentum equations $\frac{D\nu}{Dt}$ on the x y and z directions are

$$\begin{aligned}
\rho\frac{D\nu_x}{Dt} &= -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x \\
\rho\frac{D\nu_y}{Dt} &= -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y \\
\rho\frac{D\nu_z}{Dt} &= -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z
\end{aligned} \tag{3.12}$$

In Newtonian fluids the strain ϵ and stress τ are proportional

$$\tau^{\alpha\beta} = \mu\epsilon^{\alpha\beta} \tag{3.13}$$

Where μ is the viscosity coefficient of the fluid, and the strain ϵ is

$$\epsilon^{\alpha\beta} = \frac{\partial \nu^\beta}{\partial x^\alpha} + \frac{\partial \nu^\alpha}{\partial x^\beta} - \frac{2}{3}\left(\nabla \cdot \nu\right)\delta^{\alpha\beta} \tag{3.14}$$

Where α and β follow tensor notation and represent the different tensor component of the three dimensional spaces directions (x y z); and $\delta^{\alpha\beta}$ is the Kronecker delta:

$$\delta^{\alpha\beta} = \begin{cases} 1 & \text{if } \alpha = \beta \\ 0 & \text{if } \alpha \neq \beta \end{cases} \tag{3.15}$$

3.3.4 Energy equation

The fluid cell conservation of energy follow the first law of thermodynamics, the law of energy conservation. Using the fluid cell element, the law can be defined as [70, 69]:

$$\left\{\begin{array}{c}\text{Rate of energy}\\ \text{change inside the}\\ \text{fluid cell element}\end{array}\right\} = \left\{\begin{array}{c}\text{Net heat}\\ \text{flux into the}\\ \text{fluid cell}\end{array}\right\} + \left\{\begin{array}{c}\text{Time rate of work done}\\ \text{by body and surface forces}\\ \text{on the fluid cell element}\end{array}\right\} \tag{3.16}$$

In [5] the energy equation for the SPH approach is defined by separating the energy flux and the body forces (body forces affect the complete cell and will be integrated into the final acceleration). The energy is defined by the surface forces and the energy dissipation. The rate of energy change $\rho\frac{De}{Dt}$ equation is

$$\begin{aligned}
\rho\frac{De}{Dt} = &\left(\frac{\partial \nu_x}{\partial x} + \frac{\partial \nu_y}{\partial y} + \frac{\partial \nu_z}{\partial z}\right) \\
&+ \tau_{xx}\frac{\partial \nu_x}{\partial x} + \tau_{yx}\frac{\partial \nu_x}{\partial y} + \tau_{zx}\frac{\partial \nu_x}{\partial z} \\
&+ \tau_{xy}\frac{\partial \nu_y}{\partial x} + \tau_{yy}\frac{\partial \nu_y}{\partial y} + \tau_{zy}\frac{\partial \nu_y}{\partial z} \\
&+ \tau_{xz}\frac{\partial \nu_z}{\partial x} + \tau_{yz}\frac{\partial \nu_z}{\partial y} + \tau_{zz}\frac{\partial \nu_z}{\partial z}
\end{aligned} \tag{3.17}$$

3.3.5 Navier-Stokes equations

The Navier-Stokes equations define the set of differential equations in the Lagrangian description that provides the governing equations for fluid dynamics.

The previously introduced conservation equations can be defined with the aid of tensor notation and the use of the total stress tensor σ. The tensor is defined by the isotropic pressure P and the viscous stress τ.

$$\sigma^{\alpha\beta} = -P\delta^{\alpha\beta} + \tau^{\alpha\beta} \tag{3.18}$$

Where α and β follow tensor notation and represent the different tensor component of the three dimensional spaces directions (x y z). The conservation equations can be defined as:

1. ***Continuity equation:***
$$\frac{D\rho}{Dt} = -\rho\frac{\partial \nu^{\beta}}{\partial x^{\beta}} \tag{3.19}$$
2. ***Momentum equation****(Excluding the external forces):*
$$\frac{D\nu^{\alpha}}{Dt} = \frac{1}{\rho}\frac{\partial \sigma^{\alpha\beta}}{\partial x^{\beta}} \tag{3.20}$$
3. ***Energy equation:***
$$\frac{De}{Dt} = \frac{\sigma^{\alpha\beta}}{\rho}\frac{\partial \nu^{\alpha}}{\partial x^{\beta}} \tag{3.21}$$

3.4 Smoothing Particle Hydrodynamics solution for the Navier-Stokes equation

Using the definitions presented in Section 3.3 the SPH formulation to obtain the computational solutions of the equations can be defined.

The solution to the Navier-Stokes equations is obtained using the numerical method presented in Section 2

The SPH formulation in the literature has different definitions depending on the date of publication and the method that is going to be used to solve the problem been studied. The formulations presented in this section are based on [6] and [5]

3.4.1 Smoothing Particle Hydrodynamics approximation for density

SPH uses the density as the variable in the volume, mass and density relation, $\rho = m/V$. The density value per particle i, ρ_i, determines the smoothing length of the particles and their distribution. In this work we use the density summation method proposed in [71] to calculate the SPH density,

$$\rho_i = \frac{\sum_{j=1}^{N} m_j W_{ij}}{\sum_{j=1}^{N} \frac{m_j}{\rho_j} W_{ij}} \tag{3.22}$$

Where i is the collection of all particles $i = (1, 2, , N)$ in the system, and the summation is solved over all the particles j where $j \neq i$. m_j is the mass of particle j and W_{ij} is the smoothing kernel functions defined on Equation 2.16

The normalize representation on equation 3.22 is used to get a better approximation of the solution near the boundaries and material interfaces.

3.4.2 Smoothing Particle Hydrodynamics approximation of momentum

Using the SPH approximation presented in Equation 2.22 and the Navier-Stokes definition for the Equation of momentum 3.20.

$$\frac{D\nu_i^\alpha}{Dt} = \frac{1}{\rho_i}\sum_{j=1}^{N}\frac{\sigma_j^{\alpha\beta}}{\rho_j}\frac{\partial W_{ij}}{\partial x_i^\beta} \tag{3.23}$$

Where ν_i^α is the vector velocity of the particle i and $\sigma_i^{\alpha\beta}$ is its stress tensor.

In SPH formulation it is common to put the density value inside the summation. In the momentum equation the following identity is used to achieve this

$$\sum_{j=1}^{N} m_j \frac{\sigma_i^{\alpha\beta}}{\rho_i\rho_j}\frac{\partial W_{ij}}{\partial x_i^\beta} = \frac{\sigma_i^{\alpha\beta}}{\rho_i}\left[\sum_{j=1}^{N}\frac{m_j}{\rho_j}\frac{\partial W_{ij}}{\partial x_i^\beta}\right] \tag{3.24}$$

Using Equation 3.24 the momentum Equation 3.23 can be defined as:

$$\frac{D\nu_i^\alpha}{Dt} = \sum_{j=1}^{N} m_j \frac{\sigma_i^{\alpha\beta}+\sigma_j^{\alpha\beta}}{\rho_i\rho_j}\frac{\partial W_{ij}}{\partial x_i^\beta} \tag{3.25}$$

Another common representation for momentum is

$$\frac{D\nu_i^\alpha}{Dt} = \sum_{j=1}^{N} m_j \left(\frac{\sigma_i^{\alpha\beta}}{\rho_i^2}+\frac{\sigma_j^{\alpha\beta}}{\rho_j^2}\right)\frac{\partial W_{ij}}{\partial x_i^\beta} \tag{3.26}$$

Separating the stress tensor σ_i on its isotropic pressure P_i and the viscous stress (Equation 3.18), and using the particles strain $\epsilon_i^{\alpha\beta}$ and the viscous coefficient μ_i , equ.3.25 and Equation 3.26 can be defined as:

$$\frac{D\nu_i^\alpha}{Dt} = -\sum_{j=1}^{N} m_j \frac{P_i+P_j}{\rho_i\rho_j}\frac{\partial W_{ij}}{\partial x_i^\alpha} + \sum_{j=1}^{N} m_j \frac{\mu_i\epsilon_i^{\alpha\beta}+\mu_j\epsilon_j^{\alpha\beta}}{\rho_i\rho_j}\frac{\partial W_{ij}}{\partial x_i^\beta} \tag{3.27}$$

$$\frac{D\nu_i^\alpha}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}\right)\frac{\partial W_{ij}}{\partial x_i^\alpha}\left(\frac{\mu_i\epsilon_i^{\alpha\beta}}{\rho_i^2}+\frac{\mu_j\epsilon_j^{\alpha\beta}}{\rho_j^2}\right)\frac{\partial W_{ij}}{\partial x_i^\beta} \tag{3.28}$$

Where the first terms of the equations are the SPH approximation for the isotropic pressure and the second parts are the SPH solution for the viscous forces. The particle strain $\epsilon_i^{\alpha\beta}$ tensor is given by:

$$\epsilon_i^{\alpha\beta} = \sum_{j=1}^{N}\frac{m_j}{\rho_j}\nu_{ji}^\beta\frac{\partial W_{ij}}{\partial x_i^\alpha} + \sum_{j=1}^{N}\frac{m_j}{\rho_j}\nu_{ji}^\alpha\frac{\partial W_{ij}}{\partial x_i^\alpha} - \left(\frac{2}{3}\sum_{j=1}^{N}\frac{m_j}{\rho_j}\nu_{ji}\cdot\nabla_i W_{ij}\right)\delta^{\alpha\beta} \tag{3.29}$$

3.4.3 Smoothing Particle Hydrodynamics approximation of energy

The solution of the energy evolution in SPH uses the same definition of the strain component presented in Equation 3.29. For the pressure work approximation the implementation is the same as the one presented in [43]. The energy evolution in SPH depends on the pressure work approximation. The SPH approximation has many forms in the literature, the two more frequent forms for approximating the particle energy are:

$$\frac{De_i}{Dt} = \frac{1}{2}\sum_{j=1}^{N} m_j \frac{p_i + p_j}{\rho_i \rho_j} \nu_{ij}^{\beta} \frac{\partial W_{ij}}{\partial x_i^{\beta}} + \frac{\mu_i}{2\rho_i} \epsilon_i^{\alpha\beta} \epsilon_j^{\alpha\beta} \tag{3.30}$$

And

$$\frac{De_i}{Dt} = \frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nu_{ij}^{\beta} \frac{\partial W_{ij}}{\partial x_i^{\beta}} + \frac{\mu_i}{2\rho_i} \epsilon_i^{\alpha\beta} \epsilon_j^{\alpha\beta} \tag{3.31}$$

Where e_i is the particle energy.

3.5 Conclusions

The SPH implementation of the Navier-Stokes equation provides a mesh-free tool to solve the flow of fluids. The SPH technique is used to evolve the density, momentum, and energy of the particle system. The rule to select the particles interactions and neighbourhoods will be explained in Chapter 5.

The viscosity term can be removed from the Navier-Stokes solution and the system will then solve the Euler equations.

The most popular approaches to the three laws of conservation equations are presented. The selection of these equations for the SPH implementation is based on two main qualities, the first is the literature supporting their efficiency in solving complex fluid problems, but more importantly all equations were selected considering their portability to the parallel implementation on CUDA.

Chapter 4

SMOOTHING PARTICLE HYDRODYNAMICS IMPLEMENTATION FOR SOLID DEFORMATION

4.1 Introduction

In this chapter the numerical implementation of the solid mechanics constitutive equations using the SPH method is presented. The numerical implementation provides with a computational method to model mesh-free solid objects.

In the work presented in this theses the solid inclusion is used to solve the solid-fluid interactions in the studied of porous media permeability and to obtained the material characteristics of solids objects under stress.

The SPH method [63] can be adapted to provide a powerful mathematical tool for solving the equations describing the mechanical behaviour of complex solid continuum media. There are different examples in the literature with applications in fluids [5], solids [50], gases and the interaction between different media [7].

SPH can be simplified to show the behaviour of solids in the area of computer-graphics animation e.g [7] and [52].

In recent publications SPH has also been applied to solve solid mediums undergoing different interactions and deformations. Some common applications include the representation of complex solid and fluid interaction in different areas of study [54] [55].

The correct numerical solution to the behaviour of the phenomena being studied presents a challenge when dealing with solid materials. The problems is due to the different types of solid materials and the constitutive equations that are need it to implement each one of the numerical solution. For Some examples of mesh-free methods on the literature see [56], [72] and [73]

4.2 Solid characteristics classification

In this section the different types of solids that the SPH implementation can solve are discussed. The SPH implementation has the ability to solve the following solids:

I ***Hookean elastic:*** This type of elastic solids follow Hook's law to get their strain stress relation.

II ***Isotropic homogeneous elastic:*** For this type of solid the stiffness tensor has no preferred direction, an applied force will give the same displacements no matter the direction in which the force is applied.

III ***Viscoelastic:*** A new acceleration term is included to model the behaviour of the viscosity in the solids

The final type of particles defined on the SPH simulation is a solid particle that only functions as a boundary particles during the fluid simulation, their physical values will never change and during collision they will not suffer any impulse response.

4.3 Stress

A body under load suffers from surfaces forces and body forces[74]. A body force acts throughout the object volume, some examples of body forces are: gravity, magnetic force, and electrostatic force. Surface forces act only on parts of the body, the surface forces at any point in the body are described through the use of the stress vector.

4.3.1 Cauchy Stress Tensor

Figure 4.1 shows a homogeneous, isotropic body with surface S and volume V; a point $\mathbf{p}$ inside the volume and a new surface plane S^* (cutting plane) passing through the point $\mathbf{p}$ are defined inside the object surface.

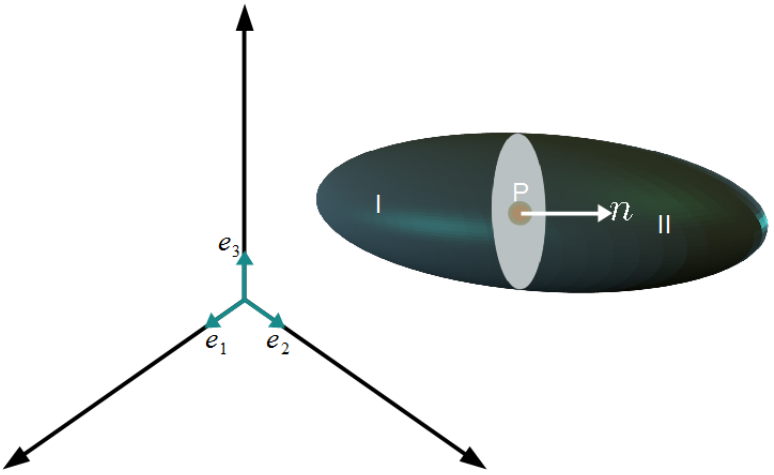


Figure 4.1: **Continuum volume sample point** - The figure shows a solid continuum volume with a cutting plane S^* over the point $\mathbf{p}$. The plane cuts the body into two section I and II. The point is used as references to defined the load effect on the continuum.

The plane S^* cuts the body into two sections, The free body I in Figure 4.2 is defined by selecting the section where the normal $\mathbf{n}$ of S^* is defined. The point $\mathbf{p}$ is used to defined the surface forces $\mathbf{F}$ and the stress cause over the surface S^*.

Considering a surface force $\mathbf{\Delta F}$ acting over a small surface ΔS^* surrounding the point $\mathbf{p}$, the stress vector at the point $\mathbf{p}$ on the plane S^* is given by

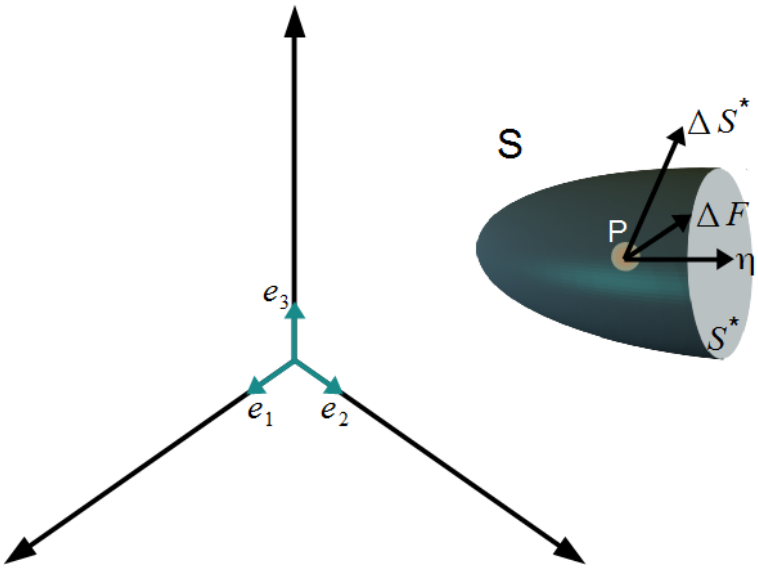


Figure 4.2: **Continuum volume under load** - Using the free body section I, the stress caused by the force $\mathbf{\Delta fs}$ acting over the surface ΔS^* surrounding the point $\mathbf{p}$ can be defined.

$$\mathbf{t_n} = \lim_{\Delta S^* \to 0} \frac{\mathbf{\Delta F}}{\Delta S^*} = \frac{d\mathbf{F}}{dS^*} \tag{4.1}$$

Depending on the orientation of the plane surface S^* with respect to $\mathbf{n}$ the stress can be defined by two components

- **Normal stress:** $\sigma_\mathbf{n}$ is the stress normal to the plane surface

$$\sigma_\mathbf{n} = \lim_{\Delta S^* \to 0} \frac{\mathbf{\Delta F_n}}{\Delta S^*} = \frac{d\mathbf{F_n}}{dS^*} \tag{4.2}$$

- **Shearing stress:** τ is the stress parallel to the plane surface

$$\tau = \lim_{\Delta S^* \to 0} \frac{\mathbf{\Delta F_s}}{\Delta S^*} = \frac{d\mathbf{F}_s}{dS^*} \tag{4.3}$$

There are three normal stress components and six shearing stress. The nine components of the stress σ are given by:

$$\sigma = \sigma^{\alpha\beta} = \begin{bmatrix} \sigma^{xx} & \sigma^{xy} & \sigma^{xz} \\ \sigma^{yx} & \sigma^{yy} & \sigma^{yz} \\ \sigma^{zx} & \sigma^{zy} & \sigma^{zz} \end{bmatrix} = \begin{bmatrix} \sigma^{x} & \tau^{xy} & \tau^{xz} \\ \tau^{yx} & \sigma^{y} & \tau^{yz} \\ \tau^{zx} & \tau^{zy} & \sigma^{z} \end{bmatrix} \tag{4.4}$$

Where in the stress tensor $\sigma^{\alpha\beta}$, α determines the spatial direction of the surface normal where the stress acts and β is the direction of the stress component.

4.4 Solid deformation

The deformation of the shape of solid materials is commonly described on the Lagrangian references frame by the strain and the stress tensors. These can be defined using only six spatial components when symmetry is considered [75]. In the SPH implementation the Lagrangian strain is used to describe

the solid objects deformation. The different solid types (e.g. elastic, Isotropic homogeneous elastic and viscoelastic) that can be solved by the SPH implementation depends on the constitutive equations that relate the strain and the stress for that solid type.

4.4.1 Displacement vector

In order to obtain a measurement of the deformation of a solid material, first it is necessary to define the initial conditions of the solid object. This initial conditions provide with the reference undeformed state configuration of the material.

When the solid has been affected by a load it could suffer a rigid body motion and undergo a deformation. The relation between the undeformed state and the deformed state defines the object strain.

Figure 4.3 shows the vectorial relation between the two states (deformed and undeformed).

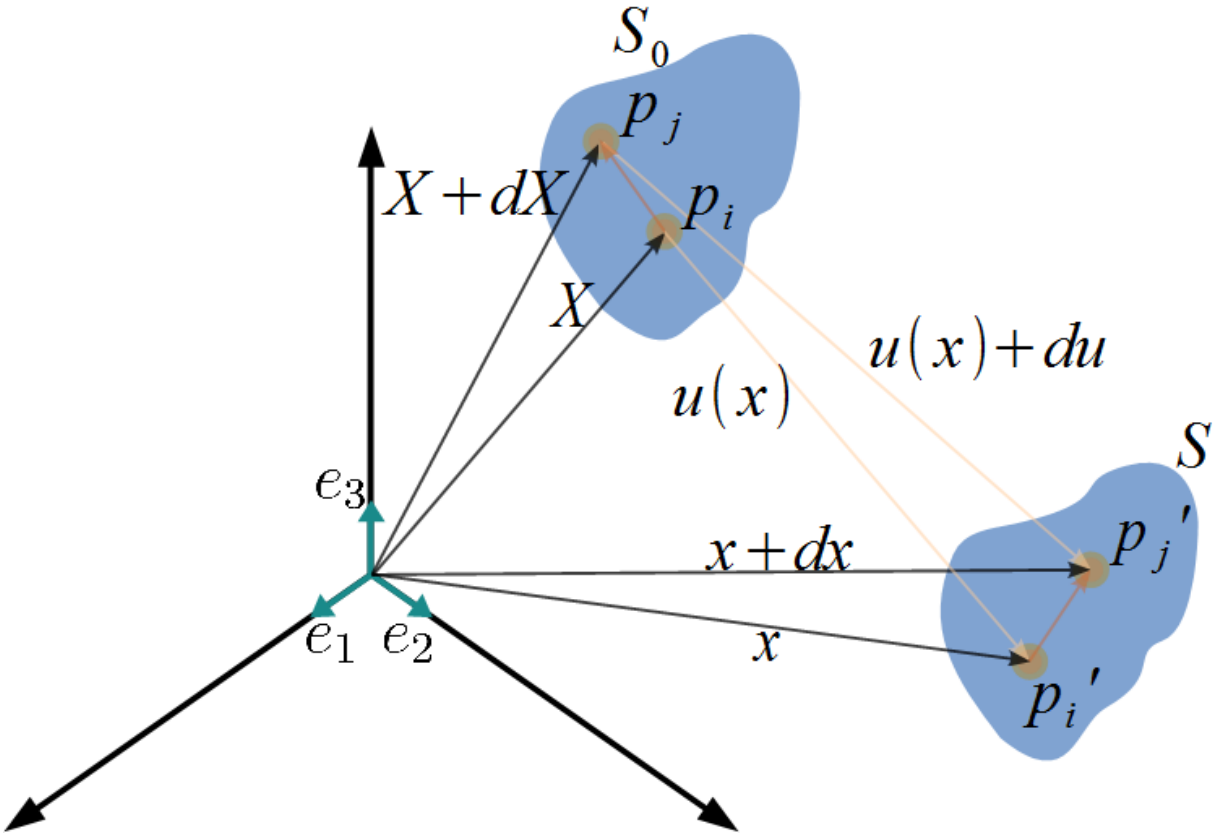


Figure 4.3: **Displacement vector definition** - The figure shows the deformation of a undeformed solid object under a load. The solid will have a rigid body motion and a deformation. The deformation is shown as a change in the internal position of $\mathbf{p}_i$ with respect to the rest of the object's differential points (e.g. $\mathbf{p}_j$).

The particle with position $\mathbf{p}_i$ has components $X = X_1e_1 + X_2e_2 + X_2e_2$ where e_1, e_2, e_3 are the Cartesian axis of reference. Adding the time component, the point p_i is defined as having the coordinates (X_1, X_2, X_3, t). After the load the point will have a new location given by $\mathbf{p}'_i$ at time $t + \Delta t$ with coordinates $(x_1, x_2, x_3, t + \Delta t)$.

The displacement between $\mathbf{p}_i$ and $\mathbf{p}'_i$ is defined by the difference in position [76]:

$$\mathbf{u}(\mathbf{X}, \mathbf{t}) = \mathbf{x}(\mathbf{X}, \mathbf{t}) - \mathbf{X}(t_0) \tag{4.5}$$

Where $\mathbf{u}$ is the displacement vector. The deformation and rigid body motion in the Lagrangian formulation can then be expressed using the initial position as $\mathbf{X}$ at time t_0 and the displacement vector

$$\mathbf{x}\left(X_1, X_2, X_3, t\right) = \mathbf{u}\left(X_1, X_2, X_3, t\right) + \mathbf{X}\left(t_o\right) \tag{4.6}$$

Using the neighbour point $\mathbf{p}_j$ with undeformed position defined by $\mathbf{X}+d\mathbf{X}$ and with deformed position given by $\mathbf{x}+d\mathbf{x}$, using the displacement vector $d\mathbf{x}$ defined as

$$\begin{aligned} \mathbf{x} + d\mathbf{x} &= (\mathbf{X} + d\mathbf{X}) + \mathbf{u}\left(\mathbf{X} + d\mathbf{X}\right) \\ d\mathbf{x} &= \mathbf{X} - \mathbf{x} + d\mathbf{X} + \mathbf{u}\left(\mathbf{X} + d\mathbf{X}\right) \\ d\mathbf{x} &= d\mathbf{X} - \mathbf{u}\left(\mathbf{X}\right) + \mathbf{u}\left(\mathbf{X} + d\mathbf{X}\right) \\ \text{And:} & \\ d\mathbf{u} &= \mathbf{u}\left(\mathbf{X} + d\mathbf{X}\right) - \mathbf{u}\left(\mathbf{X}\right) \\ \text{Finally:} & \\ d\mathbf{x} &= d\mathbf{X} + d\mathbf{u} \end{aligned} \tag{4.7}$$

Where $d\mathbf{u}$ is the relative differential displacement vector between the points $\mathbf{p}'_i$ and $\mathbf{p}'_j$

4.4.2 Displacement gradient

The gradient of the displacement vector field shows how the quantity changes from point to point in a 3D space.

To obtain the gradient of the displacement an approximation using a Taylor series expansion is performed on point $\mathbf{p}_i$ to obtain the components of the displacement vector of $\mathbf{p}_j$.

$$\begin{aligned} \mathbf{u_i}\left(\mathbf{X} + d\mathbf{X}\right) &= \mathbf{u_i}\left(\mathbf{X}, t\right) + d\mathbf{u_i} \\ \mathbf{u_i}\left(\mathbf{X} + d\mathbf{X}\right) &= \mathbf{u_i}\left(\mathbf{X}, t\right) + \frac{\partial u_i}{\partial X_j}(x_j - X_j) + O\left(x_j - X_j\right)^2 \end{aligned} \tag{4.8}$$

Neglecting the higher order terms:

$$\begin{aligned} \mathbf{u_i}\left(\mathbf{X} + d\mathbf{X}\right) &\approx \mathbf{u_i}\left(\mathbf{X}, t\right) + \frac{\partial u_i}{\partial X_j} \cdot d\mathbf{X_j} \\ &\approx \mathbf{u}\left(\mathbf{X}, t\right) + \nabla_X \mathbf{u} \cdot d\mathbf{X} \end{aligned} \tag{4.9}$$

Substituting this in Equation 4.7

$$\begin{aligned} d\mathbf{x} &= d\mathbf{X} + d\mathbf{u} \\ &= d\mathbf{X} + \nabla_X \mathbf{u}\left(\mathbf{X}, t\right) \cdot d\mathbf{X} \\ &= \left(I + \nabla_X \mathbf{u}\left(\mathbf{X}, t\right)\right) d\mathbf{X} \end{aligned} \tag{4.10}$$

Where I is the identity matrix And $(1 + \nabla_X \mathbf{u}\left(\mathbf{X}, t\right))$ is the material deformation gradient tensor $\mathbf{F}\left(\mathbf{X}, t\right)$. $\mathbf{F}$ provides the local deformation at a point $\mathbf{p}_i$ due to the changes from the undeformed to the deformed state, material continuity is assumed during the time differential dt. $d\mathbf{x}$ can be expressed using the partial derivative with respect to the original undeformed state $d\mathbf{X}$

$$\begin{aligned} d\mathbf{x} &= \frac{\partial \mathbf{x}}{\partial \mathbf{X}} d\mathbf{X} \\ &= \nabla_X \mathbf{x}\left(\mathbf{X}, t\right) \cdot d\mathbf{X} \\ &= \mathbf{F}\left(\mathbf{X}, t\right) d\mathbf{X} \end{aligned} \tag{4.11}$$

Considering Equation 4.10 and Equation 4.11 the components of $\mathbf{F}(\mathbf{X},t)$ are

$$\mathbf{F}(\mathbf{X},t) = \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_2}{\partial X_3} \\ \frac{\partial x_3}{\partial X_1} & \frac{\partial x_3}{\partial X_2} & \frac{\partial x_3}{\partial X_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} \frac{\partial u_1}{\partial X_1} & \frac{\partial u_1}{\partial X_2} & \frac{\partial u_1}{\partial X_3} \\ \frac{\partial u_2}{\partial X_1} & \frac{\partial u_2}{\partial X_2} & \frac{\partial u_2}{\partial X_3} \\ \frac{\partial u_3}{\partial X_1} & \frac{\partial u_3}{\partial X_2} & \frac{\partial u_3}{\partial X_3} \end{bmatrix} \tag{4.12}$$

4.4.3 Strain and infinitesimal strain measures

Using the relation between the differential elements $d\mathbf{x}$ and $d\mathbf{X}$ and Equation 4.11, their square differences can be expressed as

$$\begin{aligned} |d\mathbf{x}|^2 - |d\mathbf{X}|^2 &= (d\mathbf{x}\cdot d\mathbf{x}) - (d\mathbf{X}\cdot d\mathbf{X}) \\ &= (\nabla_X \mathbf{x}\cdot d\mathbf{X})\cdot(\nabla_X \mathbf{x}\cdot d\mathbf{X}) - (d\mathbf{X}\cdot d\mathbf{X}) \\ &= (\mathbf{F}d\mathbf{X})\cdot(\mathbf{F}d\mathbf{X}) - (d\mathbf{X}\cdot d\mathbf{X}) \end{aligned} \tag{4.13}$$

Using Einstein's summation index Equation 4.13 is expressed as

$$\begin{aligned} |d\mathbf{x}|^2 - |d\mathbf{X}|^2 &= (\mathbf{F}_{\alpha\beta}d\mathbf{X}_\beta)(\mathbf{F}_{\alpha\gamma}d\mathbf{X}_\gamma) - (d\mathbf{X}_\gamma\cdot d\mathbf{X}_\gamma) \\ &= (\mathbf{F}_{\alpha\beta}\mathbf{F}_{\alpha\gamma} - \delta_{\beta\gamma})\, d\mathbf{X}_\beta d\mathbf{X}_\gamma \\ &= d\mathbf{X}\left(\mathbf{F}^T\mathbf{F} - I\right) d\mathbf{X} \end{aligned} \tag{4.14}$$

The Lagrangian strain tensor is defined as:

$$\begin{aligned} \mathbf{E} &= \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - I\right) \\ &= \frac{1}{2}\left(\nabla_X\mathbf{u} + \nabla_X\mathbf{u}^T + \nabla_X\mathbf{u}^T\nabla_X\mathbf{u}\right) \end{aligned} \tag{4.15}$$

In the computational approach the time step Δt tends to dt in order to keep the displacement gradient components small. When this condition is met the measures of the small displacement are represented by a infinitesimal strain tensor ϵ. In the infinitesimal strain definition the higher order components of the strain $\nabla_X\mathbf{u}^T\nabla_X\mathbf{u}$ are neglected. In this case the strain tensor ϵ is called the Green strain tensor:

$$\epsilon = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right) \tag{4.16}$$

4.4.4 Constitutive equations

The behaviour of the solid material depends on:

I ***Displacement vector - Strain relation:*** As presented in Section 4.4.3 the deformation of points inside the solid can be described by the infinitesimal strain tensor and the displacement vector of each point.

II ***Constitutive equation of the Strain-Stress relation:*** The constitutive equation provides a relation describing the effect a change in the strain tensor will have on the solid stress tensor value.

III ***Laws of momentum:*** Using the stress and the physical description of points in the space, the behaviour of the solid is described by the equations of conservation of momentum. The laws of momentum will be described in Section 4.4.5.

The relation between the strain and stress has various definitions depending on the type of material been studied. In this section the constitutive equations of the following materials are discussed:

- **Hookean elastic solid:** This is the simples approach to simulate solid elasticity. Solids behave according to this law if the load on the solid is not greater than the solid elastic limit [77, 78].

- **Isotropic elastic:** An isotropic solid mechanical properties can be defined independently from the reference frame in which is defined [74].

- **Viscoelastic solid:** Viscoelastic solids under loading and unloading suffer from energy loss or dissipation.

Hookean solid

Hook's law is used to solve simplified problems where the solid can be considered an ideal material. This law was proposed by Robert Hooke in 1768 and states that "the power of any springy body is in the same proportion with the extension." [79].

This law considers that the material follows linear elasticity behaviour. The solid material will return to its original configuration once the load on the solid has been removed. Figure 4.4 shows the relation between strain and stress for a linear elastic solid.

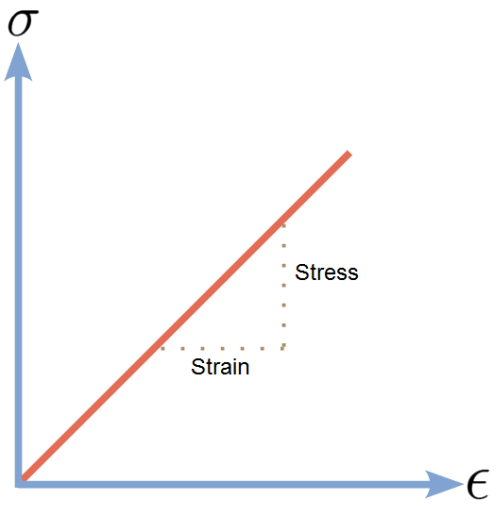


Figure 4.4: **Strain-stress relation for linear elastic solid** - In the Linear elastic solid behaviour. The solid returns to its original configuration when the load is removed, this material behaviour is described as a perfect spring system.

The linear elastic solid follow the linear relation [80]:

$$\begin{aligned} \sigma &= \mathbf{C}\epsilon \\ \sigma^{\alpha\beta} &= C^{\alpha\beta km}\epsilon^{km} \end{aligned} \tag{4.17}$$

Where $C^{\alpha\beta km}$ is the elastic tensor, and α, β, k and m are the summation coefficients used to represent the 81 components of the tensor. The tensor components can be reduced to 21 independent elastic constants.

Isotropic solid

An isotopic solid is defined as a solid whose mechanical properties are independent of the reference frame [74]. For isotropic materials the constitutive equation has only two elastic parameters that are used to defined the different solid materials.

Isotropic materials have an elastic tensor **C** defined as a fourth-order isotropic tensor. To obtain the fourth-order isotropic tensor a second order isotropic tensor is used.

The Kronecker delta $\delta^{\alpha\beta}$ tensor is the one used to obtained the fourth-order isotropic tensors [74]. Using $\delta^{\alpha\beta}$ the following three isotropic fourth-order tensors can be defined:

$$\begin{aligned} A^{\alpha\beta km} &= \delta^{\alpha\beta}\delta^{km} \\ B^{\alpha\beta km} &= \delta^{\alpha k}\delta^{\beta m} \\ H^{\alpha\beta km} &= \delta^{\alpha m}\delta^{\beta k} \end{aligned} \tag{4.18}$$

The elastic tensor in terms of the three isotropic fourth-order tensor [74] is presented in Equation 4.19:

$$C^{\alpha\beta km} = \lambda A^{\alpha\beta km} + \varphi B^{\alpha\beta km} + \eta H^{\alpha\beta km} \tag{4.19}$$

Where λ, φ and η are material constants, and the stress is defined as:

$$\begin{aligned} \sigma^{\alpha\beta} &= C^{\alpha\beta km}\epsilon^{km} \\ &= \lambda\delta^{\alpha\beta}\delta^{km}\epsilon^{km} + \varphi\delta^{\alpha k}\delta^{\beta m}\epsilon^{km} + \eta\delta^{\alpha m}\delta^{\beta k}\epsilon^{km} \end{aligned} \tag{4.20}$$

Reorganising the terms and simplifying the Kronecker deltas:

$$\sigma^{\alpha\beta} = \lambda\epsilon^{kk}\delta^{\alpha\beta} + (\varphi + \eta)\epsilon^{\alpha\beta} \tag{4.21}$$

$(\varphi + \eta)$ are commonly represented by 2μ so the final expression for the stress of an isotropic material:

$$\sigma^{\alpha\beta} = \lambda\epsilon^{kk}\delta^{\alpha\beta} + 2\mu\epsilon^{\alpha\beta} \tag{4.22}$$

Where λ and μ are the Lamé constants.

Viscoelastic solid

When the modelling of a solid requires that the system dissipates energy over time a viscoelastic model is implemented.

Viscoelastic solids under the process of loading and unloading suffer from energy lost or dissipation (hysteresis). Figure 4.5 shows the behaviour of the relation between the stress and strain of a solid material with hysteresis.

Viscoelastic materials have two main characteristics, stress relaxation and creep.

Stress relaxation is the phenomena of stress decreasing over time when the strain is keep constant (Figure 4.6).

The creep property describes the increase in deformation under a constant stress until an asymptotic level of strain is reached (Figure 4.7).

The relation between the strain and stress is obtained using the Kelvin-Voigt model [81], in Figure 4.8 is shown the geometry relation of the model .

The viscosity component η provides the model with a viscous damper strain component ϵ_D and the elastic spring component ϵ_S. In Figure 4.8 the components are shown to be parallel so the total strain and the strain on each component are identical.

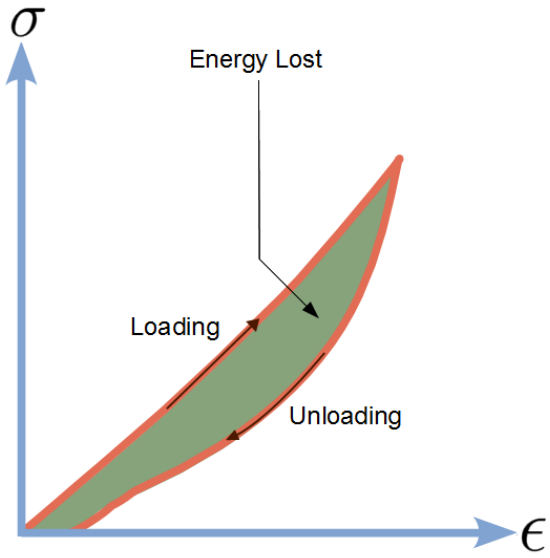


Figure 4.5: **Strain-stress relation for viscoelastic solid** - The loading curve of the stress - strain relation is higher than the unloading curve. The hysteresis is the area between the loading and unloading curve.

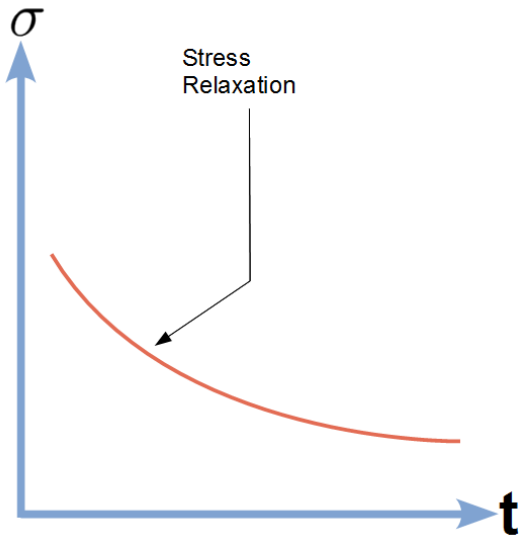


Figure 4.6: **Stress relaxation** - The figures shows the viscoelastic solid material stress relaxation over time for a constant strain.

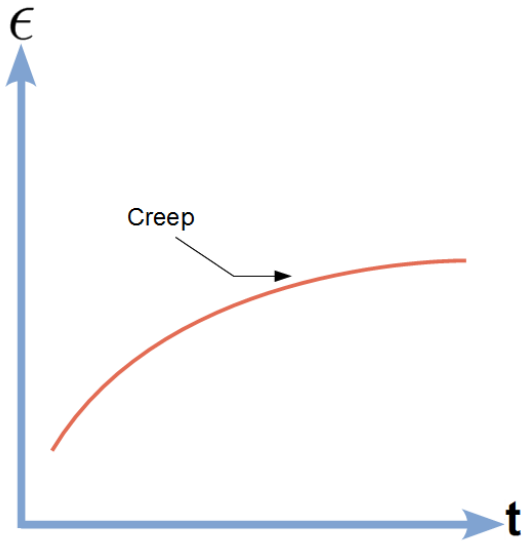


Figure 4.7: **Creep under relaxation** - Viscoelastic solid material creep over time for a constant strain.

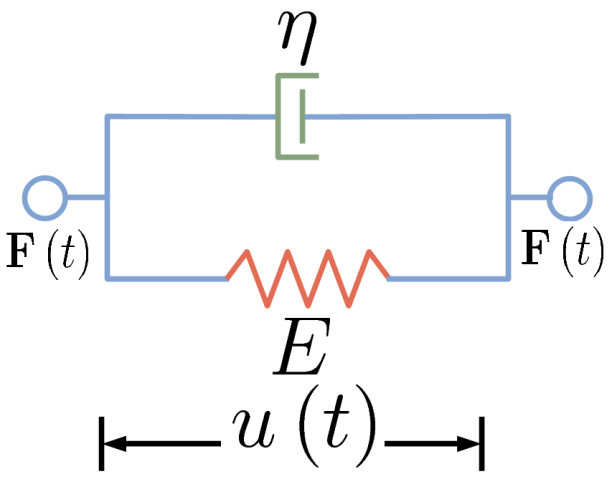


Figure 4.8: **kelvin-Voight strain-stress relation** - In the figure is presented the kelvin-voight model geometry. The mechanical analogues are used to describe the model of the relation between the components (elastic and viscosity) and the change in the displacement vector of a point.

$$\epsilon_{Total} = \epsilon_D = \epsilon_S \epsilon_{Total} \quad = \epsilon_D + \epsilon_S \tag{4.23}$$

The stress σ and strain ϵ relation and their rates of changes are modelled with the following equation:

$$\sigma(t) = E\epsilon(t) + \eta \frac{d\epsilon(t)}{dt} \tag{4.24}$$

Where E is the material elastic coefficient and η is its viscosity. In viscoelastic solids the stress depends on the strain ϵ and the strain rate $\dot{\epsilon} = \frac{d\epsilon(t)}{dt}$

4.4.5 Law of conservation of momentum

Using the relation between the displacement tensor and the strain, the constitutive equation for different types of solids and the equation of momentum the behaviour of a solid under stress can be described.

Using the definition of a solid material presented in Sections 4.4.1 and 4.4.2, the linear momentum of a point i is given by:

$$\mathbf{M_i}(\mathbf{t}) = \int_V \rho \nu_i dV \tag{4.25}$$

Where the momentum is equal to the forces acting on the object (body forces and surface forces). Equation 4.25 can be expressed on terms of the surface forces and body forces as:

$$\frac{d}{dt}\int_V \rho \nu_i dV = \int_S \sigma^{\alpha\beta} dS + \int_V \rho b_i dV \tag{4.26}$$

Where b_i is the vector representing the body forces. Passing the surface integral to a volume integral Equation 4.26 can be expressed as:

$$\frac{d}{dt}\int_V \rho \nu_i dV = \int_V \sigma^{\alpha\beta}{}_{,\beta}\, dV + \int_V \rho b_i dV \tag{4.27}$$

The body forces are solved separately as external forces since they act over the entire object volume and can be approximated to be equal for the differential control volume, the momentum equation using the material derivative then can be expressed as:

$$\frac{D\nu^\alpha}{Dt} = -\rho \frac{\partial \sigma^{\alpha\beta}}{\partial x^\beta} \tag{4.28}$$

4.5 Smoothing Particle Hydrodynamics solution

All the previously described SPH approximations to model particle behaviour are used in this section to obtain the discrete solution of solid materials. There are only two different sets of equations that need to be solved for solids:

- **Internal force:** The solution for the stress-strain relation and the final momentum equation for solid particles will be based on the SPH equations derived in this section.
- **Particle collisions:** The interaction between frontier solid particles or solid-fluid particle interactions is solved using a collision system and an anti-penetration force, these will be derived in Chapter 5.

In the following section the SPH equations to solve the internal forces on a solid are presented.

4.5.1 Smoothing Particle Hydrodynamics internal forces: momentum equation

The first step is to define the displacement vector in terms of the particle system. Having as a reference the original object configuration x_i^o of each particle i belonging to the particle system of size N of the object, the displacement of a particle is obtained using Equation 4.29 .

$$\Delta u_i = x_i - x_i^o \tag{4.29}$$

Where x_i is the position of the particle on the current time-step and x_i^o is the original particle position, during a SPH simulation the original particle configuration x_i^o can be redefined when the solid will not return to its original configuration, this redefinition could be due to plastic deformation or when a solid object is separated into two or more components that result in permanent surface deformation.

All the forces are solved in a pair-wise way, so for each pair of particles a displacement relation between them is calculated. The displacement vector between two particles is defined as:

$$\begin{aligned} u_{ji} &= u_j - u_i \\ &= \left(x_j - x_j^o\right) - \left(x_i - x_i^o\right) \\ &= \left(x_j - x_i\right) - \left(x_j^o - x_i^o\right) \end{aligned} \tag{4.30}$$

The gradient tensor of u_{ji} with respect to particle i is defined as ∇u_{ji}, the gradient is obtained with the help of the first spatial derivative of the smoothing kernel $\nabla_i W_{ij}$

$$\langle \nabla u_i \rangle = \sum_j \frac{m_j}{\rho_j} u_{ji} \nabla_i W_{ij} \tag{4.31}$$

The strain over a particle is derived from the equation of strain after the summation in Equation 4.29 has been solved for each particle in the solid particle system. Then the corresponding stress is calculated using the constitutive equation of the solid.

The momentum equation for solid objects is solved using the equation defined on Section 3.4.2, Equation 3.26:

$$\frac{D\nu^\alpha}{Dt} = \sum_{j=1}^{N} m_j \left(\frac{\sigma_i^{\alpha\beta}}{\rho_i^2} + \frac{\sigma_j^{\alpha\beta}}{\rho_j^2} \right) \nabla_i W_{ij} \tag{4.32}$$

4.5.2 Particle collision and anti-penetration forces

Solid particles that belong to different objects and solid-fluid interaction require the solution of a collision responses. To improve the computational time the octree neighbourhood search approach presented in Section 5.3 is used to find all the particles that are in proximity and have a possibility of colliding. The test of proximity [82] for two particles is defined as:

$$\begin{aligned} \mathbf{Sd} &= D\left(\mathbf{p_i}, \mathbf{p_j}\right) \\ \mathbf{Sr} &= \left(\iota h_i + \iota h_j\right)^2 \\ \mathbf{Sd} - \mathbf{Sr} &= \begin{cases} \text{True} & \text{if } \mathbf{Sd} - \mathbf{Sr} <= 0 \\ \text{False} & \text{if } \mathbf{Sd} - \mathbf{Sr} > 0 \end{cases} \end{aligned} \tag{4.33}$$

Where **Sd** is the distance between the position of two particles i and j, **Sr** is the square sum of the particle radii determined by ιh_i, where ι is a scale factor of the smoothing length, the scale factor is used to determine how big the particle volume of influences is.

If the particles are intersecting then a test for collision is performed. There are three cases to be solved depending on the result of the collision test. Before the test is solved a threshold th is defined, this threshold has a value close to 0 and depends on the maximum velocity of the particle and the time differential. The possible results are [83]:

- **Collision:** This condition is true when the value of $\nu_{rel} < th$. If this condition is true then the particles need to be solved by the collision algorithm.
- **Contact:** This condition is true when $\nu_{rel} = th$. These particles will remain in contact and a force preventing interpenetration must be computed.
- **Separation:** If $\nu_{rel} > th$ then the particles are separating and no operation needs to be perform on the particle pair.

Where ν_{rel} is the relative velocity between the pair of particles:

$$\nu_{rel} = (\nu_i - \nu_j) \cdot \mathbf{Sd} \tag{4.34}$$

Particle collision

If two particles have a $\nu_{rel} < th$, the collision response is calculated for the particle pair. The fist step is to obtain the impulse [84] j_p scalar value

$$j_p = \frac{-(1+e)\,\nu_{rel}}{\mathbf{Sd} \cdot \mathbf{Sd}\left[\frac{1}{m_i} + \frac{1}{m_j}\right]} \tag{4.35}$$

where e is the coefficient of restitution (determines if the collision is plastic, elastic or in between), the impulse is applied to the particles i and j and the final velocity ν^f after collision is obtained:

$$\begin{aligned} \nu_i^f &= \nu_i + \frac{j_p}{m_i}\mathbf{Sd} \\ \nu_j^f &= \nu_j - \frac{j_p}{m_j}\mathbf{Sd} \end{aligned} \tag{4.36}$$

Resting contact

When two particles intersect $\nu_{rel} = th$ a penalty force is introduced to avoid interpenetration and maintain the resting contact. The external force $\mathbf{B}_{ij}$ is:

$$\mathbf{B}_{ij} = \begin{cases} \Omega\left[\left(\frac{r_0}{\mathbf{Sd}}\right)^{n_1} - \left(\frac{r_0}{\mathbf{Sd}}\right)^{n_2}\right]\frac{Sd}{\mathbf{Sd}^2} & \text{if } \left(\frac{r_0}{|\mathbf{Sd}|}\right) <= 1 \\ 0 & \text{if } \left(\frac{r_0}{|\mathbf{Sd}|}\right) > 1 \end{cases} \tag{4.37}$$

Ω is a constant value that should be in the same scale as the square of the largest particle velocity in the problem, n_1 and n_2 are constants with values of 12 and 4 respectively. r_0 is a cut-off distance that prevents particles being affected by the force$\mathbf{B}$ after a given distances. Typically r_0 has a value close to the initial particle spacing.

4.6 Conclusions

The addition of a SPH method to simulate solid particles allows the numerical solution for solid-solid particle interactions and solid-fluid interactions to be obtained. The solid-fluid interaction with virtual solid particles and a collision method is used to solve the porous medium fluid infiltration presented in Section 7.

Solving for solid objects requires the used of extra memory to keep track of the original particle distribution. When the solid behaviour is viscoelastic it is also necessary to maintain the values of the stress σ from the last time step in memory. This increases the memory required to represent solid objects in the SPH implementation.

The simulation of solid objects has the advantage that until the solid suffers a plastic deformation or is divided into multiple solid objects the neighbourhood structure of the system and smoothing kernel does not need to be recalculated at each time step.

Chapter 5

DATA STRUCTURE

5.1 Introduction

One of the most computationally expensive parts of any mesh-free particle-based algorithm is the construction of the data structures that contain the information to solve the computational problem. The first data structure holds the information necessary for the management of the relation between particles.

The list containing the different data structures is created using a search method over all the particles i on the particle system of size N, where $i = 1, 2, \ldots, N$, against all the particles j where $i \neq j$. In most mesh-free particle based algorithms the implementation is required to solve the following cases:

- Particle neighbour list: the list contains all the particles j within a distance radius r_{ij} to the focus particle i that is less than a given constant C_r. The neighbour search is only made between particles that belong to the same object.
- Particle collision list: a collision is defined either between solid boundary particles and fluid particles or two solid boundary particles.
- Particle state list: this list contains the state of the particle. The common states are active or inactive, inactive particles are not considered when building the other two lists.
- Problem specific lists: some problems could need to list more types of relations between the particles.

In the case of a SPH implementation $C_r = k * h_i$, where k is a scale factor that depends on the kernel type and the object type (fluid, solid or gas). Neighbour and collision list creation can be synthesised into a single problem.

5.2 Particle Neighbour Search problem: state of the Art

The Particle Neighbour Search problem comes from the amount of data that needs to be analysed to create each one of the particle lists. Depending on the conditions of the problem the neighbour and collision search may need to be performed at each time step. This makes improvement of the search algorithm necessary to find a more efficient solution of the simulation. In the following section the naive approach to solved the Particle Neighbour Search problem will be presented first. A short introduction to the octree solution for the neighbour search problem will be given on Section 5.3.

5.2.1 Smoothing Particle Hydrodynamics all-pair search

The naive solution to the Neighbour Search problem is the all-pair search approach [5]. For each particle i the distance r_{ij} to every particle j on the N size particle system is obtained. If the distance r_{ij} is less than the support domain of the particle i given by $k * h_i$, then the particle j is added to the neighbour list of particle i (Figure 5.1).

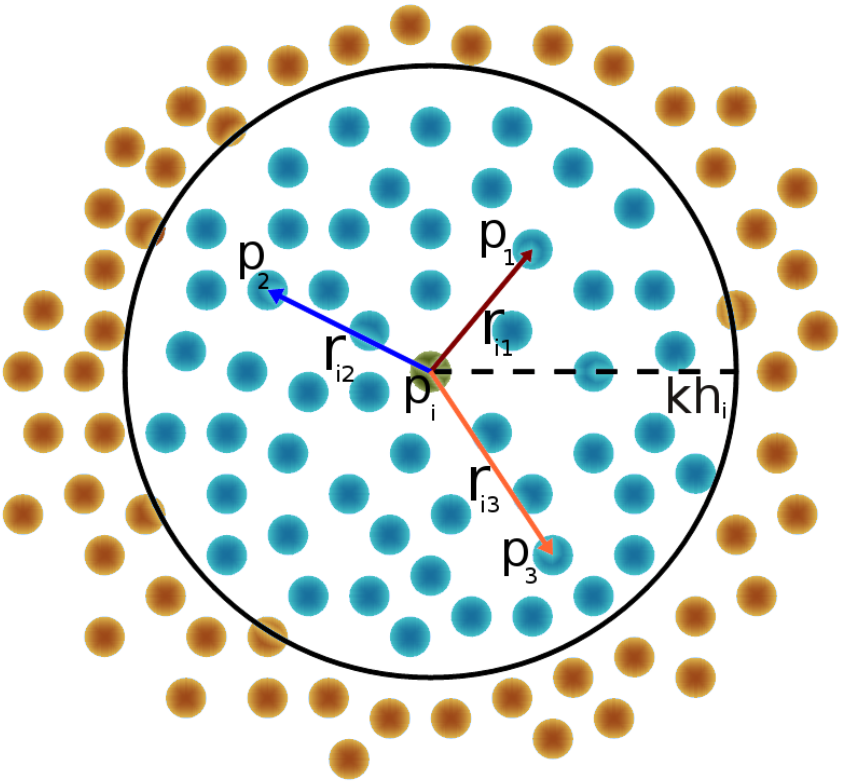


Figure 5.1: **Particle Neighbour Search** - All particles with distances less than the maximum support domain $k * h_i$ belongs to the particle i neighbour. Using the all-pair search approach each particle needs to look for neighbour particles in the whole particle system.

If we use a symmetric smoothing kernel in the definition of the particle system problem then particle j will also have particle i on its neighbour list. In the case of the symmetric smoothing kernel the complexity of the solution is $O\left(N^2\right)$.

To improve the computational solution time a tree search method is implemented.

5.2.2 Tree structure

A tree is a hierarchical representation of a data structure in a graphical form [85]. The tree structure is composed of individual elements called nodes connected by branches. In cases such as the ones presented in this thesis the trees are finite and have a root node that serves as the initial position of the hierarchical representation.

In a tree representation we use relationships between the nodes to define the hierarchy. The node at the top level (level 0) of the tree is the root node, the tree is formed by parent and child nodes. A parent node is higher in the hierarchy than its child (children) and both of them share a branch. The nodes that

do not have any children of their own are called leaf nodes and are the bottom level of the tree branch, in Figure 5.2 is shown a full binary tree and in Figure 5.3 is shown a partial tree.

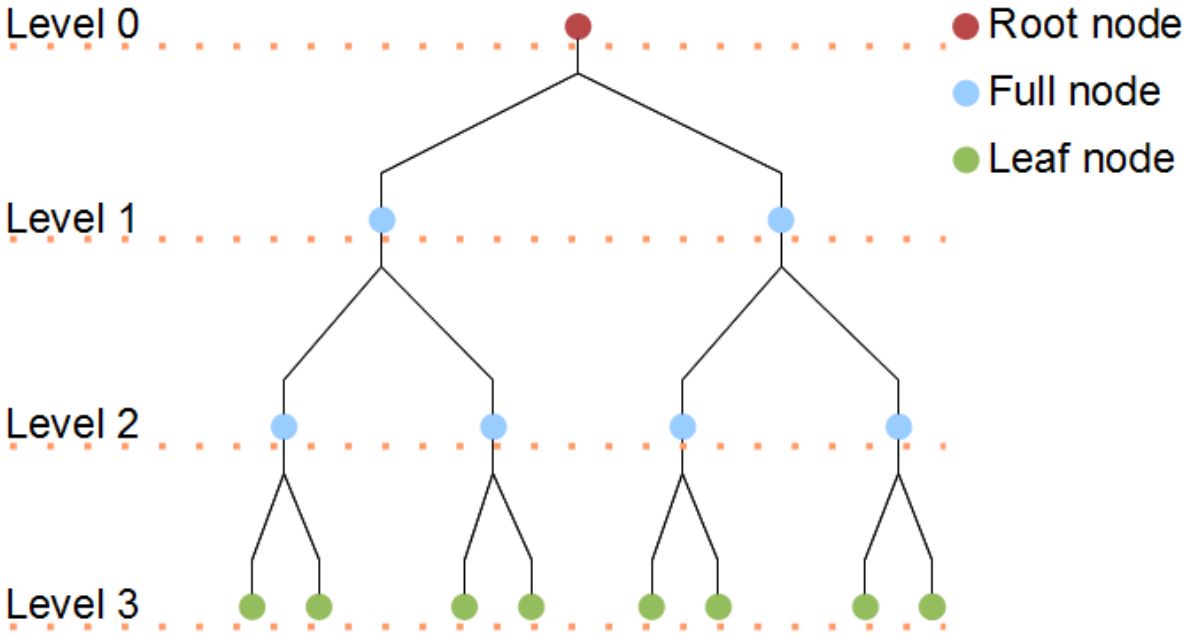


Figure 5.2: **Full tree structure** - The Figure shows a graphic representation of a full binary tree. In a full tree all nodes except the leaves have full children lists.

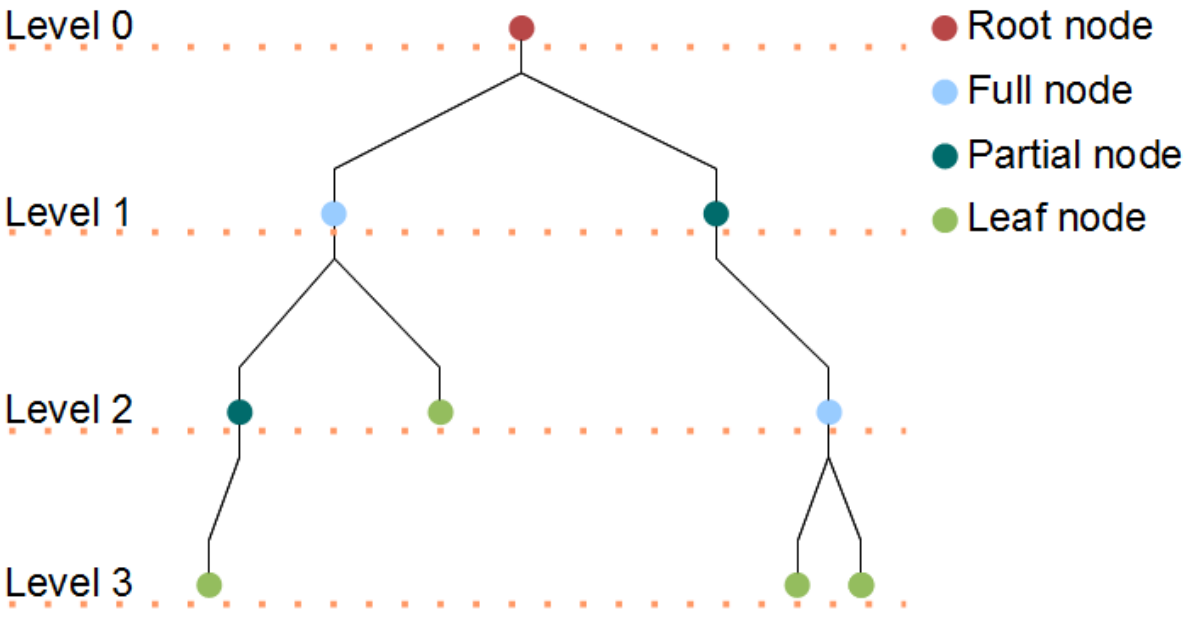


Figure 5.3: **Partial tree structure** - The Figure shows a graphic representation of a partial binary tree. In a partial tree some parent nodes only have one child.

In the SPH computational solution the nodes are stored in structures. The structures contains the necessary information for the SPH solution and a set of memory pointers that are used to store the tree structures and keep the relation between nodes. Each node structure contains a pointer to its parent node and zero or more pointers to the child nodes (Figure 5.4).

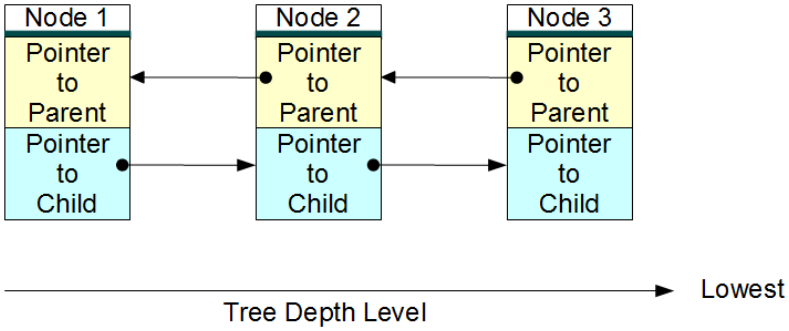


Figure 5.4: **Node pointer structure** - In a tree structure each node has a pointer to the parent node(excluding the top level one) and a pointer to zero or more children.

In the SPH implementation the data structure provided by the tree gives the relation between the full problem volume size and the individual particle volumes. The leaf nodes are used to store the list of particles contained in the space occupied by the node volume. One type of tree used in the computational solution of three-dimensional problems is known as an octree.

5.2.3 Octree

An octree [86] is a type of tree in which each node has between zero and eight child nodes. The octree divides the full problem volume space (volume occupied by particles) into groups of octants. The final structure is an organised hierarchical volumetric tree (Figure 5.5).

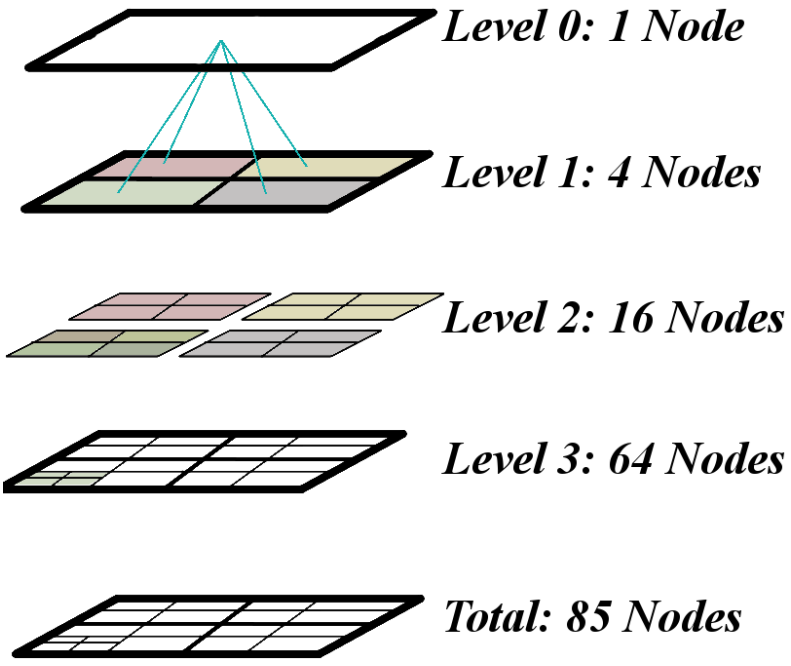


Figure 5.5: **Quadtree division example** - The example shows nodes being divided into quadrant components as the tree is traversed. The final number of nodes in the quadtree is 85 with four levels.

During each operation on the octree structure the parent - child relationship is maintained, preserving the tree structure throughout the computational process. Each node contains a pointer to its children and a pointer its parent node. At the bottom level of a branch the child nodes contain the list of elements belonging to the volume of the child.

Figure 5.6 presents an example of a parent node with a full list of leaf node children, the parent node contains eight pointer to memory address of each one of its children, the children also have a pointer to the parent node and because they are leaf nodes the children nodes also contain a pointer to a linked list where the index of all the elements belonging to the children volume.

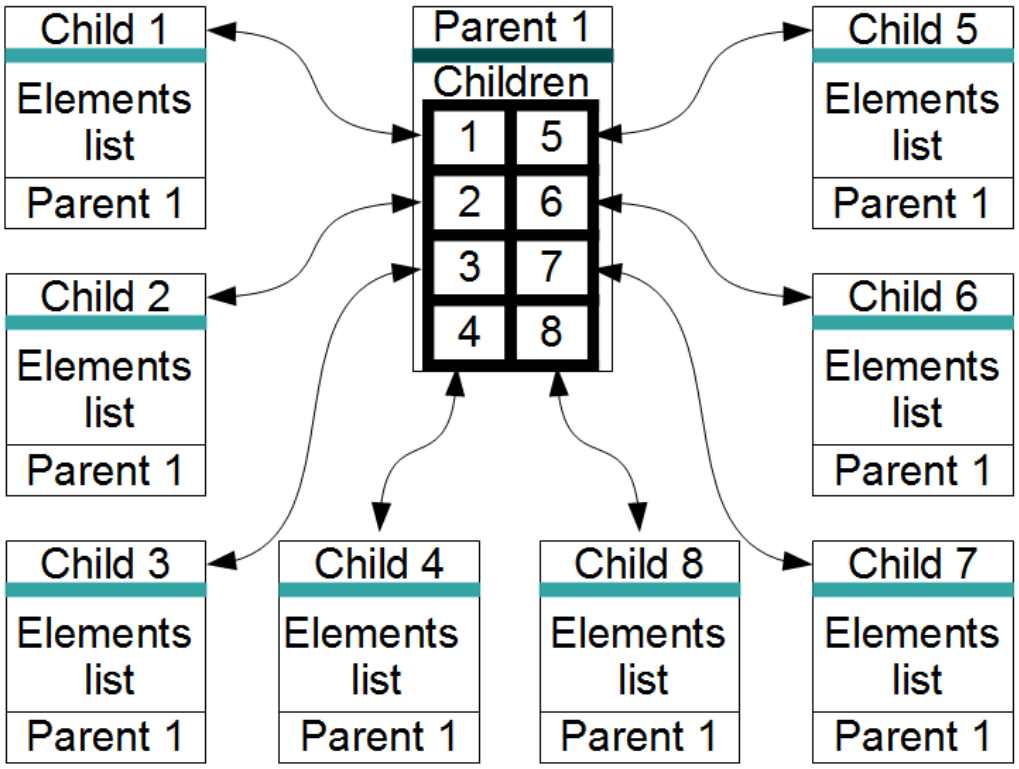


Figure 5.6: **Octant node** - A node on an octree containing pointers to eight children, each child is a leaf. The leaf nodes contain a list of all the elements that belong to that volume.

5.2.4 Octree traversal

When looking for a specific position in space defined by a node, we just need to traverse the tree. The basic tree traversal consist in processing every node in the data structure exactly once, during the traversal a node can be visited more than one time but it should only be process once [87]. Some of the traversal order are:

- Pre-order traversal: do the $node_n$ before anything else, with $n = 1...N$ and N is the total number of nodes.
- Post-order traversal: do the $node_n$ after everything else, with $n = 1...N$ and N is the total number of nodes.
- In-order traversal: do the $node_n$ in between the sub-trees.

On Figure 5.7 an example of the order the different nodes on tree will be visit with the different traverse methods is shown.

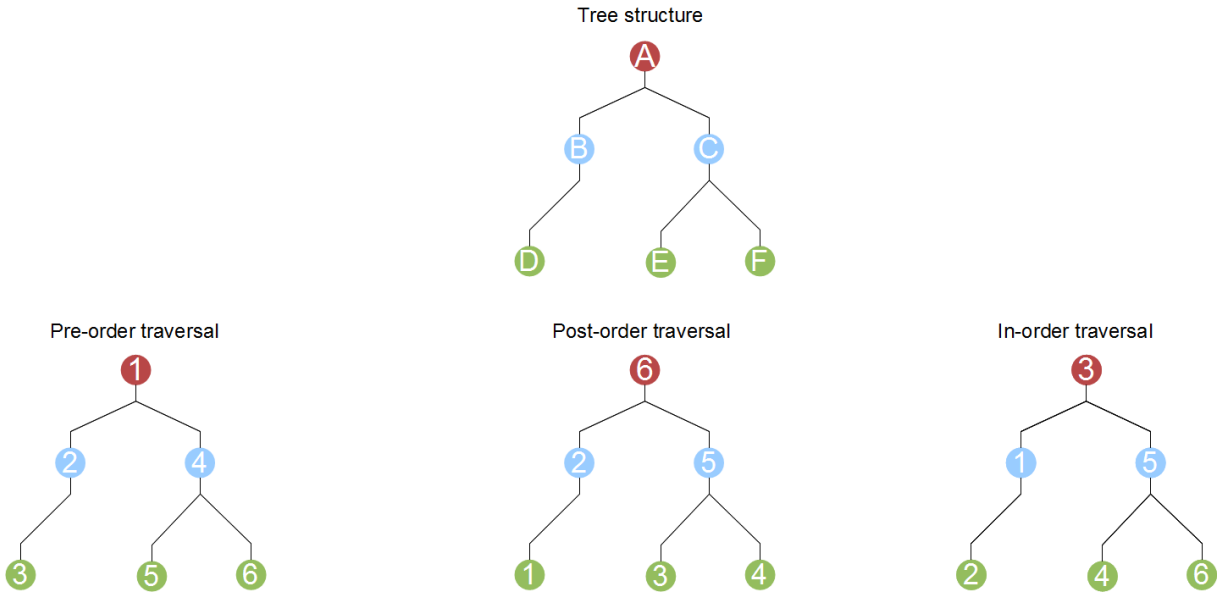


Figure 5.7: **Tree traversal methods** - Different methods to do the tree traversal are shown on the figure, all methods visit all the nodes at least once to process all the nodes.

5.2.5 Morton key

The octree reduces the search range when looking for specific positions in the volumetric space. To further reduce the time required for the octree creation and octree traversal, the Morton key technique [8] was implemented.

The Morton key assigns a 32 bit location code to each object (nodes and elements) in the problem. The key location allows us to navigate the tree structure without needing to start the tree traverse from the root node all the time.

Using this key we have a fast computational method for child creation and element insertion onto the leaf list. Our objects (for example in SPH particles and nodes) contain the node structure from the root parent node to the lowest possible level that can be contain in their 32 bit keys.

The Morton key is built using the space location of each object. Morton keys are assigned to tree nodes using the nodes' centre position. Different object types will require different approaches to obtain the Morton key, the general solution is to select a point that identifies the position of the element, in the case of SPH this is the position of the particles. Another example is when using three-dimensional meshes, in this case the centroid of the triangle can be used to define the position.

After selecting the position of references, the method to obtain the Morton Key for different type of objects is similar. The Morton key is based on a point defining the three-dimensional position of the object (**object_centre**). The first step consists of normalizing the *object_centre* value with respect to the tree size (*tree_size*). The maximum problem volume size is equal to the (*tree_size*) volume.

$$\overline{\mathbf{object\_centre}} = \frac{\mathbf{object\_centre}}{tree\_size} \tag{5.1}$$

After normalisation the values of $\overline{\mathbf{object\_centre}}$ are right shifted to obtained an integer representation.

$$\mathbf{object\_centre_i} = (\overline{\mathbf{object\_centre}})400_{16} \tag{5.2}$$

The values of $\mathbf{object\_centre_i}$ are integer representations of the positions of the objects. The hexadecimal value 400_{16} (10000000000 in binary representation) is used because the final binary key will be 32 bits long and each one of the spatial components (x,y,z) will take 10 bits.

The original interval of values are between $(0, \ldots, 1)$ after multiplying by 400_{16} the new value interval will be $(0, \ldots, 1024)$. Finally the single 32 bit key needs to be obtained.

Using the binary representation of $\mathbf{object\_centre_i}$ the Morton key is obtained by interleaving the three components:

$$(object\_centre_i.x, object\_centre_i.y, object\_centre_i.z) \tag{5.3}$$

The full Morton key will be 31 bits long, the first bit is a place-holder that gives the key direction (which side of the number is going to be the starting point when reading the bits), the other 30 bits are formed by groups of triplets. Each triplet is formed by 1 bit value from each one of the spatial position components in the order zyx, the first triplet is built using the first bit from each component, then the next three bits are taken and form the second triplet, the process is continued until the full 10 bits of each component are used.

In Section 5.3.1 an example on how to obtained a Morton key of an object is shown.

5.3 Octree based Smoothing Particle Hydrodynamics

The SPH implementation requires the following information for each particle in the system at each time step to be able to obtain the computational solution:

- Particle neighbour list: The list for each particle i on the particle system is created by searching over all the particles j on the particle system size N, where $j = 1, 2, \ldots, N$. The particles j that belong to a volume defined by $k * h_i$ (the volume surrounding the focus particle i)and in which $pobj_i = pobj_j$ and $i \neq j$ form part of the neighbour list of particle i.

  Where $ptype_i$ defined the type of particle been solved and $pobj_i$ the object it belongs to. The operations are done for each particle i on the particle system size N

- Particle collision list: A collision is defined either between the solid boundary particles and the fluid particles or two boundary solid particles.

  Fluid - Solid: In the case of the fluid-solid collision the list is created by searching over all the particles j on the particle system size N, where $j = 1, 2, \ldots, N$. The particles j that belong to a volume defined by $kc * h_i$ surrounding and centred on the focus particle i and in which $ptype_i \neq ptype_j$, $pobj_i \neq pobj_j$ and $i \neq j$ form the collision list of particle i. The operations are done for each particle i on the particle system of size N.

  Solid - Solid: In the solid-solid collision, there are two possible cases:

  (a) Same object: The same type of search as for the fluid-solid case is made, the conditions that two particle must fulfill to be colliding is that particle j belongs to the volume defined by $kc * h_i$ and: $ptype_i = ptype_j$, $pobj_i = pobj_j$ and $i \neq j$.

  (b) Different objects: the conditions that two particle must fulfill to be colliding is that the particle j belongs to the volume defined by $kc * h_i$ and: $ptype_i = ptype_j$, $pobj_i \neq pobj_j$ and $i \neq j$.

To create the list defining the SPH problem in each time step an octree implementation is used. The tree T is defined using the tree structure presented in the Table 5.1.

Tree structure			
Symbol	Property	Units	Data type
$T.w$	Tree Width	meters m	1 double value
$T.h$	Tree Height	meters m	1 double value
$T.d$	Tree Depth	meters m	1 double value
$T.v$	Tree volume	cubic meters m^3	1 double value
$T.l$	Tree levels	dimensionless	1 Unsigned integer (uint32) value
$T.s$	Sub - tree	dimensionless	1 uint32 value

Table 5.1: Tree structure

Width, height and depth define the size of the full tree in the three-dimensional spaces (x, y, z respectively). $T.l$ gives the maximum number of levels the tree has grown and $T.s$ is used to track and count sub-trees.

Each node n forming the octree is stored in a structure. The definition of the node structure is presented in Table 5.2.

Node structure			
Symbol	Property	Units	Data type
$n.p$	Spatial position of the node centre	meters m	3 double values (x,y,z coordinates) (pointd)
$n.M$	Individual Morton Key value	dimensionless	1 uint32
$n.c$	Child flag with the quantity and position of the node children	dimensionless	1 Char
$n.d$	Depth of node	dimensionless	1 integer
$n.lc$	List of pointers to each existent child	dimensionless	8 integers values
$n.lp$	Pointer to parent	dimensionless	1 integer value
$n.li$	Array with list of particles	dimensionless	50 uint32 values

Table 5.2: Node structure

The value $n.M$ gives the spatial location (Morton key) of the node. In Section 5.3.1 an in depth explanation of the Morton key and how it is used in the SPH implementation will be presented. The child flag $n.c$ stores how many children the node has. The nodes depth value $n.d$ stores the level in which the node is found in the tree, $n.lc$ contains memory pointers to all existent children. $n.lp$ is a pointer to the parent node, and $n.li$ is an array with the index values of all the particles belonging to that node. If $n.li$ is not empty then the node is a leaf node and it will not contain any children.

Figure 5.8 presents an example of a structure between a parent with a full list of leaf children, the node structure is the same presented on Figure 5.6. In the case of SPH the leaf nodes have a pointer to the memory position of a linked list containing all the particles that are in the volume defined by the leaf node.

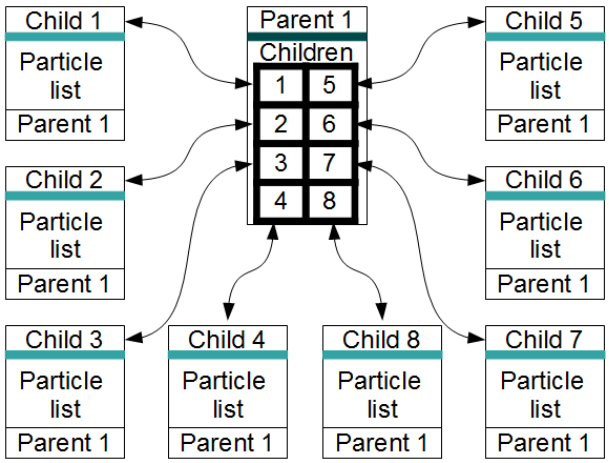


Figure 5.8: **Octant node of a particle system** - The leaf nodes contain a list of all the particles that belong to that volume, in the SPH implementation the nodes contains the particle list belonging to the volume represented by the node.

5.3.1 Smoothing Particle Hydrodynamics Morton key

The method used to obtain the Morton key for a tree node and a particle is similar, the only differences are that in the case of the node the centre position $n.p$ is used for the computational physics calculations and for the particle the spacial position of the sph particle i $\mathbf{p_i}$ is used. The explanation of how to calculate the Morton key will be done using an example particle.

To build the Morton key the values of the particle position $\mathbf{p_i}$ on the three-dimensional space $\mathbb{R}^3$, with components $\mathbf{p_i} = (x, y, z)$ where $x, y, z \in \mathbb{R}$ are used, the variable containing the value of $\mathbf{p_i}$ is defined as a double type and to solved the key they need to be mapped to a single integer value with respect to the volume defined by the tree.

The mapping operation consists of obtaining the representation of the three double value components of $(p_i.x, p_i.y, p_i.z)$ using a single 32 (or 64) bits value this binary value is assigned to the object as its own Morton key.

The first step of the mapping involves normalising the particle positions with respect to the tree size. In the SPH algorithm all particles have positive position values and the tree size is equal to the problem volume size. The problem volume size is defined as the necessary volume space that is big enough to contain all the particles during a complete simulation.

$$\begin{aligned} \overline{p_i.x} &= \frac{p_i.x}{tree.w} \\ \overline{p_i.y} &= \frac{p_i.y}{tree.h} \\ \overline{p_i.z} &= \frac{p_i.z}{tree.d} \end{aligned} \tag{5.4}$$

After normalisation the double values $\overline{\mathbf{p_i}}$ are obtained for all the coordinate points. A right shift by 10 bits is used to pass the values to integer representation.

$$\begin{aligned} pi_i.x &= \overline{p.x} \times 0 \times 400 \\ pi_i.y &= \overline{p.y} \times 0 \times 400 \\ pi_i.z &= \overline{p.z} \times 0 \times 400 \end{aligned} \tag{5.5}$$

The values $\mathbf{pi_i}$ are integer representations of the positions of the particles. The original double interval of values were between $(0, \ldots, 1)$, after multiplying by 0x400 the new value interval will be $(0, \ldots, 1024)$. The integer location of the particle is stored in $\mathbf{pi_i}$. The three 10 bit components $(pi_i.x, pi_i.y, pi_i.z)$ are then combined into a single 32 bit key.

Using the binary representation of $\mathbf{pi_i}$ the Morton key is obtained by interleaving the three components $(pi_i.x, pi_i.y, pi_i.z)$. Figure 5.9 shows an example of the steps needed to calculate a Morton key for a single particle.

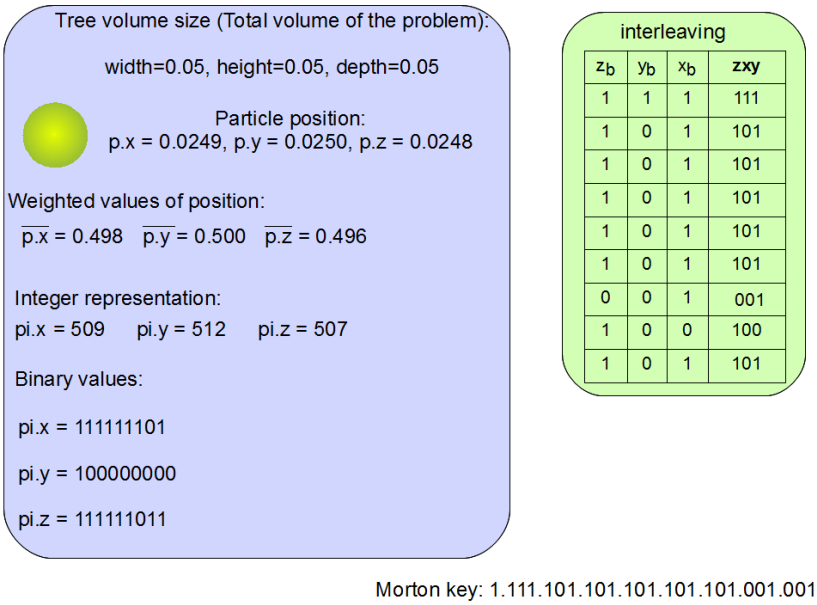


Figure 5.9: **Particle Morton key calculation** - The key is formed of groups of three binary values, the first digit (1) is used as a placeholder bit.

5.3.2 Smoothing Particle Hydrodynamics Morton key octree traversal

To use the key to locate an object in volumetric space we divide it into groups of three bit values (without considering the place holder value). Taking the three most significant bits of the key gives the root's child node the element belongs to (the root's children are the first 8 children from the top node). The next three significant bits will provide the location of the sub-node within the root's child node. Doing this recursively will find the leaf node that the particle belongs to (e.g. see Figure 5.10).

One advantage of using a Morton key is that it provides a method for fast object location search (nodes and particles). All the objects are ordered using a sorting method, in the case of the SPH implementation the Quicksort method [88] is used. After the elements have been sorted the tree traversal can be initiated using the location of the last node visited on a tree traversal operation. The advantage of this method is that it does not require a full tree traversal each time a new operation is performed on the tree.

This method is implemented at each time step of the simulation in order to update the particle tree location and obtain a new data structure list for the particle system.

Eventually, the leaves of the tree contain the particles situated in the specific leaf child volume (in our case the volume takes the form of a cube).

The octree system allows for different resolution representations, a parent node with a set of leaf children can be used as the representation of the particles contained on its children. When this operation is made the children are removed from the tree and the parent node becomes a leaf node. The particles

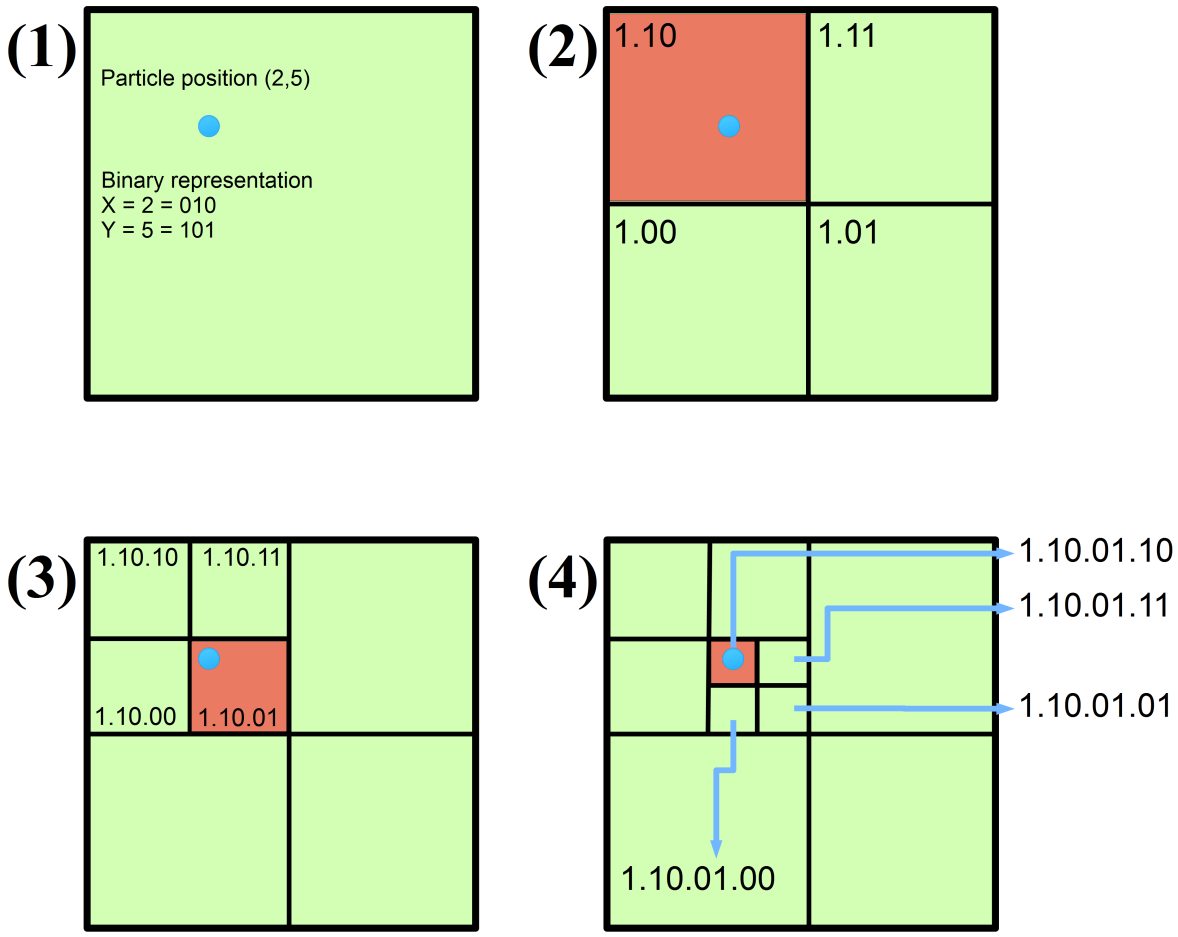


Figure 5.10: **Morton key traverse** - Example of locating a particle in a tree using a Morton key value of 1.10.01.10. A quadtree is used for a simple representation. The tree structure is traversed (from 1 to 4) using a pair of bits at a time step.

contained on the children nodes are used to define a new particle with mass equal to the total mass of all the particles. The density will be the same and the volume of the new particle is easily obtained using the mass and density values of the new particle.

5.3.3 Parallel Morton key octree

There are different methods that can be used to improve the computational time of trees algorithms. The literature there are different implementations that deal with the problem of generating and/or using on parallel architecture. A non-exhaustive list of the most important approaches is provided below:

- In [8] a multi-thread CPU solution to parallelize the Morton octree is presented. The solution takes advantage of the shared RAM memory capabilities of the CPU allows all nodes on the tree to have access to all the elements on the system struture.

- In [89] a GPU tree is used to model texture. The tree is built on the CPU and is assumed constant throughout the execution of the solution. The GPU parallel implementation is used only to traverse the tree volume.

- In [90], a GPU quadtree approach makes uses of the lower requirement of memory compared to an octree to predefine the tree structure on the GPU memory. Any tree update still requires the use of the CPU.
- [91] introduces GPU octrees as an optimized Search method. The method focuses on non dynamic trees and does not solve the multy-threads memory access problems on GPUs.
- In [92] Morton hash keys are introduced in order to obtain a linear representation of the elements. The key is used to build the tree on the host (CPU) and the tree traverse is done on the device (GPU). The solution presented in this work required to transfer memory between the CPU and the GPU each time a tree structure update was needed (creation or deletion of nodes).
- [93] describe multiple GPUs implementation of the N-body problem. The octree is solved in multiple GPUs and the method use to operate on the tree is called "tree walk". It should be noticed that this method make uses of multiple-GPUs (GPU server) and each one of them is used to solve part of the structure.

The different systems presented above make use of the GPUs on different ways. The CPU-GPU systems have to deal with memory management. I require to have a GPU solution using the CUDA architecture that is fully implemented on the GPU and reduce the number of transfers to the Host (CPU) in order to reduce the computational time. In the following section a description of the method introduced in this thesis is discusses.

My parallel octree implementation can be divided into two main sections:

(a) Octree construction: In the first time-step a full tree needs to be built taking into consideration the original particle structure.

(b) Octree update: After the first time-step the tree needs to be updated to reflect the changes in the problem structure. Octree updates use the previous time step Octree to construct the new one.

Parallel octree construction

In the parallel implementation of the octree using the GPU parallel architecture a top down tree construction is used. This is due to a memory access problem that occurs when starting at the root and using the Morton key to traverse to the top (leaf). The problem is that the memory handling on the GPU with CUDA architecture allows only serial access to data structures.

The first step is to create the necessary memory for the number of nodes needed for a complete solution for the entire volume. The lowest h value and the total volume of the problem are used to determine the number of levels the octree needs to have. The maximum tree volume $T.v = T.w * T.h * T.d$ is equal to the total volume of the problem. The number of nodes per level is shown in Table 5.3.

The top tree level needed to represent the whole problem Tl (Tl is also the starting point of the parallel tree construction) depends on the relation between the total volume size $T.v$ and the volume of the effective size $k * h_i$ of the particles i. Tl is the level in which the node size is equal or less than the $k * h_i$ size. The node half width size for each level is obtained using Equation 5.6. The structure defining the properties of Tl is shown in Table 5.4.

$$Node\_HalfWidth = \frac{T.w}{2^{n.d+1}} \tag{5.6}$$

The half width units are in meters m. In Table 5.5 an example of the node half width value when the root node size (full width) is equal to 1 m for different tree levels is shown.

Number of nodes per level	
Tree level	Number of nodes
0	1
1	8
2	64
3	512
4	4,096
5	32,768
6	262,144
7	2,097,152
8	16,777,216
9	134,217,728
10	1,073,741,824

Table 5.3: Number of nodes per level. Using the complete full top level nodes will always present a memory management problem.

Top tree level structure			
Symbol	Property	Units	Data type
$Tl.Nnumberx$	Number of nodes on the x dimension of the level	dimensionless	1 integer value
$Tl.Nnumbery$	Number of nodes on the y dimension of the level	dimensionless	1 integer value
$Tl.Nnumberz$	Number of nodes on the z dimension of the level	dimensionless	1 integer value

Table 5.4: Top Level of interest, this level is the starting point of parallel tree construction, it keeps track of the nodes that are available on memory

Number of nodes per level	
Tree level	Half width in m
0	0.50000
1	0.25000
2	0.12500
3	0.06250
4	0.03125
5	0.01563
6	0.00781
7	0.00391
8	0.00195
9	0.00098
10	0.00049

Table 5.5: Half width size of the nodes at different tree levels.

After selecting the node level in which the node half width is equal or less than the minimum $k*hsml_i$ the selection of the necessary number of nodes per level follows the conditions:

(a) Number of particles is less than the number of nodes in the level: If this condition is true then the maximum possible number of nodes required to represent all the particles is equal to the number of particles.

(b) Number of particles is greater than the number of nodes in the level: If this condition is true then the maximum possible number of nodes required to represent all the particles is equal to the maximum number of nodes of the level.

These are important conditions for our SPH computational implementation since there are simulations that require that $T.v >> k * h_i$ since the particles can displace inside the total volume. If the number of particles is not greater than $200,000$ then even if using tree levels greater than level 5 it is not necessary to allocated memory to accommodate the maximum number of nodes per level.

Using these considerations a memory node pool stack is created, the pool stack [94, Chapter 10] contains the total number *pool.s* of node structures required to maximise the possible tree representations for the specific problem. At the beginning of the process the pool stack nodes are not assigned to specific positions on the octree, they are free to be assigned to any position on the tree as the solution needs them.

In the parallel implementation the tree is built from the top level to the bottom root node. The first step is to build "ghost nodes" *gn*. The number of initial *gn* needed is equal to the number of nodes of the Tl level. The *gn* are used to represent a full tree level. To facilitate the necessary mathematical operations the maximum number of nodes are represented in each one of the dimensions (x, y, z) in Table 5.6.

Number of nodes per level				
Tree level	**Total**	Number of nodes in x	Number of nodes in y	Number of nodes in z
0	**1**	1	1	1
1	**8**	2	2	2
2	**64**	4	4	4
3	**512**	8	8	8
4	**4,096**	16	16	16
5	**32,768**	32	32	32
6	**262,144**	64	64	64
7	**2,097,152**	128	128	128
8	**16,777,216**	256	256	256
9	**134,217,728**	512	512	512
10	**1,073,741,824**	1024	1024	1024

Table 5.6: Number of nodes per level. The octree values are cubic so the values for all dimensions on the level are equal.

The *gn* solution is based on the block and thread index on the CUDA device, a virtual grid is created representing all possible node positions on the level Tl. In the CUDA kernel the index defines each one of this position and they are obtained using Equation 5.7.

$$\begin{aligned} gn.x &= blockIdx.x * blockDim.x \\ gn.y &= blockIdx.y * blockDim.y \\ gn.z &= treadIdx.x \end{aligned} \tag{5.7}$$

Where $blockIdx.x$ goes from 0 to $gridDim.x - 1$, $blockIdx.y$ goes from 0 to $gridDim.y - 1$ and $treadIdx.x$ interval is from 0 to $blockDim.x - 1$. Each interval is defined using the Tl number of nodes as per dimension Equation 5.8.

$$\begin{aligned} gridDim.x &= Tl.Nnumberx \\ gridDim.y &= Tl.Nnumbery \\ blockDim.x &= Tl.Nnumberz \end{aligned} \tag{5.8}$$

Each position of the possible gn is obtained using the internal CUDA thread loop. Using all possible combinations of blocks and threads the full tree level volume can be visited once. An example of a complete gn level is shown in Figure 5.11.

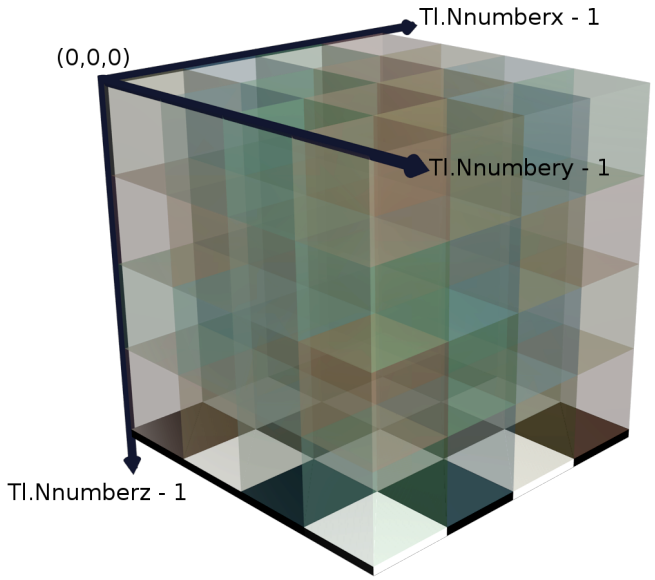


Figure 5.11: **Example of ghost nodes** - The figure shows the ghost nodes for tree level 2. The gn nodes are used to build the tree in the CUDA architecture.

The index for a gn is obtained using Equation 5.8, a pseudo-code algorithm of the parallel call and kernel is shown in Algorithm 5.1.

gn are virtual nodes that are not saved to memory. Each time the CUDA kernel runs an individual thread a different gn is obtained, after the Morton key for the gn, $gn.M$, is obtained the key is used to find all the particles contained in the ghost node (Equation 5.9), the CUDA kernel compares the value of the ghost node Morton key $gn.M$ against the Morton key of the particles ($particle_i.M$).

$$\begin{aligned} Maskindex &= 10 - Tl \\ particle_i.M\_part &= (particle_i.M \wedge (\neg MortonMask\,[Maskindex])) \\ &>> ((Maskindex) * 3) \\ keyxor &= \neg\,(gn.M \oplus particle_i.M\_part) \end{aligned} \tag{5.9}$$

Algorithm 5.1: The ghost node algorithm is used to create a virtual node using the parallel system on the CUDA architecture and the tree information

```
__global__ void TreeBuildKernel(){
  //$gn$ position:
  int x = __mul24(blockIdx.x,blockDim.x);
  int y = __mul24(blockIdx.y,blockDim.y);
  int z = threadIdx.x

  float xn,yn,zn;

  //function to normalize x,y and z with respect to the tree size
  tree_norm(x,y,z, T);

  //Morton key function
  gn.M = morton_key(xn,yn,zn);
  //gn.M is the morton key of gn on the position defined by blockIdx.x, blockIdx.y
     and threadIdx.x.

  //search for particles
  search_p(gn.M, particle_Mkey_list);

  //node update, if a particle is found in the volume defined by gn.M, then the
     algorithm either create a new tree node or if a tree node exist on the same
     position updates the particle list of the tree node.
  node_update(particle_list);

}

void parallelTreeBuild(){
  //threads per block
  dim3 dimBlock(Tl.Nnumberz,1,1);
  //number of blocks
  dim3 dimGrid (Tl.Nnumberx,Tl.Nnumbery,1);

  TreeBuildKernel<<dimGrid, dimBlock>>();
  }
```

$MortonMask$ is a set of binary values that is used to select the appropriate part of the Morton key for the Tl level of the gn nodes. The values of $MortonMask$ are presented in Table 5.7.

If $keyxor = 11111111111111111111111111111111$ then the particle p_i belongs to gn, gn will then either:

(a) If tree node with same Morton key exists: Update the particle list of the tree node.

(b) If tree node with same Morton key does not exist: gs will use the pool stack of nodes to create a new tree node. This will be solved using a CUDA atomic operation [95].

When a gn creates a new node it will pull a node structure from the pool stack, while this operation is being performed the rest of the CUDA threads will need to wait before accessing the pool. This is because the stack will be busy assigning the node memory location to the thread.

These types of operation are solved in CUDA with an atomic operation. Figure 5.12 shows an example of two threads making a memory request for a node structure at the same time. The atomic operation

Binary mask value per level	
Maskindex	MortonMask binary value
0	00000000000000000000000000000000
1	00000000000000000000000000000111
2	00000000000000000000000000111111
3	00000000000000000000000111111111
4	00000000000000000000111111111111
5	00000000000000000111111111111111
6	00000000000000111111111111111111
7	00000000001111111111111111111111
8	00000011111111111111111111111111
9	00001111111111111111111111111111
10	01111111111111111111111111111111

Table 5.7: Binary mask values used to select the part of interest from of the Morton key.

provides a system that will assign the first node in the pool to the first thread while the second thread waits. When the pool is free again, the second thread will obtain their unique node.

After a tree node has been related to a gn position, subsequent updates of the particle list for that node do not need any atomic operations. Looking for particles per gn is a serial process.

When all the parallel threads are finished, the full top tree level will be defined. The next step is to create a set of gn for the level $Tl - 1$, the new gn will not work on the particle system, instead they will search to find any tree nodes (the nodes created in the last step) that exist in their volume of influence.

The same parallel process will be followed and the new level of tree nodes will be obtained. This method is followed until the root level is reached and the full tree definition for the problem is obtained.

Parallel octree update

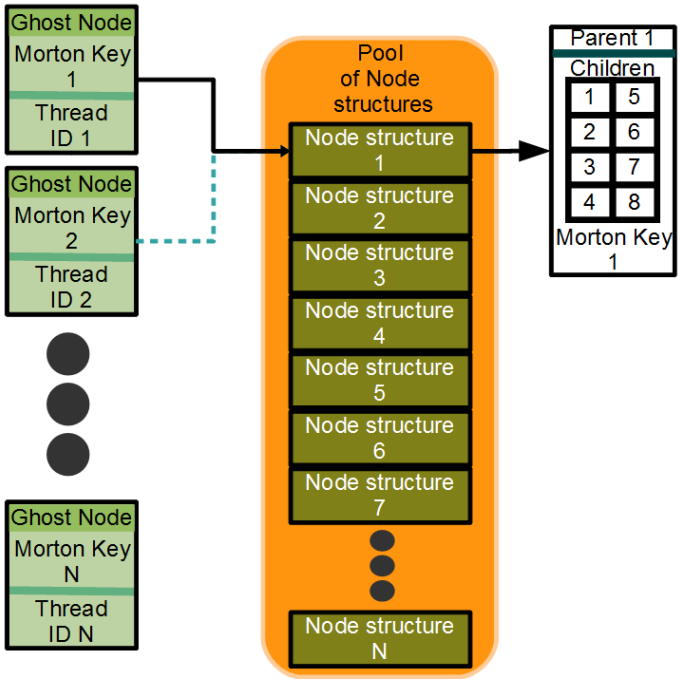
The tree construction is only performed prior to the first time step of the simulation. After the tree has been obtained the following updates use a different technique, each time the tree needs to be updated due to particle movement the tree defined in time $dt - 1$ will be used.

The technique for updating the tree is similar to the serial process presented in Section 5.3.2. The only difference is that in the parallel process if two particles need to create a new node, they will obtain the node structure memory from the node pool stack and the atomic operation procedure to create new nodes will be followed

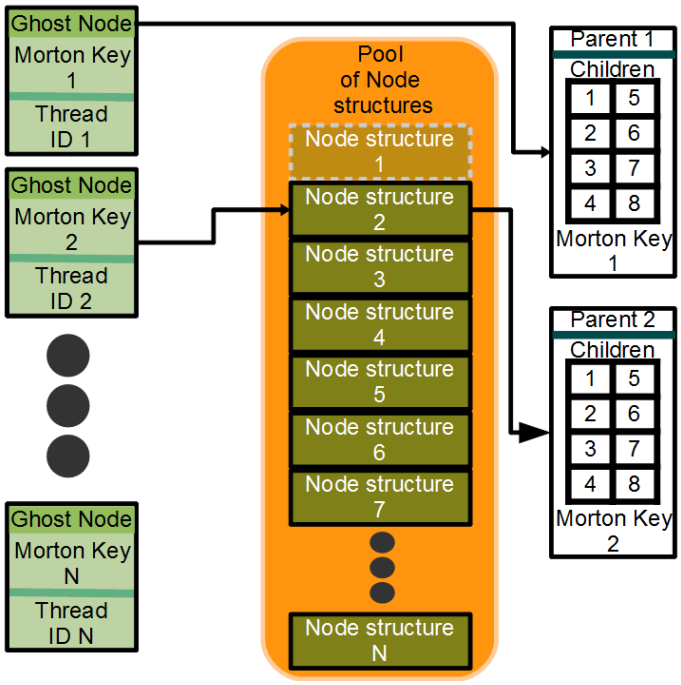
Deleting nodes from the tree can be performed in a parallel scheme. Each time a particle moves, a parallel kernel will remove the focus particle from the node particle list (The maximum number of particles per node is 50).

The particle list structure consists of two values, the first one is a boolean value that keeps track of the status of each individual index (from 0 to 49). If the value is 0 the list index is empty and can be used, the second value is an $uint32$ with the particle index value. Each time a particle list is removed the boolean value is set to 0 and the index value is cleared.

A second parallel kernel checks each node and clears all the nodes in which the 50 index values are equal to 0, the node is inserted back into the pool stack and is free to be used again. Because this is an operation on the pool stack the process is solved by an atomic operation.



(a) First node assignment



(b) Second node assignment

Figure 5.12: **Parallel node creation** - the pool is used to obatin the resolution of a simultaneous parallel kernel call. The atomic operation allows serial control over the assignment of free node structures from the pool to different ghost nodes.

5.4 Conclusions

The data structure definition and the specific neighbour relation of the particles presents a complex problem in mesh-free methods. The requirement for time dependent updates required the construction of an algorithm that reduced the computational cost when defining the neighbour structures. The use of the following computational methods reduces the computational cost of building the neighbour list:

- Octree
- Morton Keys
- Quicksort Algorithm

Due to the nature of the system an implementation of a parallel solution was possible. The improvement in computational time due to parallelisation of the task overcomes the problems presented by the use of the CUDA architecture parallel process. The most significant problems that the system needs to solve to achieve the fastest computational solution are:

Serial memory access: Access to memory operations (push and pop) on the same stack cannot be parallelised.

Tree parent-child structure: The creation of new children requires the use of a stack pop operation.

Node cleaning: The final step of cleaning the nodes require a push stack operation.

To solve this problem the ghost node method is used. The parallel implementation presents important advantages in the solution due to the ability to perform the following operations in parallel:

Node update: Particle insertion/deletion on tree nodes.

Tree traversal: Particle tree traversal.

Particle update: Time step update of the particle-node relations.

Sections 6.3 and 6.6 present the numerical results of implementing the Tree structure with different parameters and how it compares against the use of the linear search method.

Chapter 6

SMOOTHING PARTICLE HYDRODYNAMICS IMPLEMENTATION RESULTS

6.1 Introduction

In this chapter different tests and results on using the SPH implementation will be presented. The chapter will be divided into the following sections:

- **SPH computational results:** In this section the different computational solutions and results will be presented. A discussion on the computational time under different conditions will be given.
- **SPH physical results:** The behaviour for different physical values and boundary conditions is presented. This section introduces the limitations on different modelling situations and where possible the solution to these problems.
- **SPH memory management:** This last section will discuss the different memory management options that the SPH implementation provides when solving problems that require more memory than the one that is available on the hardware been used.

The following parameters and conditions need to be considered when evaluating the different implementations of the SPH method:

- **Computational time:** As the number of particles increases the time taken by the computational solution to compute a time step increase with it.
- **Memory cost:** The second limit to the number of particle that can be solved is the memory capabilities of the computer running the program.

  If enough memory is available then some modules on the solution can used it to reduced the complexity of operations and reduced the computational time taken by the solution.
- **Accuracy of the solution:** For the numerical solution on physical problems the accuracy of the simulation is of great importance and in many cases it is preferable to increase the memory used and the computational time in order to obtain a more reliable solution.

In the other hand in non physical solutions applications such as computer animation the accuracy of the solution can be decreased in order to improve the computational time taken by the implementation when solving the numerical model.

- **Modular solution:** The SPH implementations is built using different modules. A module is a set of instructions that solved a component of the numerical solution, Figure 6.1 shows a simple diagram of the modules involve on a simple SPH computational solution.

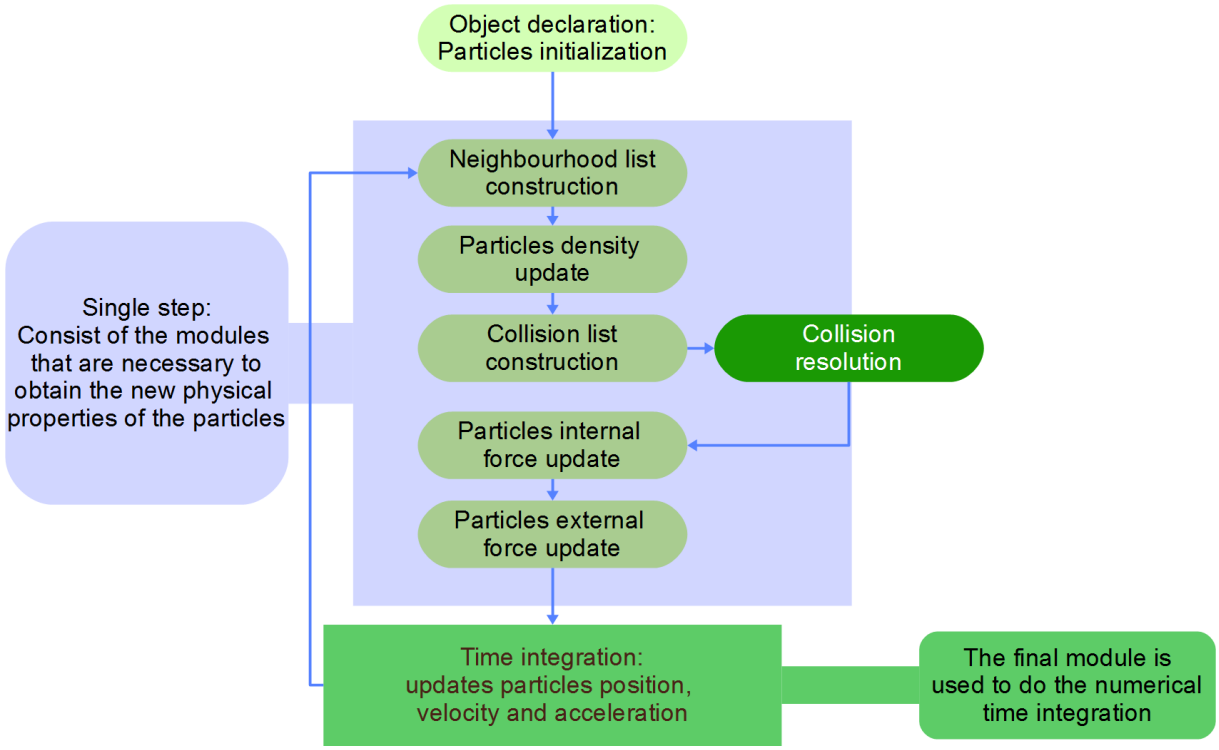


Figure 6.1: **SPH modular implementation** - The SPH implementation built on a set of modules, each module is in charge of the solution of one of the necessary components of the numerical method.

The modules involved in a numerical solution will vary depending on the material type of the solid and/or fluid been solved. Each module has different memory requirement and computational time cost.

When implementing the SPH method on the CUDA architecture it was necessary to provide independent modules. A module is considered independent if all the variables that are necessary for the numerical solution of all the particles involved in the module are defined. This will guarantee that when a particle i is been solved all the values provided by the particles j on the neighbourhood and/or collision list are available and up-to-date.

The independent modules are necessary on the CUDA implementation due to the parallel approach used by the CUDA architecture, in a parallel environment the order in which a particle from the particle system will be solved is unknown so if a CUDA implementation updates a value of a particle i that is needed in the same module by a particle j it is not guarantee that the particle j will have access to the update value.

- **Implementation bias:** The numerical solutions presented on Chapters 3 and 4 were chosen due to the ability to solved different problems based only on the constitutive equation of the material

been solved, methods that require an a priori condition additional to the constitutive equation were avoided in the SPH implementation.

- **Portability:** Portability is also taken into account in the SPH implementation, the serial SPH is portable and can be run on systems that can compile the C++ language (e.g. Windows, Linux, Unix systems). The system does not make use of external libraries to get the computational solution. The 3D display used the OpenGL library to maintain the portability.

  The parallel CUDA implementation requires a compatible GPU (fifth generation onwards) and the ability to compile the CUDA libraries, at this time the implementation has only been tested on a Windows system. The supporting literature for CUDA provides the steps for an implementation on a Linux [96] system but this has not yet been considered.

6.2 Smoothing Particle Hydrodynamics computational results

In this section the different tests and results for the computational implementation are presented. The SPH implementation allows for a set of parameters to be defined before the computational numerical solution is obtained. These parameters are related to the computational numerical solution and not to the physical problem. Changing the value of each one of the parameters will produced a changed on the behaviour of the particles and the final numerical solution.

In order to be able to quantify the changes produced over the numerical solution when the initial parameters of the solution are modified on different problems, three test objects are defined. A visual representation of the test objects is shown in Figure 6.2

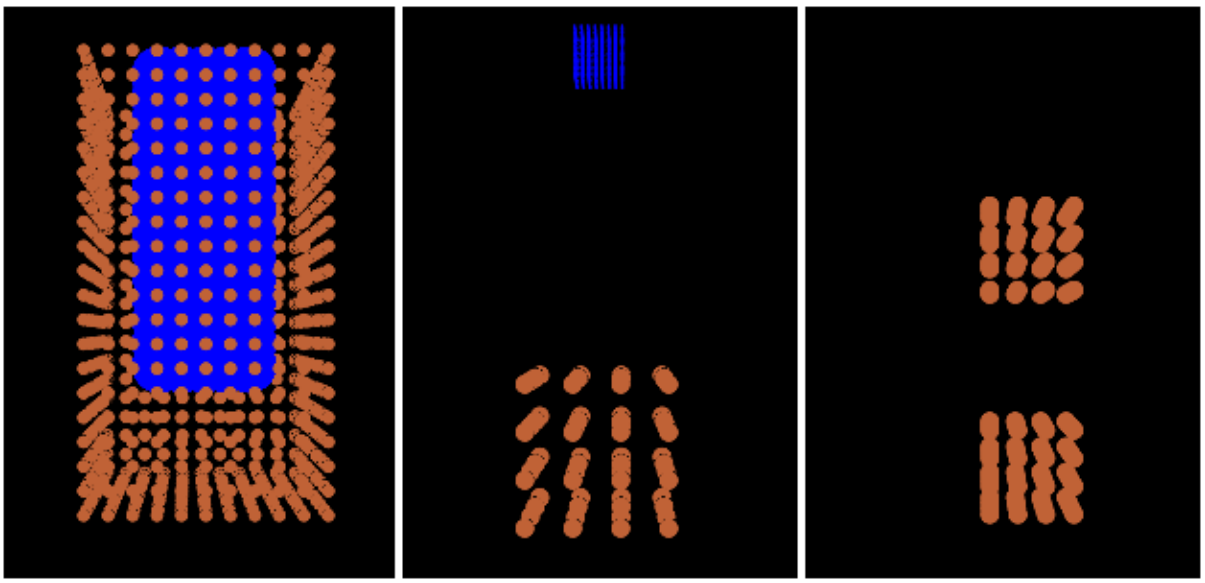


Figure 6.2: **SPH computational test objects - Left**: shows the basic object used for the computational test for fluid-solid interactions when the solid functions as a virtual boundary particle. **Centre**: Test objects for the fluid-solid interaction when solids can have a shape deformation. **Right**: Solid-solid interaction test objects.

The test objects are used to solve solid - internal fluid, solid - fluid and solid - solid interactions. The objects are simple geometric representations that will provide simple solutions allowing us to focus on the effect of changing the initial parameters on the computational solution.

Algorithm 6.1: **Pair-wise search algorithm** - The pair-wise search needs to visit all the particles j with internal index greater than i and test for neighbourhood or collision

```
1  //The first for loop over all the objects on the simulation
2  for (object_index = 0; object_index < total_objects; object_index++){
3    //Second loop visit all the particles that belong to the object
4    for (indexi = particles_list[object_index].ini; indexi < particles_list[
       object_index].last; indexi++){
5      //Test for neighbourhood of particle i and j
6      for (indexj =indexi+1; indexj < particles_list[object_index].last; indexj++){
7        test_add_N(indexi, indexj, hash_table); //call test
8      }7
9      //Also test for collision
10     for (object_index2 = 0; object_index2 < total_objects; object_index2++){
11       if(particles_list[object_index].type == solid && object_index !=
           object_index2  ) || (particles_list[object_index].type == fluid &&
           particles_list[object_index2].type == solid) {
12         //check over all the particles on the object
13         for (indexj = particles_list[object_index2].ini; indexj < (particles_list
             [object_index2].ini + particles_list[object_index2].total)-1; indexj
             ++){
14           //check only for particles $j$ with internal index greater than $i$
15           if(indexj>indexi){
16             test_add_C(indexi, indexj, hash_table); //test for collisions
17           }
18         }
19       }
20     }
21   }
22 }
```

6.3 Neighbourhood and collision search volume

The first computational test objective is to describe the effect of the kernel size that defines the influence volume of a particles i over the particle j.

During the computational numerical solution there are two types of list that need to be created and update during the simulation; the neighbourhood list and the collision list.

To improve the performances of the SPH implementation In Section 5.3 the octree solution of the neighbourhood search was presented. The advantage of the octree search engine depends on:

- **Number of particles:** The Linear Search algorithm used the all-pair search method (e.g. see Section 5.2.1) to find the neighbourhood and the collision lists. The all-pair search approach has a complexity of $O\left(N^2\right)$ so as the number of particles increased the effectiveness of the method is reduced drastically. The Algorithm 6.1 shows the all-pair search pseudo-code for a list of particles.

  The functions *test_add_N* and *test_add_C* are the same for the octree and the linear search, so the only difference is the number of calls to the functions that are required to test all the particles. For the description of the octree implementation see Section 5.3.

- **Volume distribution:** The octree solution depends on the relation between the tree volume size and the particle volume size. This is due to the limitations on the number of levels a tree can have. If the tree volume is much bigger than the objects inside it the chance of all particles belonging to

the same node increases. If all the particles belong to the same node then the tree will not help in discriminating the particle positions and the list search will be solved for all the particles. When this happens the use of the octree solution will be worse than the linear search since it will effectively be a linear search with tree traversal.

- **Octree kernel search:** Besides the internal list kernel, the octree also has a kernel that provides a discriminant of how far apart the centre of two tree nodes can be and still belong to the same interacting volume during the tree traversal (e.g. see Section 5.2.4). If the value is too small then the lists will not contain sufficient particles, if it is too big then the number of nodes visited to get the lists will increase beyond the necessary distances.

6.3.1 Number of particles results

The initial parameters for the number of particles test are defined in Table 6.1:

Test setup		
Property	Value	Units
Total number of particles	150000 elements	dimensionless
Tree width	0.002	m
Tree height	0.002	m
Tree depth	0.002	m
Differential time step dt	0.00005	s
k	1.0	dimensionless
Ω	0.05	dimensionless
Collision scale factor	1.0	dimensionless
e	0.2	dimensionless
Solid cube size	$(x : 0.00050, y : 0.00050, z : 0.00050)$	m
Fluid cube size	$(x : 0.00015, y : 0.00015, z : 0.00015)$	m
Solid cube dimensions	$(x : 7, y : 7, z : 7)$	dimensionless
Fluid cube dimensions	from $(x : 3, y : 3, z : 3)$ to $(x : 30, y : 30, z : 30)$	dimensionless

Table 6.1: Test setup for the comparison between the octree and the linear search of the lists to solve the neighbourhood and collisions.

The "Total number of particles" determines how many structures defining the properties of the particles are initialised. The SPH implementation for the test used $3,875,338kB$ of memory for $150,000$ particles.

In the following section the relation between problem size and tree size will be described. The time step Δt is fixed during the simulation and is chosen to be $50\mu s$ in order to have a consistent behaviour in all the tests independently of the number of particles and their velocity.

k is the scale factor for the particle domain (e.g. see Section 2.3), Ω is the constant used in the anti-penetration force (e.g. see Section 4.5.2), a collision scale factor is used in some experiments to change the influence volume of the particles during the collision responses and e is the impulse constant (e.g. see Section 4.5.2).

The objects size is kept constant for the test, the variable is the number of particles used to discretised the object volume. As the number of particles increases the computational time increases for both the linear and the octree solutions.

Both the linear search algorithm and the octree algorithm used the Morton key spatial description (e.g. see Section 5.2.5) as a flag that tracks the changes to the particle. In the SPH implementation there

is a new data structure that maintains the information about the particles with respect to the problem space, there are three discriminants that help with the computational time:

- Neighbourhood flag: This flag tracks the particles which needs to update their neighbourhood list. The flag uses the Morton key resolution. If the Morton key value of a particle changes the particle is no longer in the same spatial position. The resolution for the test is the full Morton key (30 values for 32 bit systems or 60 values for 64 bit systems). If the problem does not require a full spatial resolution the number of triplets used to determine if a particle has changed can be reduced to the amount needed to represent the spatial position of the particles.

- Collision flag: This flag is activated using the same technique as above. The collision flag tracks the particles that undergo a collision. The number of triplets used for the flags may be different than the one used for the neighbourhood, for the computational test presented on this chapter the full resolution is used.

- Node flag: In the case of the octree solution there is also a flag which determines when a particle has left the child node it belongs too. The flag is use to determine is a particle needs to be deleted from a node and a search of a new node needs to be done.

The first two flags are used in both the Linear Search and the octree search. The node one is only used in the octree for the tree update. Without the flag the list search would need to solved for all particles at each time steps.

As the number of particles increases the ability of the octree to divide the problem volume into subsets results in an increase in the time performance. Table 6.2 shows the relationship between the number of particles and the total time taken for the algorithm to solve 1000 iterations for each case. For both of the solutions, linear and octree, the computational time of all components are measured every 10 time steps. The measurement calculates the time it takes to solve the problem up to that time step.

Number of particle test results			
Test #	Number of particles	Linear solution total time s	Octree solution total time s
1	370	17.2	17.2
2	559	17.7	19.0
3	1072	23.2	22.2
4	2071	30.6	42.3
5	3718	77.8	82.3
6	6175	194.6	138.4
7	9604	442.0	224.2
8	14167	950.1	343.5
9	20026	1882.5	519.2
10	27343	3380.9	745.8

Table 6.2: Computational time of 1000 time steps for the different number of particles tested. The octree implementation using flags outperforms the Linear Search in problems with higher number of particles.

Figure 6.3 shows a plot representation of the results, the octree performance over the Linear Search increases as the number of particles increased.

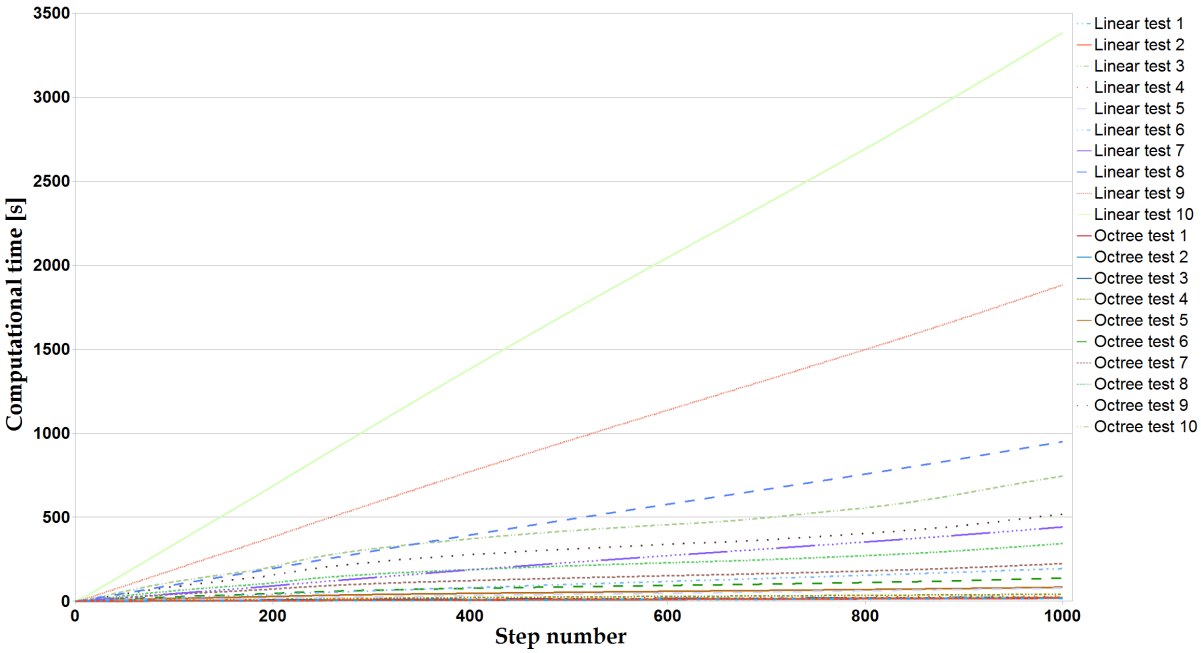


Figure 6.3: **SPH computational particle number test** - The computational time of the simulation increases as the number of particles defining an object increases. The use of the Morton key, Morton key flag system, and the octree structure improves the computational time of the solution.

6.3.2 Octree size - problem volume size relation

The second important consideration when setting up a numerical solution of a model is the relation between the tree volume size and the problem volume size, the problem volume size is defined as the necessary volume that the problem requires to represent the particles positions during the simulation. The following factors need to be considered:

- Tree size and particle resolution: The Morton key used to provide the tree structure has a limit of 10 levels on a 32 bit system and 20 on a 64 bit one. Each level is related to a digit on the numerical position of the particle. As the differences between particle resolution and tree size increases the number of particles contained in the leaf levels increases. This is caused by the lack of available information contained in the Morton key to discriminate between particle positions after the limit of tree levels (10 levels on a 32 bit system and 20 on a 64 bit one).

- Number of tree levels: When applying the tree implementation on different 3D objects (fluid or solids) another problem may appear. The objects' volume can be divided into more levels than what it is necessary. Tree depth is obtained with the algorithm shown in Algorithm 6.2. The number of tree levels is obtained per object, since different objects in the same simulation can have different particle resolution.

- Tree of trees: The problems discussed in the the two previous points can usually be solved by defining proper tree size and limits. In some simulations the problem size - particle resolution can

Algorithm 6.2: Obtaining the leaf depth value for an object on a SPH simulation

```
1   //First obtain the smallest $kh$ value of the object
2   psystem_partobj[object_index].min_hk = min(particle_hsml[object_index])*k*2.0;
3   //Obtain the leaf limit for the object using the minimum $kh$
4   psystem_partobj[object_index].leaf_limit = ceil(
5                 (log(psystem_Tree.width)-log((double)(psystem_partobj[
                      object_index].min_hsmlk)))/log(2.0)+1.0
6                 );
7   //Check if the leaf depth value is inside the tree
8   if((psystem_partobj[object_index].leaf_limit-psystem_Tree.levels)-1 <= 0){
9   psystem_partobj[object_index].leaf_limit = psystem_Tree.levels + 2;
10  }
11  if((psystem_partobj[object_index].leaf_limit-psystem_Tree.levels)-1 >= 9){
12  psystem_partobj[object_index].leaf_limit = psystem_Tree.levels + 8;
13  }
```

be bigger than an efficient tree limit, there are two variables to defined if a tree will have an efficient performance.

- Number of particles per node: If the volume size of the nodes is much bigger than the particle resolution, the nodes will contain a large amount of particles making the internal search on each node more computational time expensive.
- Problem volume size relation: If the size of a problem volume is much bigger than the objects volume that are contained in it then the tree size will need to subdivided the larger volume and at the same time produce a large quantities of levels to solve the particle resolution of the objects.

When a good performance can not be achieved with one tree, a tree containing trees is defined. The root tree (tree containing the trees) contains pointers to the top node of the internal trees in its leaf nodes. This provide the system with more depth levels and a way to traverse bigger volumes. Each tree has its own Morton key assigned to the nodes, so a collection of two trees will have access to 20 levels of resolution in a 32 bit system.

To improve the computation time the root tree is kept in memory and is not updated, this is possible because the internal trees do not change their spatial position.

To test the efficiency of the octree two different setups are used, the first one will test the relation between the tree size and the problem volume size, in the second test the tree of trees is used.

Octree size test

In the first test the particle system is kept constant while the tree size varies from a tree with size $[0.011, 0.011, 0.011]m$ to a tree with size $[0.029, 0.029, 0.029]m$. The initial particle setup is defined in Table 6.3:

Test setup		
Property	Value	Units
Total particle memory elements size	150000 elements	dimensionless
Differential time step dt	0.00005	s
k	1.0	dimensionless
Ω	0.05	dimensionless
Collision scale factor	1.0	dimensionless
e	0.2	dimensionless
Solid cube size	$(x : 0.00050, y : 0.00050, z : 0.00050)$	m
Fluid cube size	$(x : 0.00015, y : 0.00015, z : 0.00015)$	m
Solid cube dimensions	$(x : 7, y : 7, z : 7)$	dimensionless
Fluid cube dimensions	$(x : 8, y : 8, z : 8)$	dimensionless

Table 6.3: Test setup for the octree size test. The particle setup is kept constant and only the Tree size is modified.

The results for the different tree size configurations are shown in Figure 6.4.

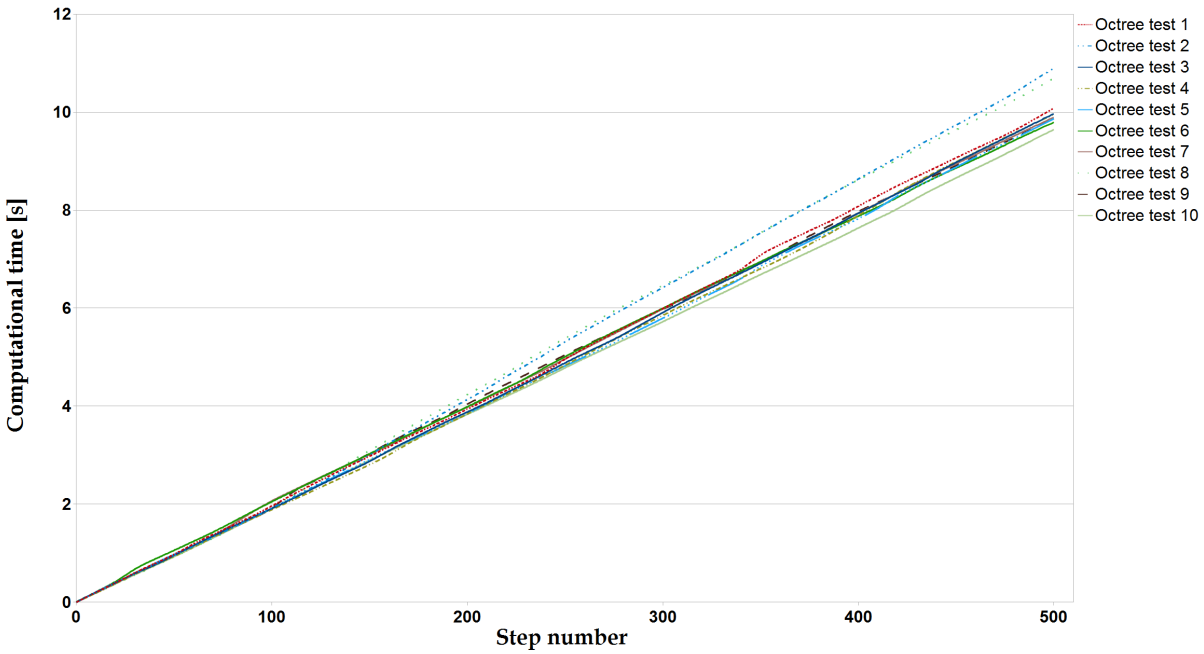


Figure 6.4: **SPH computational tree size test** - The computational time is maintained as the tree size grows. This is due to the Morton key approach to the search of nodes operations, the key restrict the search volume and provide with direct access to the nodes.

Table 6.4 summarises the results of the test.

The average computational time value for this test is 10.26 with a standard deviation of 0.53, the computational time taken by the different tree size is maintain in the same range due to the used of

Octree size test results		
Test #	Tree size m	Octree solution total time s
1	0.002	10.08
2	0.004	10.89
3	0.006	9.96
4	0.008	9.955
5	0.010	9.865
6	0.012	9.786
7	0.014	9.889
8	0.016	10.68
9	0.018	9.89
10	0.020	9.639

Table 6.4: Computational time for the octree size test for 500 time steps. The octree implementation depends on its initial definition

the Morton key approach, the Morton key is used to reduce the volume search and to obtain a quick discriminant of which nodes on the tree have leaf on their branch.

Tree of trees

The following test uses the Tree of trees structure approach. The test is run using four different tree structures.

- The first one is the basic single tree structure, the single tree have a root node from which the tree grows.
- The second one is a two level tree of trees structure. This structure provides a top root tree with eight leaf children from which eight trees can be created.
- The third one is a tree of trees with three levels, the total number of leaf nodes from which the internal trees can grow is 64.
- The fourth one is a four level tree of trees, the root tree has 512 children from which trees can grow.

The tree of trees approach provides with a method that can solved models were the volume problem is several orders of magnitude bigger than the particle resolutions.

The test consists on analysing the computational time taken to solve the same amount of particles for the different tree of trees configuration. The test conditions are the same that the ones given in Table 6.3, the only changed is the size of the fluid cube dimensions $(x : 10, y : 10, z : 10)$.

On table 6.5 is presented the computational time taken for the different configurations to solved 1000 time steps. Figure 6.5 shows a plot of the results.

The performance when using the tree of trees approach depends on the original number of particles and the relation between the problem volume size and the particles volume resolution.

6.3.3 Octree, collision and neighbourhood kernel size

As presented in Chapter 2, the scale factor for the particle domain k defines the surrounding volume of a particle i that will be used to obtain the neighbourhood list from the set of particles j, the SPH

Octree Tree collection test results		
Test #	Main Tree levels	Octree solution total time s
1	0	24.58
2	1	24.42
3	2	27.86
4	3	36.42

Table 6.5: Computational time for different levels of tree structures for 1000 time steps.

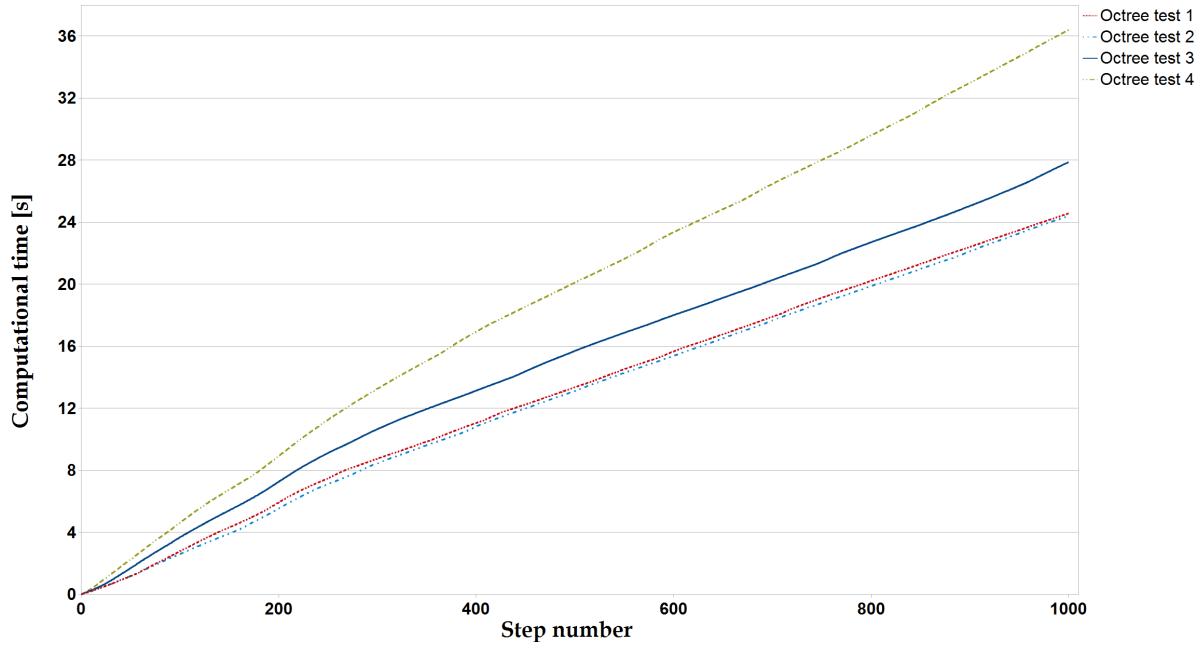


Figure 6.5: **SPH computational tree of trees test** - The use of the four level tree of trees with 512 leaf nodes increments the final computational time require to solve 1000 time steps by 11.84 seconds. The difference between a single tree and a second level is minimal since the only differences is the top node calculation.

implementation also makes use of two other scale factors, one for the collision list and one for the tree node search limits (e.g. see Section 5.2.4).

The three scale factors affect the computational time needed to solve the problem and the physical result of the simulation. This section will focus on the computational time.

As the scale factors for neighbourhood and collision are increase the number of pairs (for the particle domain k and the collision) that need to be solved by the rest of the implementation also increase.

The tree node search limit controls how many nodes need to be visited while the search for pairs and collisions is performed. In most solutions the tree scale factor is maintained close to the k factor value since looking futher away from it will result in particles too far away from the focus particle i.

In summary, the following effects will be caused by changing the different scale factors values:

- Neighbourhood search scale factor (k): This factor is used in conjunction with the particles properties to determine if two of them are neighbours (e.g. see Algorithm 6.3).

  The factor k determines how far a particle j can be from the focus particle i to be considered part

Algorithm 6.3: Neighbour search on the Tree traversal iteration.

```
1  void test_add_N(i, j){
2    //distances between particles
3    driac = distances_oct(i, j);
4    //average of particles h values
5    mean_h = (particle_h[i]+particle_h[j])/2.0;
6    //The scale factor is used to defined the interaction volume of the particles
       scale h
7    scal_h=k*mean_h;
8    //square root
9    r=sqrt(driac);
10   //if distances is less than the scaled h
11   if(r<scal_h){
12     //the particles are neighbour
13     add_pair_ij(i, j);
14   }
15 }
```

Algorithm 6.4: Collision search on the Tree traversal iteration.

```
1  void test_add_N(i, j){
2    //distances between particles
3    driac = distances_oct(i, j);
4    //average of particles h values
5    mean_h = (particle_h[i]+particle_h[j])/2.0;
6    //The scale factor is used to defined the interaction volume of the particles
       scale h
7    scal_h=kc*mean_h;
8    //square root
9    r=sqrt(driac);
10   //if distances is less than the scaled h
11   if(r<scal_h){
12     //the particles are neighbour
13     add_col_ij(i, j);
14   }
15 }
```

of the neighbourhood. The two extremes cases are when the value of k is equal or greater than the focus object so all particles j belong to the neighbourhood of i and when $k = 0$ so each particle is solved as an individual object without any internal forces interaction.

- Collision search scale factor (k_c): This factor is used in conjunction with the particles' properties to determine if two of them are colliding during the current dt (e.g. see Algorithm 6.4).

  The factor k_c determines how far a particle j can be from the focus particle i for them to collide. Similarly to the neighbourhood procedure, the extreme cases are when k_c is too big or is 0. If k_c is too big then the particles will collide even if their defined volumes are not interacting, if it is too small the collision could occur when the particles have already inter-penetrated each other, and for 0 no collision is considered.

- Tree search scale factor (Ts): Increasing this value changes the number of nodes traversed by the

Algorithm 6.5: Tree traversal to build the neighbourhood and collision lists, the factor Ts scales the search volume.

```
1   //Start the search of the list per particle $i$
2   init_search(i){
3     //The kernel search volume is assigned per particle. Its sizes depend on Ts
          and $h$
4     kernel_particle_search.center = particle_position[i][2];
5     kernel_particle_search.radius   = Ts*particle_h[i];
6     //The tree traversal is started at the top node of the internal tree
7     Node* pTree_top = pTree_Itop;
8     tree_traversal(pTree_top, i);
9   }
10  tree_traversal(pTree_local, i){
11    //For each node we need to check all its children
12    for(child_index=0; child_index<8; child_index++){
13      if(child_index){
14        pChild_local = pTree_local->ChildP[child_index]; //child node
15        //check if the centre distances from the focus kernel volume to the node
              is lest than their sum radius
16        if(dist_centres < sum_radius){
17          //check if nodes have particles
18          if(g_slist_length(pChild_local->par_list)!=0){
19            //test for all the particles $j$
20            for (all_particle_on_list){
21              j =((key_Ind*)g_slist_nth(pChild_local->par_list, list_index)->data
                    )->P_Index;
22              //following the rules of list creation test $j$
23              if(test(i,j)==case1){
24                //$i$ and $j$ need to be tested to see if they are neighbours
25                test_add_N(i, j);
26              }
27              if(test(i,j)==case2){
28                //$i$ and $j$ need to be tested to see if they collide
29                test_add_C(i, j);
30              }
31            }
32          }
33        }
34      }
35    }
36  }
```

particle i (e.g. see Section 5.2.4).

Two operations need to be solved per particle, one for the neighbourhood list and one for the collision list depending on the type of the particle j, the pseudo-code showing this process is shown on Algorithm 6.5.

The worst case scenario is when Ts makes the search greater or equal than the Tree size, this will mean that all the nodes are visited and the algorithm will do as many tests as the linear search algorithm and add to the computational time the cost of creating, updating, and traversing the tree.

- $test\_add\_N(i, j)$ and $test\_add\_C(i, j)$ operations depend on k and k_c, respectively. From the pseudo-code shown in Algorithm 6.5 it can be seen that increasing the value of Ts will increased the number of nodes visited during the traversal and this will increase the number of times the operations $test\_add\_N(i, j)$ and $test\_add\_C(i, j)$ need to be executed.

The test is separated into 3 different runs so the results can be presented per scale factor.

Octree scale factor Ts

The solid - fluid test object (Section 6.2) is used to analyse the effect of changing the tree scale factor while maintaining the rest of the setup constant. Figure 6.6 shows the results of the computational time for different scale factors, in Table 6.6 the resulting values are presented .

octree scale factor size test results		
Test #	Tree scale factor	Octree solution total time s
1	0.0	21.21
2	0.1	22.01
3	0.2	21.77
4	0.3	22.73
5	0.4	23.20
6	0.5	22.74
7	0.6	22.94
8	0.7	24.33
9	0.8	22.92
10	0.9	24.14

Table 6.6: Computational time for the scale factor test of 1000 time steps and 1343 particles. Tests 1 to 4 will return inconsistent physics simulations due to the small kernel size

If the scale factor is too small we will get a faster computational time but the resulting simulation will not follow the rules of SPH (due to the low number of particles per neighbourhood) and the results will be incorrect.

As the scale factor value increases the computational time will also increase. There is a limit when the physical simulation will no longer change, the limit is defined by k, if Ts is bigger than k, all the nodes visited with a distances from the focus particle bigger than k will only contain particles outside the interaction volume and they will not be added to any of the lists.

Figure 6.7 shows the results of some of the simulations. Even in the case were the physic simulation is wrong the SPH implementation continues running, most of the computational time is due to the octree updates and particle list tests.

In most simulations Ts is kept equal to k but in some cases there is a need to check if a particle will interact in the next time step This could be for reasons such as: preventing particle penetration, obtaining force and energy values that could change the object's shape, or keeping a check on energy dissipation.

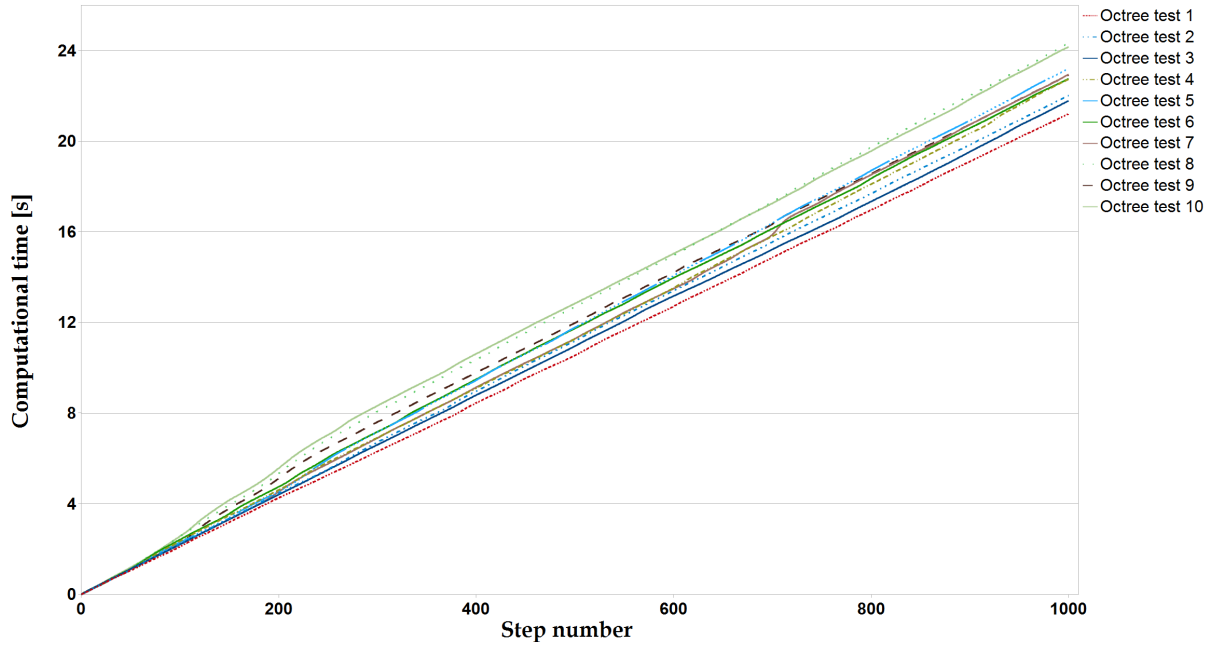


Figure 6.6: **Computational time for the tree scale factor test** - Results of the scale factor test for 1000 time steps. Tests 1 to 4 will not return any consistent physical results. SPH will not have enough particles in the neighbourhood list of each particle to compute the fluid solution, the particles will behave as independent spherical objects.

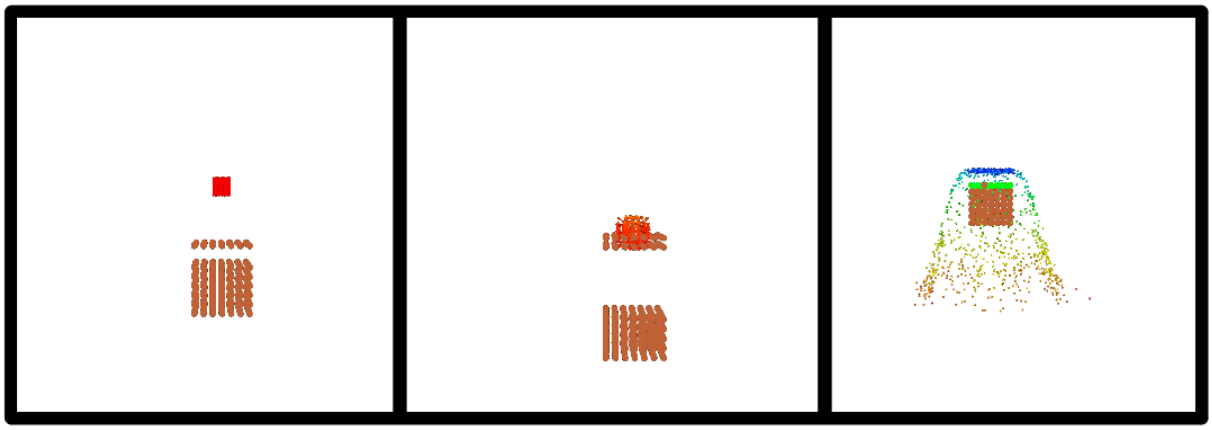


Figure 6.7: **Tree scale factor results** - If the tree scale factor is too small the solution will fail to present proper physical behaviour. Left and centre images show the results for $Ts = 0.0$ and $Ts = 0.3$, these cases present an incorrect solution with zero or small particle interaction. For the solid components the particles will not maintain the solid form (orange box).

Neighbourhood scale factor k

For this test the Ts value is maintain equal to k. Any difference in the computational time of the solution depends only on the initial scale factor k value.

When solving the creation and update of the neighbours list per particle there are two errors that need to be avoided in order for the numerical solution obtained from the simulation to be correct.

- The first one is a neighbours list containing too few particles, if this is the case then the physical values obtained for the particles will not correspond to the macroscopic behaviour of the material.

- Te other error appears if the number of particles in the list is too big then the microscopic changes will be smoothed and the precision on small changes will be lost.

It is common in the literature to define the best value around 35 particles for three-dimensional problems [5], this will ensure that the particle solution contains enough neighbour particles so the value represents the continuum behaviour and that the microscopic changes are not smoothed by operating in a large number of particles.

The results for the test for different k values are shown in Figure 6.8 and Table 6.7.

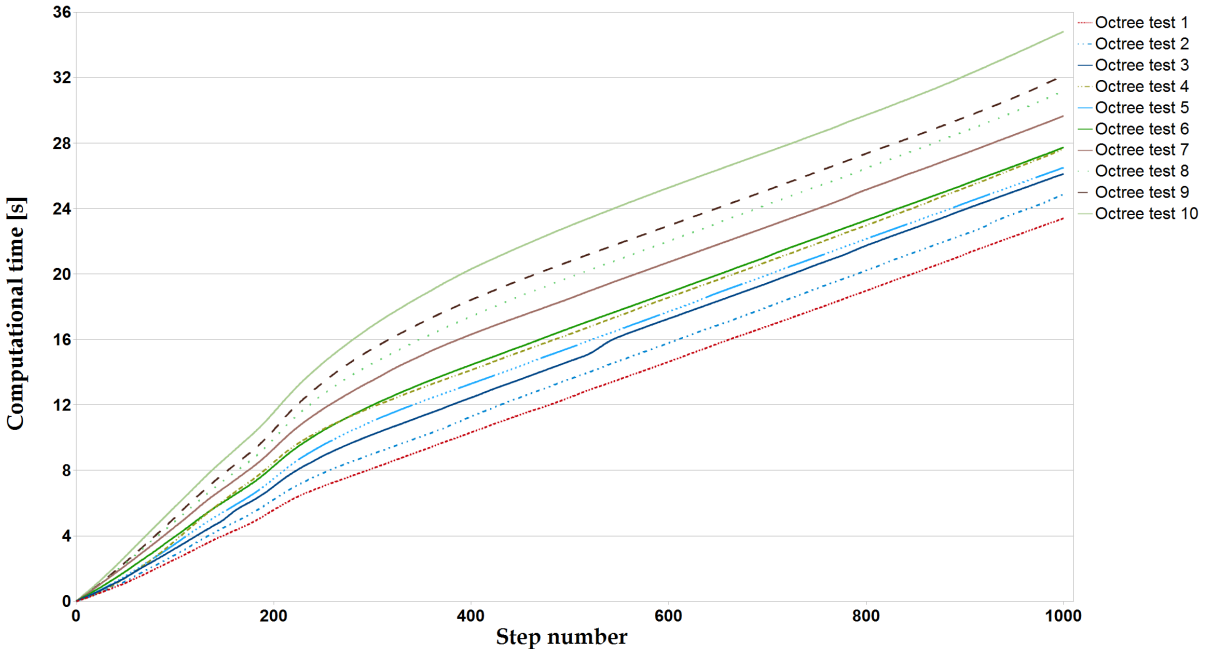


Figure 6.8: **Neighbourhood scale factor results** - The initial value for the scale factor is $k = 0.9$ this ensures that all the simulations will return a correct representation of the flow and collision, after this the size value of k is increased to 1.8, .1 units at a time, as the value of k increases the number of particles in the neighbourhood of all the particles also increases and the computational time to solve a time step is higher

The scale factor affects the computational time since as the value increases more pairs are found after the neighbour test is done. Each new pair will add to the computational time cost since each new pair of particles requires to be solve by the physics simulator components of SPH implementation.

Neighbourhood scale factor size test results		
Test #	Neighbourhood scale factor	Octree solution total time s
1	0.9	23.41
2	1.0	24.87
3	1.1	26.11
4	1.2	27.65
5	1.3	26.51
6	1.4	27.73
7	1.5	29.64
8	1.6	31.18
9	1.7	32.14
10	1.8	34.80

Table 6.7: Computational time for the neighbourhood scale factor test of 1000 time steps.

6.4 Smoothing Particle Hydrodynamics physical results

In the last section was presented the requirements the SPH method must fulfil in order for the numerical solver provides with a correct solution to the models been studied. In this section the test will be focus on the accuracy of the physical solutions and the limitations the method has on different type of materials.

The tests are devised so the number of variables defining the behaviour of the objects is keep to a minimum and the models been solved have simple geometric representations. The test on this section are:

- **Fluid flow down an inclined circular tube:** For this test two objects are defined, a solid with a circular tube shape and a set of fluid particles filling the solid object. The inclination angle on the tube will be used to modified the behaviour of the flow.
- **Tortuosity fluid test:** In the second test the effect of the solid boundary tortuosity over the fluid flow will be tested. To get different values of tortuosity a set of geometrical shapes with different relations between their effective length and its skeleton length are used.
- **Volume fluid test:** The SPH particles can be defined with different mass and density at the beginning of the simulation. For this test the fluid particles are considered to be water with a density of $1000[kg/m^3]$ so the volume of the fluid particles will change depending on the initial particles mass. This test makes use of the Solid - internal fluid objects.
- **Solid deformation:** For the solid deformation one solid object is deformed by a force over one of its surfaces. The test consists of two parts, the first one increases the force magnitude, and in the second test the force is kept constant while the number of particles is changed.
- **Solid collision:** For this test different conditions of collision are solved. To modified the collision behaviour the value of the factor k_c is modified at the beginning of each simulation.

In the fluid particles test a continuous flow solution is defined. In this solution, when a fluid particle leaves the solid object at the outlet it is restarted at a random inlet position while maintaining its velocity. The analysis of the test is done using the volumetric flow rate m^3/s as the flow measurement. This property is selected because any changes in the physical conditions around the fluid will be reflected in the inlet and outlet volumetric flow rate.

For solids, the internal force magnitude N and particle position m are used to measure their deformation, the changes in the position of a particle with respect to its neighbourhood will change its internal force, this makes the analysis of the deformation on a solid object possible.

When studying the fluid flow inside a boundary the volumetric flow Q is used as the principal descriptor of the behaviour of the flow, the value of the volumetric flow is obtained directly from the SPH implementation. The volumetric flow can be derived from the Darcy's law Equation 6.1,

$$Q = \frac{-\kappa A}{\mu \epsilon} \frac{P_{outlet} - P_{inlet}}{L} \tag{6.1}$$

Where ϵ is the porosity, $\mu\ Pa \cdot s$ is the dynamic viscosity of the fluid, $A\ m^2$ the flow area (transversal cut), $P_{outlet} - P_{inlet}$ is the pressure differential between inlet and outlet, $\nabla P\ Pa$, $L\ m$ the length between inlet and outlet, and $\kappa\ m^2$ is the permeability given by:

$$\kappa = \frac{-Q\mu\epsilon}{A} \frac{L}{P_{outlet} - P_{inlet}} \tag{6.2}$$

6.4.1 Fluid flow down an inclined circular tube

To study the fluid flow under different external forces and boundary conditions a simple model of the fluid flow down an inclined circular tube is used. The objective of the test is to defined a simple problem in which the external forces and boundary conditions are the only flow behaviour modifiers. Even more in the inclined circular tube problem the only external force is given by the gravitational force and the value of this force will be dependant on the boundary conditions of each test.

In the inclined circular tube problem the flow is defined parallel to the tube centreline and is caused by the component of gravity in the direction of the tube centre line, the flow velocity in this conditions is a function of the distances from the centre to the boundary walls, in Figure 6.9 is shown the components of the gravitational force acting on the fluid flow inside a tube inclined with an angle θ.

Using a no-slip boundary condition where $\nu_x = 0$ at $D = r_t$ the Poiseuille parabolic flow equation [97] can be defined as(Equation 6.3):

$$\nu_x(D) = \frac{\Delta P_x + \rho g_x}{4\mu} \left(r_t^2 - D^2\right) \tag{6.3}$$

Where $\nu_x(D)$ is the flow velocity on the x direction as a function of the distances to the centreline of the tube, ΔP_x is the pressure differential on the x direction, g_x is the gravitational force component on x, D is the distances to the centreline and r_t is the tube radius.

If ΔP_x and the tube radius r_t are keep as constants then the only variable defining the flow velocity for a specific fluid is g_x. The value of the gravitational component on x depends on the angle θ of the inclined circular tube.

The solid tubes models that will serve as boundaries conditions on the SPH numerical solution are build from a set of images. Each image contains a circle with a radius of 10 pixels (the final area depends on the area per pixel defined during the problem setup) and the centre of the circles depend on the centre line angle (θ) with respect to the horizontal. The images are transformed into a 3D object using an implementation of the Marching cubes method.

The SPH implementation has its own Marching cubes system and accepts 2D image datasets as input. The final setup consists of three-dimensional tubes with different angle (θ) values, Figure 6.10.

Using a voxel size of 0.1 μm the final three-dimensional tubes have a radius of 1 μm and a vertical length of 20 μm

The initial state of the SPH particles for the different tests is show in Table 6.8.

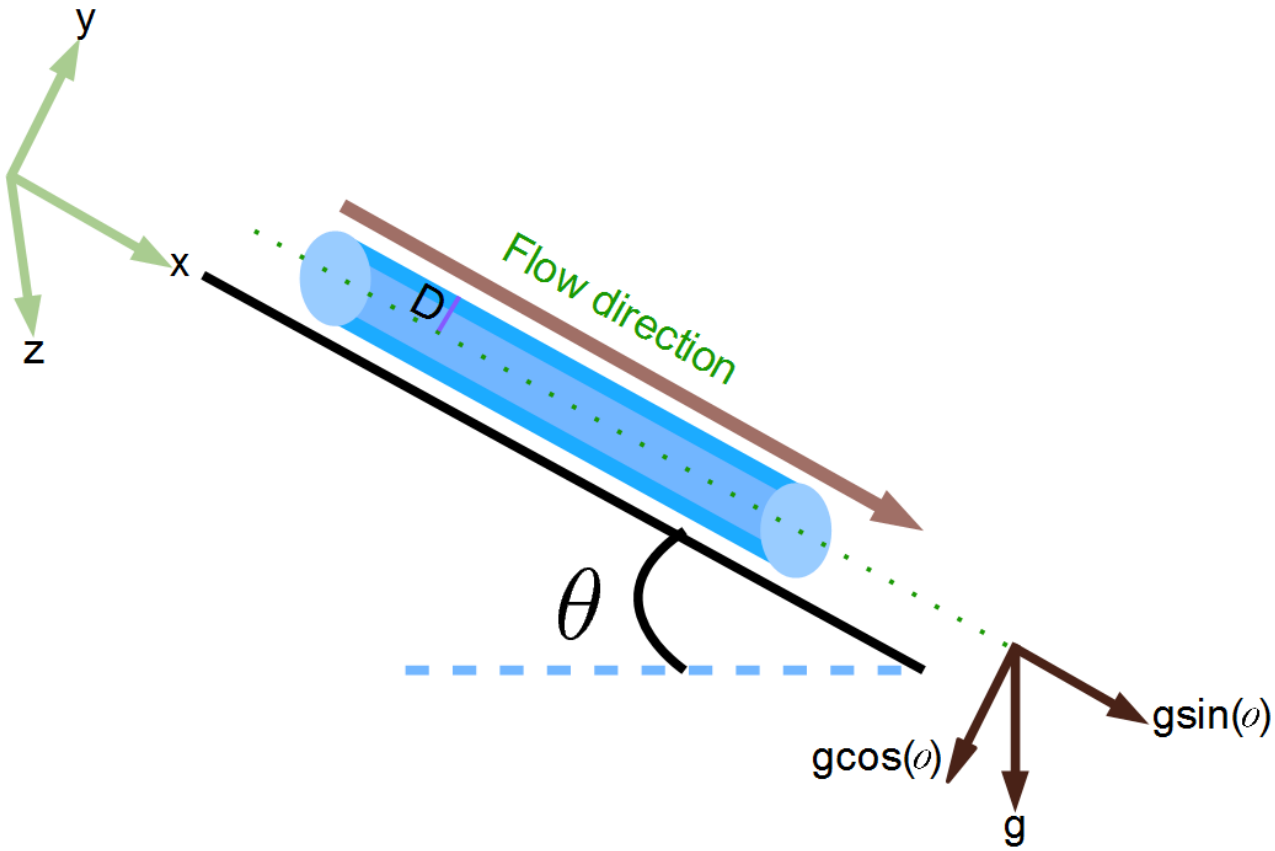


Figure 6.9: **Inclined cylindrical tube test conditions** - To defined the fluid flow equation for the inclined circular tube, cylindrical polar coordinates are introduce with the x axis along the centreline of the tube.

Test setup		
Property	Value	Units
Total particle memory elements size	150000 elements	dimensionless
Tree diameter	0.002	m
Differential time step dt	0.0001	s
k	0.95	dimensionless
Ω	0.55	dimensionless
Collision scale factor	1.0	dimensionless
e	0.5	dimensionless
fluid particle resolution	$[1.74 \times 10^{-4}, 1.74 \times 10^{-4}, 1.74 \times 10^{-4}]$	m
Solid particle resolution	$[8.7 \times 10^{-5}, 8.7 \times 10^{-5}, 8.7 \times 10^{-5}]$	m
Fluid type	Water	
Solid type	Virtual	

Table 6.8: Initial conditions for the inclined circular tube test. In the test the boundary conditions are defined by solid particles

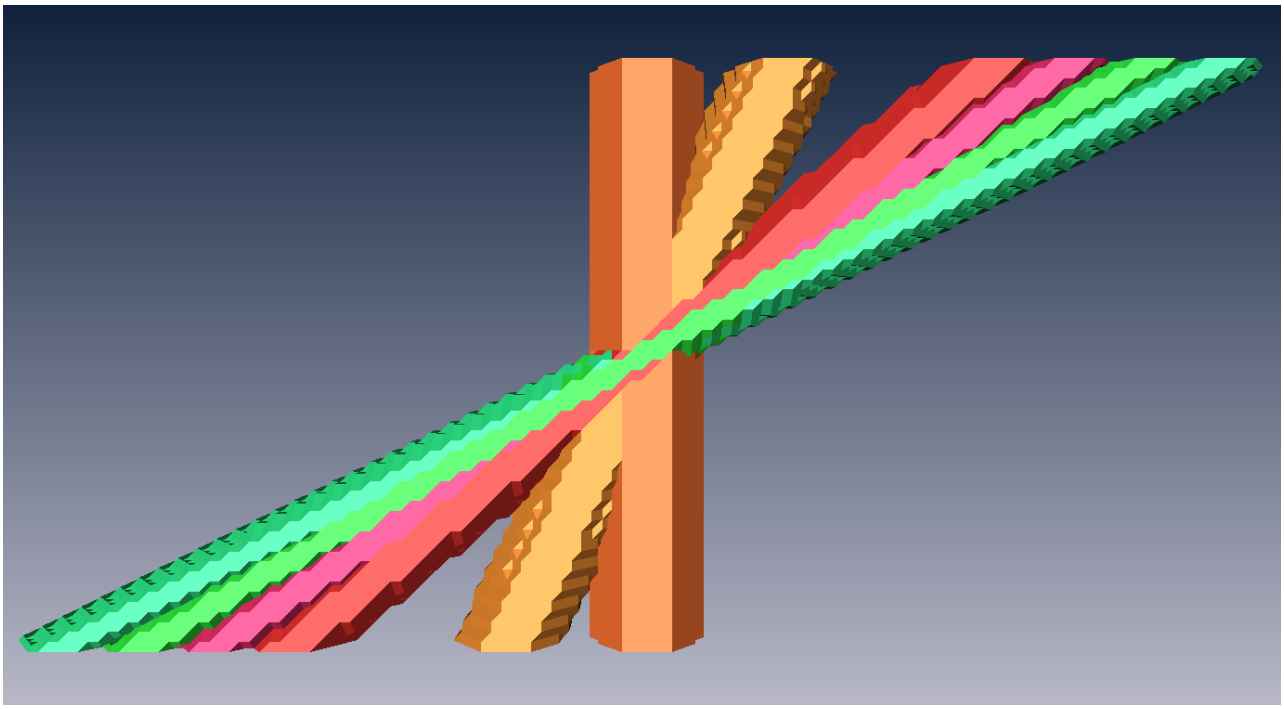


Figure 6.10: **Inclined circular tube test boundary objects** - Different test objects for the computational simulation. Each boundary particle system has a specific angle the values are shown on Table 6.9

Table 6.9 presents the resulting inflow, outflow, and permeability for each test. Since the boundary conditions are obtained from a mathematical model the porosity is set to 1.

Slope test results				
Test #	Slope °	Inflow $\times 10^{-12}[m^3/s]$	outflow $\times 10^{-12}[m^3/s]$	permeability $\times 10^{-8}[m^2]$
1	90	3.136	3.049	2.103
2	85	1.970	1.905	1.314
3	80	1.040	0.907	0.626
4	77.5	0.651	0.589	0.406
5	75	0.192	0.164	0.113
6	70	0.0148	0.0113	0.00779

Table 6.9: Outflow and permeability results for the inclined circular tube test. As the angle decrease the resulting outflow decreases.

In the Table 6.9 it can be seen how the outflow is affected by the boundary conditions and the angle of the different tube test. The inflow for each tests has different values since it is updated using the particles that exit the boundary volume, so after some iterations the inflow and the outflow approximated.

As the angle of the tube decreases the gravitational component on g_x decreases and the particles will have a lower external body forces acting over them, reducing the final acceleration of the particles.

Figure 6.11 and Figure 6.12 shows the outflow result for the different tests.

The figure shows that the only tube that maintain a more symmetric representation at the outlet is the one with an angle value of 90°, for this test tube the gravitational force only acts on the direction of the centreline of the tube and it can be seen that the flow velocity reduces as the distances from the

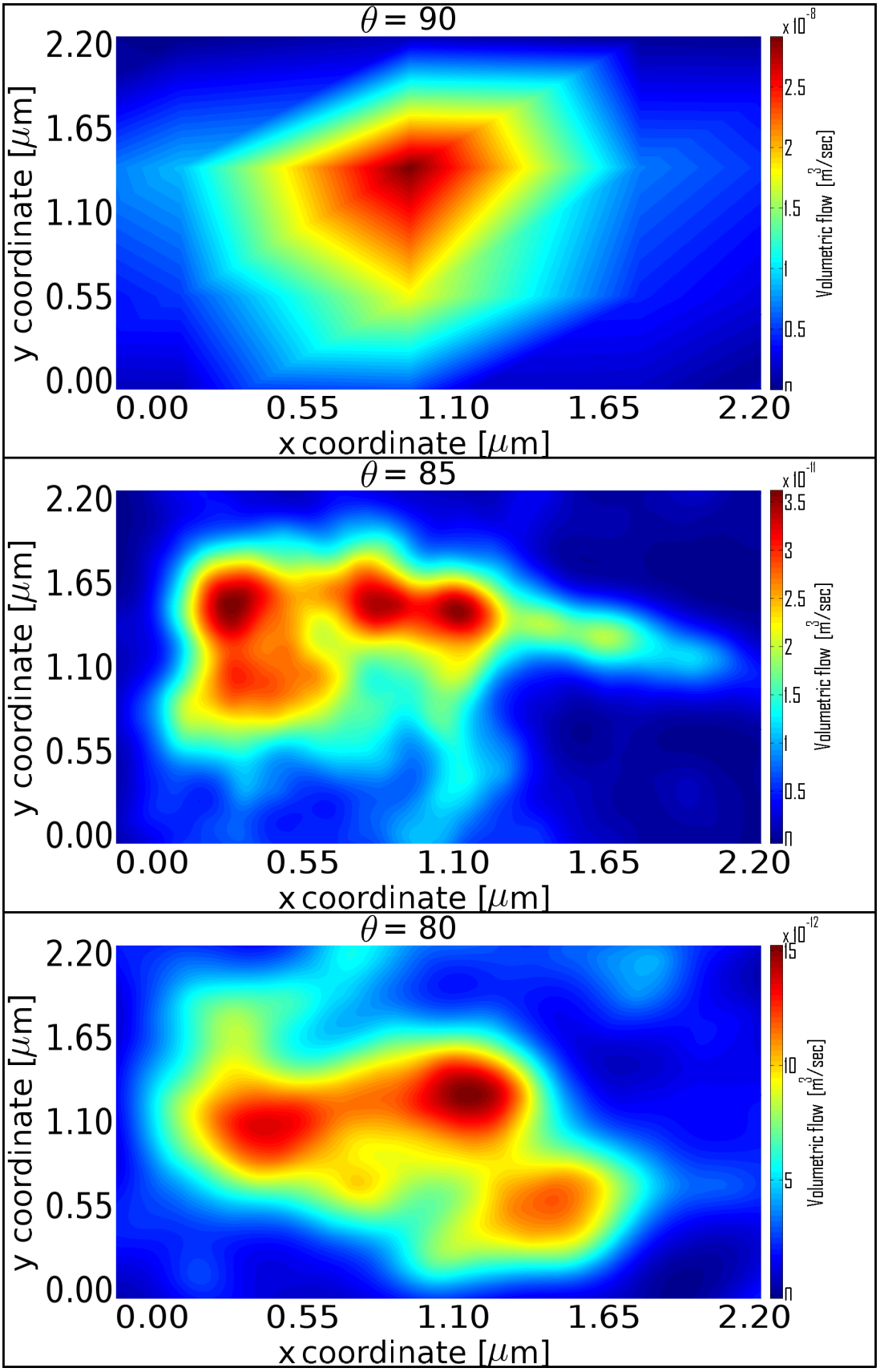


Figure 6.11: **Test outflow result for angles: 90, 85 and 80 degrees** - The change on the angle of the boundary conditions produced a change on the acceleration force and the resulting outflow.

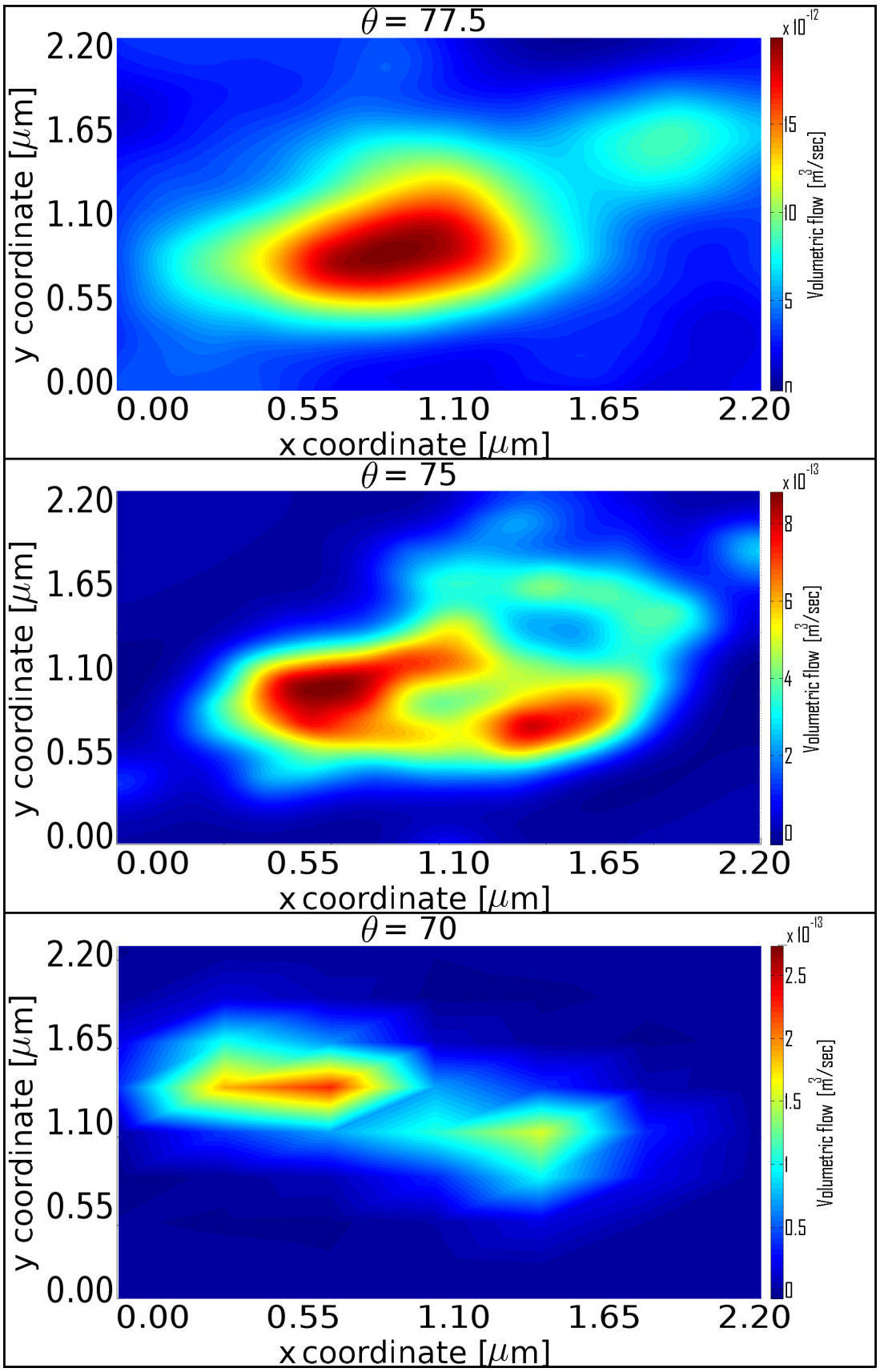


Figure 6.12: **Test outflow result for angles: 77.5, 75 and 70 degrees** - The change on the angle of the boundary conditions produced a change on the acceleration force and the resulting outflow.

centreline increases.

6.4.2 Tortuosity fluid test

The initial state of the particles for the tortuosity fluid test is shown in Table 6.10.

Test setup		
Property	Value	Units
Total particle memory elements size	150000 elements	dimensionless
Tree width	0.002	m
Tree height	0.002	m
Tree depth	0.002	m
Differential time step dt	0.0001	s
k	1.0	dimensionless
Ω	0.05	dimensionless
Collision scale factor	1.0	dimensionless
e	0.5	dimensionless
Solid cube size	$(x : 0.00050, y : 0.00050, z : 0.00050)$	m
Fluid cube size	$(x : 0.00015, y : 0.00015, z : 0.00015)$	m
Solid cube dimensions	$(x : 7, y : 7, z : 7)$	dimensionless
Fluid cube dimensions	from $(x : 3, y : 3, z : 3)$ to $(x : 30, y : 30, z : 30)$	dimensionless

Table 6.10: Initial conditions for the Tortuosity test. The conditions presented in the table are used as the initial value for all the tests.

Since tortuosity is going to be analysed, the geometric properties of the test boundary need to be considered. Table 6.11 presents the geometric properties of the test objects.

Geometric properties				
Test #	Equation	Skeleton length m	Effective Distance m	Tortuosity
1	Line	0.0050	0.0051	0.985
2	Sine	0.0071	0.0054	1.322
3	Parabola	0.0076	0.0052	1.460
4	Tractrix	0.0121	0.0052	2.309

Table 6.11: Geometric properties of the different test objects, the shape of the test objects determines how the boundary will affect the fluid flow inside it.

Figure 6.13 shows the different shapes with their respective fluid flow, the colour of the particles represents their velocity magnitude. The HSV colour scheme is used as the colormap for the visual output, where blue is set to the lowest velocity and red to the highest velocity.

Tortuosity changes the fluid flow at the outlet since it puts two new conditions on the flow. The first one is the total real distances the fluid needs to travel inside the shape form the inlet to the outlet, the skeleton length represents the central distances the fluid will travel.

The second condition is the collisions with the solid walls, as the tortuosity increases the fluid will change its course at more points of inflection, this will cause points of reduced acceleration that will reduce the final flow output. Table 6.12 presents the outflow results of the tests.

The outflow is shown in Figure 6.14

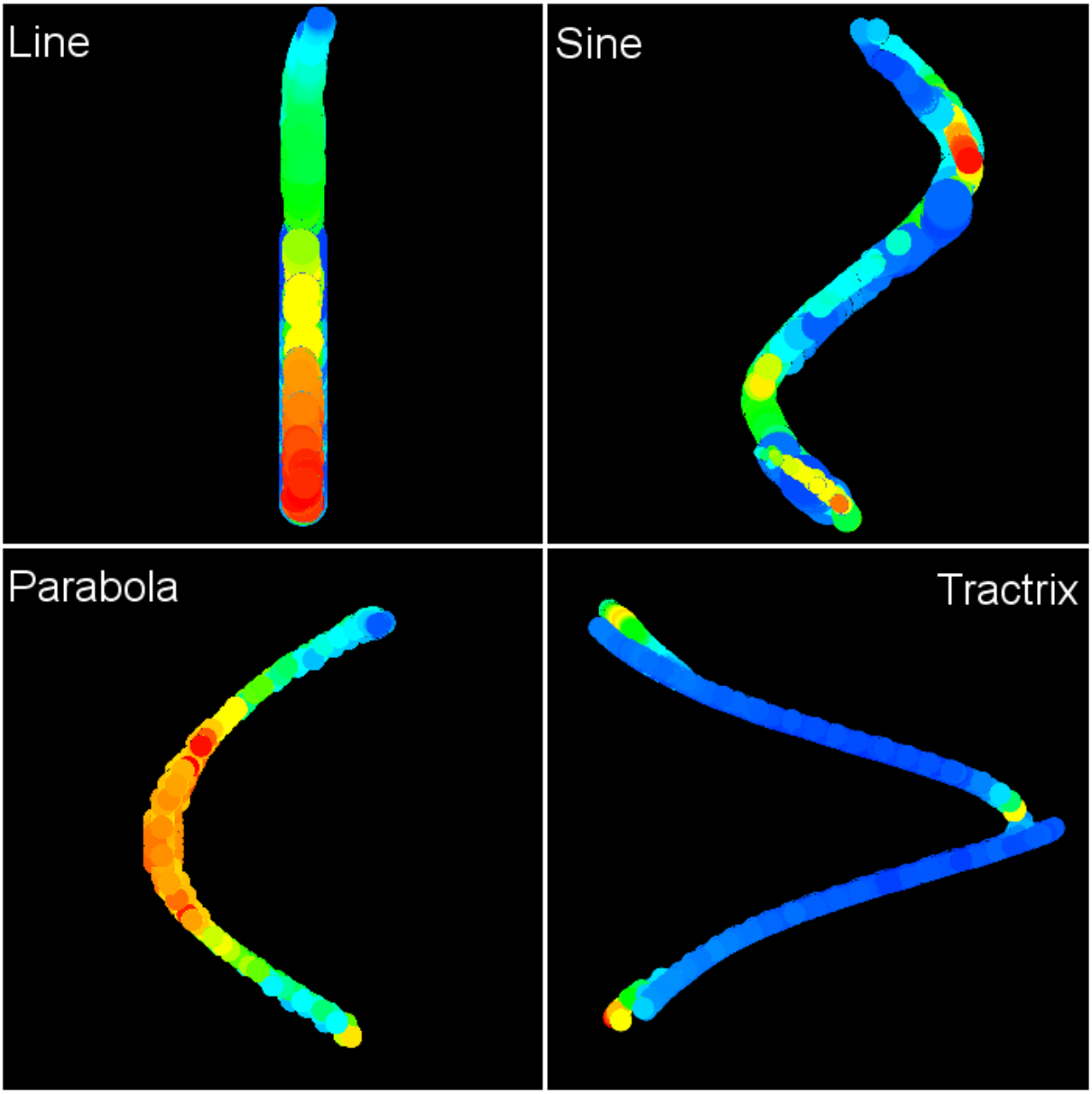


Figure 6.13: **Tortuosity fluid flow results** - Tortuosity fluid flow result on the test boundary. The different boundary conditions modified the behaviour of the fluid flow. In the images it can be seen how the shape of the fluid will adapt to the different types of inflections on a boundary.

Tortuosity test results				
Test #	Equation	Inflow $\times 10^{-12}[m^3/s]$	outflow $\times 10^{-12}[m^3/s]$	permeability $\times 10^{-6}[m^2]$
1	Line	1.406	1.049	2.264
2	Sine	0.591	0.502	0.522
2	Parabola	0.00329	0.00368	0.00571
3	Tractrix	0.00988	0.00988	0.00734

Table 6.12: Tortuosity tests outflow result for the different boundary conditions.

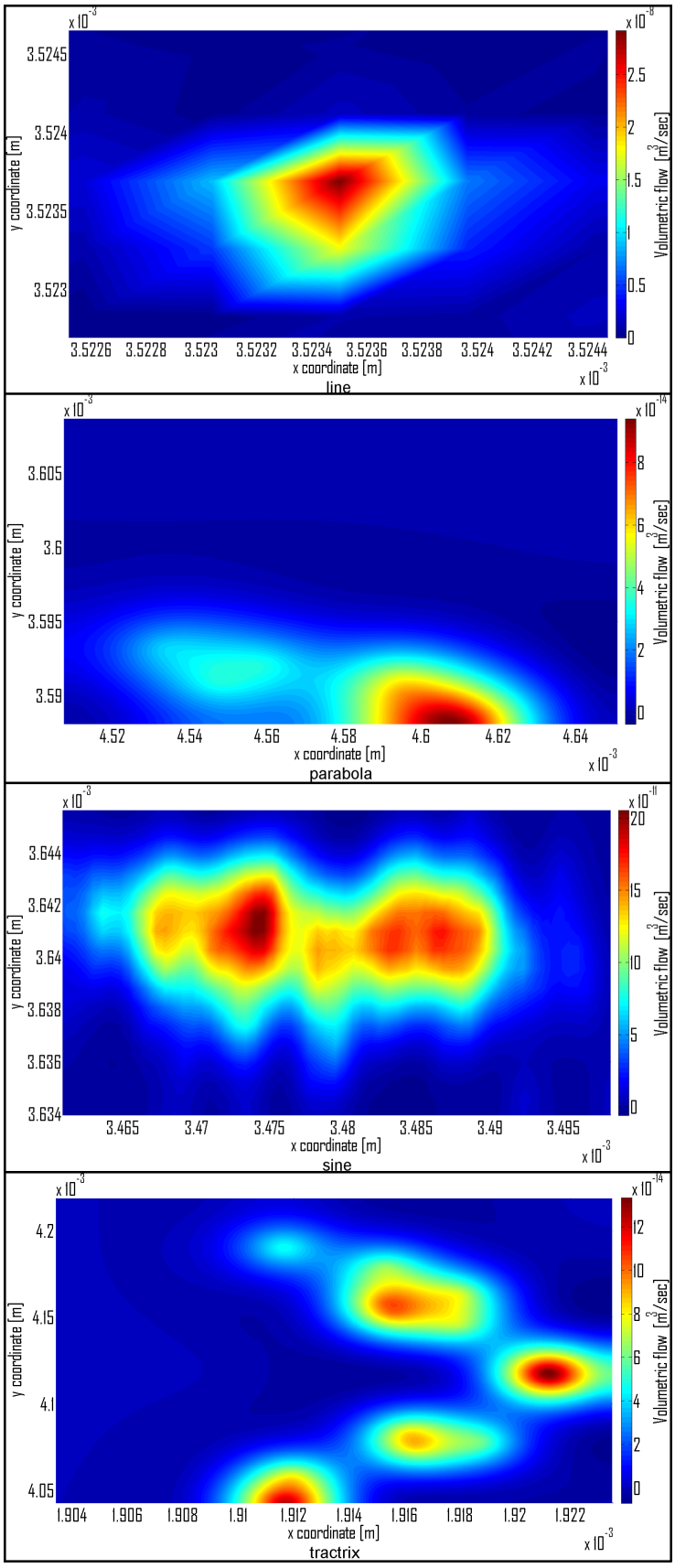


Figure 6.14: **Tortuosity test outflow result** - The figure shows the outflow behaviour on the different test objects at the outlet. The plots shows flow velocity per voxel, depending on the boundary shape the amount of the outflow will changed.

6.4.3 Particle volume size

The last point of consideration when using SPH to simulate fluids inside a boundary is the relation between the input data and the particle volume. This problem will be of special importance for solving flow of fluid on the porous mediums.

In the cases where the input data is obtained using a method that provides real volume values, for example CT-scan data, the volume per voxel is used with the density of the fluid to define the particle mass that will represent the voxel.

If the solution requires a higher resolution than the one assigned to the data then each voxel can be divided, the fraction of the voxel will provide the new volume to obtain the mass of the particle representing that fraction.

When this approach is used there are two limitations that need to be considered:

- Collision volume resolution: The voxels are cubic representations of volumes, when the voxels are pass to a particle representation the volume of influence is changed to a spherical representation. If the solid particles that form the boundary are obtained directly from the voxels the collision factor needs to be greater than 1 to prevent empty space between the boundary particles. Increasing the volume of interaction in this manner can cause two types of error:
  - 1: If the scale factor is to big some particles can start the simulation with interpenetration, if the amount of interpenetration between two or more particles is big enough the repulsive force resulting on the collision resolution will accelerate the particles to a point were the time step calculations will not be able to prevent the fluid particles to escape the boundary.

    The same problem happens in simulations where solid deformation is allowed, in this case the acceleration will cause the destruction of the solid objects.
  - 2: If the scale factor is too small then the boundary particles will occupied a volume smaller than the one needed to obtained the discrete continuum representation of the original solid boundary. In this case empty space will appear on areas were the original solid was present and some fluid particles will be able to flow through the boundary at this points.

  Figure 6.15 presents two different frames for two boundary test that present these problems.

- Internal object volume resolution: Another problem is when the objects particles are defined with too little or too much mass, this will mean that the continuum approach is not followed. The problem is due to the relation between the volume, density and mass.

  When a new object has been defined with the aid of a particle system the three variables that are commonly known are the density per particle (same density as the material been solved), the total problem volume and the number of particles that will be used to represent the volume (the number could be related for example to the number of pixels, voxels or triangle nodes) when all these variables are known then the mass per particle is easily obtained.

  The problem is presented when either the density per particle or the volume per particle is unknown in such cases the mass per particle is provided in the problem definition. The mass value must ensures that the sum of all the particles volume will approximated the total volume problem.

  If the defined mass per particle is too little then the continuum representation of the material will not be keep since the final volume per particle will not be big enough.

  On the contrary if the defined mass is to much the volume from two different particles will overlap and occupied the same space producing errors on collision detection and internal forces resolution.

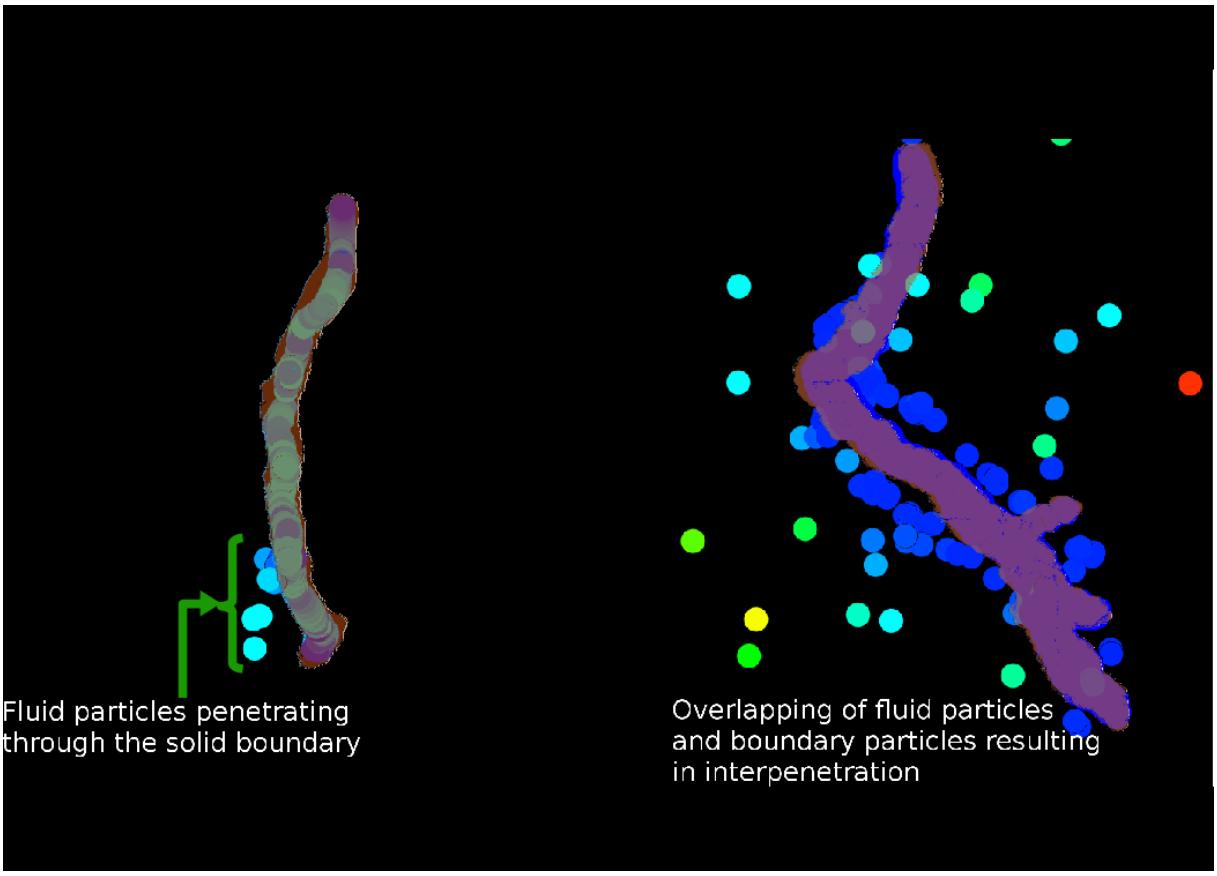


Figure 6.15: **Error on initialisation over boundary limits - Left:** Simulation where the particles forming the boundary walls do not have sufficient volume to occupy all the space originally defined as solid, the fluid will flow through this empty space. **Right:** Fluid particles accelerating outside the boundary volume due to the initialisation of solid-fluid particles with high interpenetration. The interpenetration will cause a collision responses with and acceleration higher than the differential resolution of the numerical solution

- Object particle resolution: This problem is presented when the number of particles obtained from the database that defined the objects (set of pixels or voxels) is lower than the particle resolution required by the parameters of the numerical approximation in order to obtained a correct solution of the problem.

  One common computational error when solving fluid-solid or solid-solid interaction problems is presented when one of the objects, the solid or the fluid, have a lower number of particles per volume, the differences in particle resolution can produced errors on the particles interaction (e.g. collision resolution).

One solution for the differences on the objects particle resolution is the use of a meshing algorithm to build a triangle mesh of the solids boundary surfaces or a tetrahedral mesh for the solids volume. the vertex from the meshes will be used as the initial position of the particles.

One application of this approach is for example when the fluid particles numerical solution requires a better quality solid boundary than the one originally defined.

To increase the boundary quality resolution a mesh can be defined for the solid boundary and after that a mesh subdivision algorithm can be applied to increase the number of vertex forming the boundary.

In this thesis the loop mesh subdivision method presented in [98] is used. Figure 6.16 presents the same mesh with different resolutions.

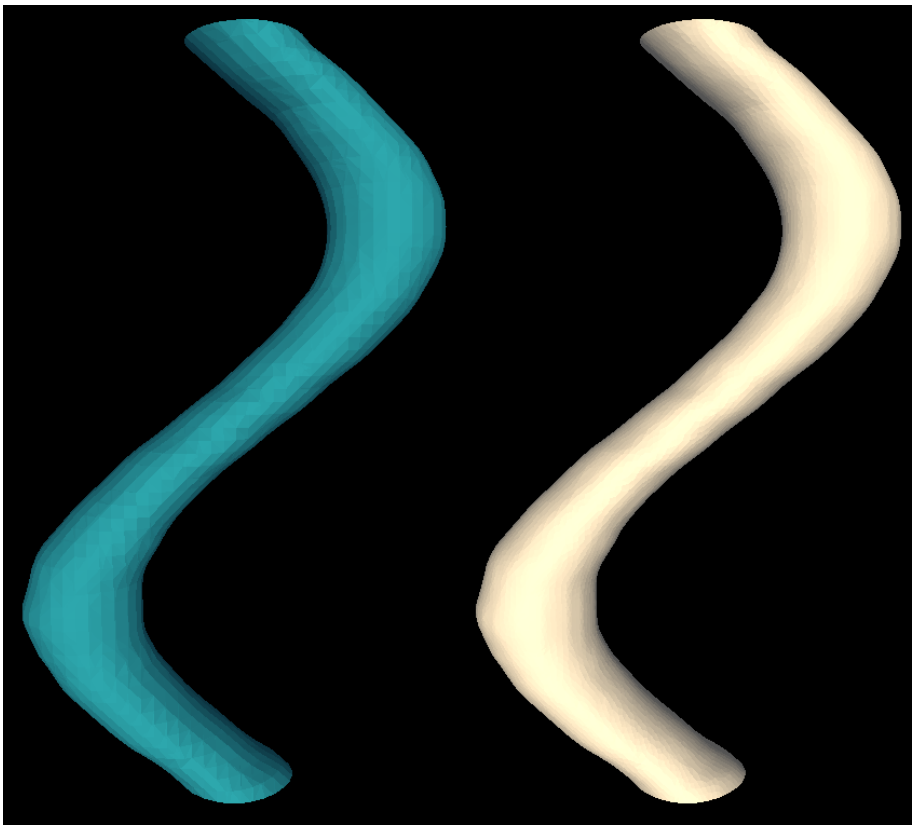


Figure 6.16: **Mesh boundary resolution - Left:** Original mesh of the sine function with 3505 vertex. **Right:** Mesh resulting from the loop re-meshing algorithm with 11877 vertex, this will provided with a higher quality solid boundary to be used on the numerical solution of the fluid flow.

One of the improvements obtained by using a more define mesh is in the collision detection system, when the resolution is increased the collision components are more precise since the particles will approximate a smoothed surface with less discontinuities. Figure 6.17 shows the same solid object with different particle resolution.

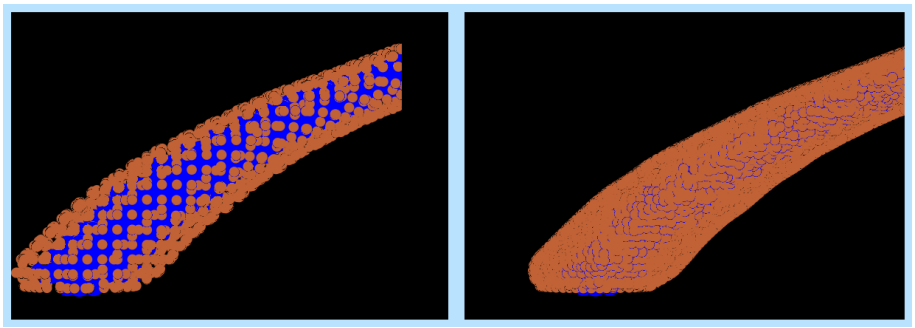


Figure 6.17: **Particle boundary resolution - Left:** Original particle system, the OpenGL visualisation is set-up to show the particle components, the collision scale factor and the h value of the particles will define the numerical approximation to the material continuum. **Right:** Same boundary model with an increased number of particles, as the resolution increases the numerical approximation to the original boundary is more accurate.

Finally for small size problems (for example on the areas with pore structures in the ranged of the micrometers) empty points with small volume surrounded by solid material need to be discriminated . If the particle resolution is too low then the interaction volume will occupy the empty space and the flow will be interrupted.

6.4.4 Solid deformation

Using the SPH equations described in Chapter 4 the solid can be considered to be a deformable object inside the simulation, the implementation of the SPH solution has some differences from the fluid SPH:

- Solid SPH particles will not need to update their neighbour unless the solid composition is modified (break apart).
- Collision responses are only solved in the boundary particles of the objects.
- Internal forces have different definitions depending on the solid type.

The correct solution of the solid deformation using the proper SPH implementation depends on:

- Time step relation with the force magnitude and the velocity of particles. The time step should have sufficient resolution for the solid particles to be able to interact with their neighbourhood through the internal force and the particles from other objects when colliding.
- k, k_c, and Ts, scale factors that provide the neighbourhood and collisions list to be solved by the system.
- Solid type definition and solid material constants.

The first two points are similar to those used for the simulation of fluids and require the same consideration. For the last point, the material conditions (depending on the constitutive equation) are obtained (in most cases) from the literature. When studying complex materials a stereoscopic system to measure material deformation is used.

In order to test the system the first step is to study the solid deformation modelling in the SPH implementations.

Solid kinematics

Solid deformation due to external forces depends on the numerical time step (Δt), number of neighbours, and the solid constitutive equation. On this test a single object will be subject to a load produced by the gravitational forces.

In the first test the solid properties are kept constant (Table 6.13)

Then Δt is varied from 500 μs to 62.5 μs in order to obtain different particle behaviours. The value 500 μs is chosen because for values over 800 μs the simulation of the viscoelastic models fails and the solid is destroyed.

The position of the top layer particles of the test object is used to obtain the final distribution of particles on the z depth plane. The particles start with 0 in acceleration and a uniform distribution on the plane (all particles on the plane have the same z value), after the gravity acts over the body the position of the particles will changed. Table 6.14 shows the results for different Δt values and different solid constitutive equations.

Solid Test setup	
Property	Value
Total particle memory elements size	150000 elements
Tree width	0.002 m
Tree height	0.002 m
Tree depth	0.002 m
k	1.4
Ω	1.0
Collision scale factor	1.0
e	0.65
Solid cube size	$(x : 0.001, y : 0.001, z : 0.001)m$
Solid cube particles dimensions	$(x : 5, y : 5, z : 5)$
Volumetric force acceleration	9.8 $\frac{m^2}{s}$

Table 6.13: Initial conditions for the change in time step scale for the solid configuration.

Single object deformation results				
Test #	dt μs	Solid type	Particle z-position mean mm	Particle z-poisition std $\times 10^{-6}$ m
1	500	Elastic	3.3	6.23
1	500	Linear elastic	3.4	1.17
1	500	Viscoelastic	3.4	3.36
2	250	Elastic	3.3	7.74
2	250	Linear elastic	3.4	0.85
2	250	Viscoelastic	3.4	3.37
3	125	Elastic	3.3	8.34
3	125	Linear elastic	3.4	0.36
3	125	Viscoelastic	3.4	3.37
4	62.5	Elastic	3.3	8.61
4	62.5	Linear elastic	3.4	0.56
4	62.5	Viscoelastic	3.4	3.37

Table 6.14: Single object deformation test under different time steps and constitutive model equations. The object is only under the action of gravitational forces and the interest of the test is to measure particle discrepancies over a plane

Since the volumetric force (gravitational acceleration) is kept constant and with a value of $9.8[m^2 \cdot s^{-1}]$, the differences between the test are minimal, but using the std value it can be seen that different materials will have a different final particle distribution on the top layer.

Other important observation is that for the same material the changes on the value of the std when the numerical resolution on time is reduce is minimum and that all the time steps values will produced a similar numeric result. This is due to the fact that the acceleration over the particles can be solved with all the values of Δt presented.

If the resolution on time is reduce even more the solid objects will explode and the numerical resolution will fail.

The most important differences between the elastic and viscolestic materials is that in the viscoelastic model a damping value that will produce energy dissipation is introduced. Figure 6.18 shows a visual result of the simulation with the three methods.

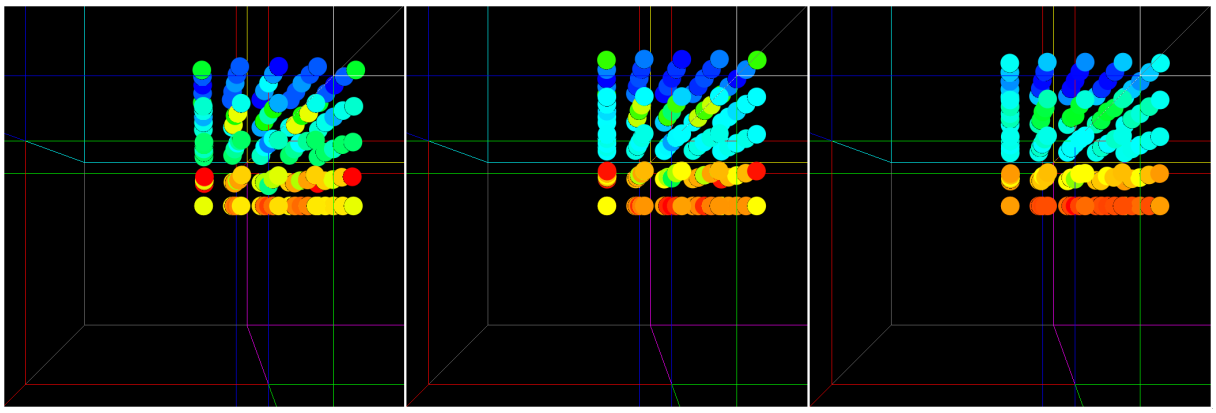


Figure 6.18: **Single solid deformation** - Single solid object under the action of gravity. **Left:** Elastic behaviour of the solid, the particles show a greater amount of strain overall (HSV colour scheme represent strain value) than the linear elastic and viscoelastic modelling. **Centre** The strain amount is reduced but the top corners of the objects have still a high amount of strain and the deformation will continue **Right:** The viscoelastic components of damping have stopped the deformation, the strain is still in effect since the volumetric forces are acting on the body.

Solid collision responses

The next test is the collision system and the differences in results when using different materials and collision k_c factor value.

The test uses the same conditions for all the types of solid materials the collision responses is modelled without the anti-penetration module. Without this module the collision between particles is solved until the magnitude of the velocity between the colliding particles is less than a limit (in theory the contact velocity will be 0).

When the velocity of the particles during collision is less than the collision limit and without the anti penetration module particle interpenetration will occur. After the particles penetrate each other the amount of energy in the interaction will increase rapidly resulting in the destruction of the models. Figure 6.19 showns the results of the different runs.

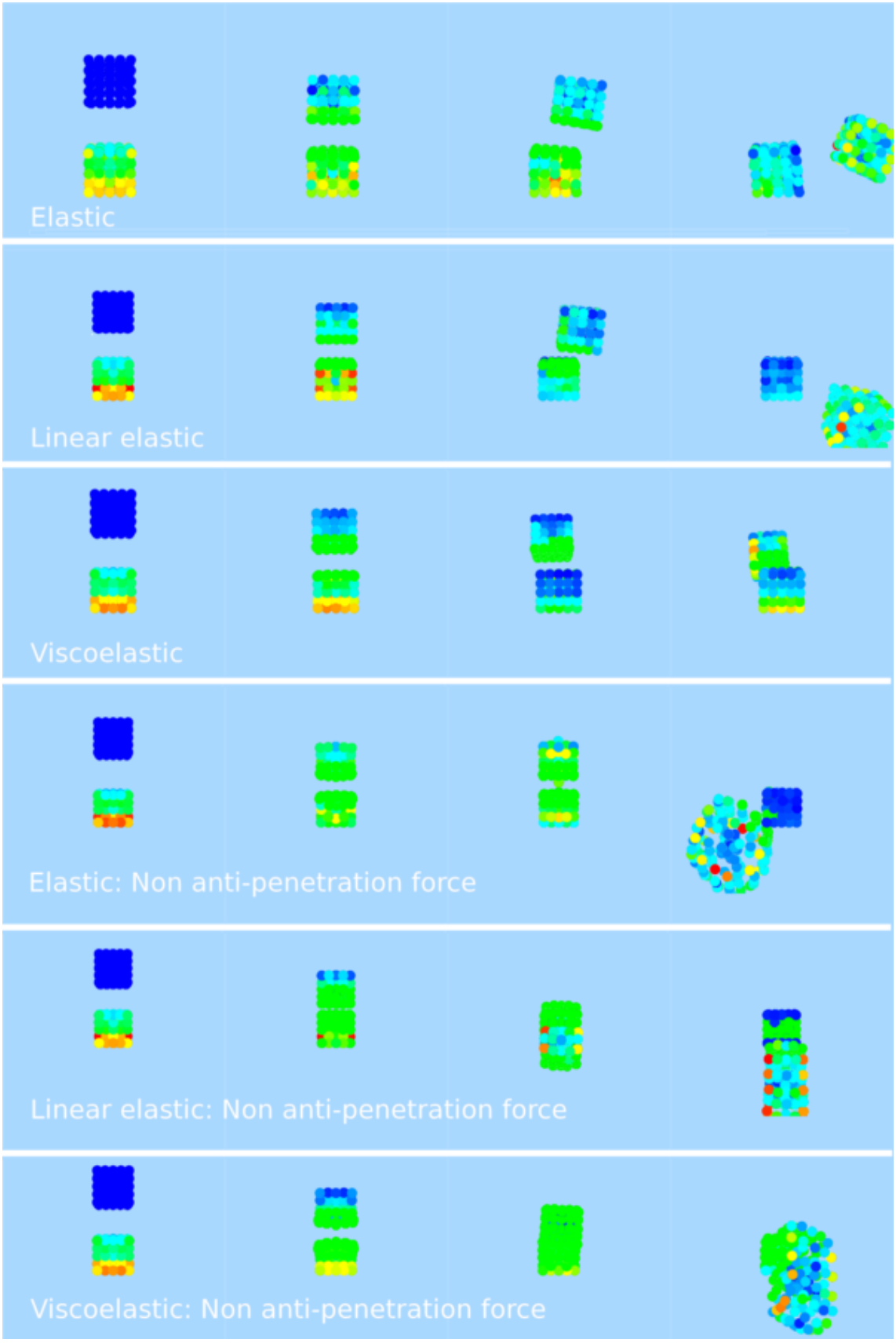


Figure 6.19: **Solid collision** - Collision results between solid objects with different properties, the use of different constitutive equations modified the way the objects will react, the collision is defined for all cases. When the anti-penetration module is not used the penetration error can be seen and the problem of two particles in the same position at the same time causes an error in the physical simulation.

6.5 Solid deformation measurement using stereo-vision

The Elastic surface deformation of a real cubic object is described using depth information obtained at video frame-rate with camera calibration and a stereo matching technique, the deformation of tracking points on the object surface will be used to describe the process of deformation undergo by the cubic object under stress, the points will be the input data to the computational model.

The images pairs are obtained using two synchronised USB cameras, model uEye UI-1220SE-C-HQ (752 X 480 pixels), running at 57 fps. Using this system it is possible to track surface deformation at a rate of 0.0175 s.

The frame rate determines the amount of external forces that can be input as a deformation over the object. If the forces are too big the differences in the pixel position between two frames will have a resolution in the scale of cm and the small deformation theory analysis will not describe the study.

The setup of the experiment is shown in Figure 6.20, the camera baseline is 4 cm and the distance to the object surface is 18 cm, the object of study is a black rubber and foam mix box with a side size of 6 cm.

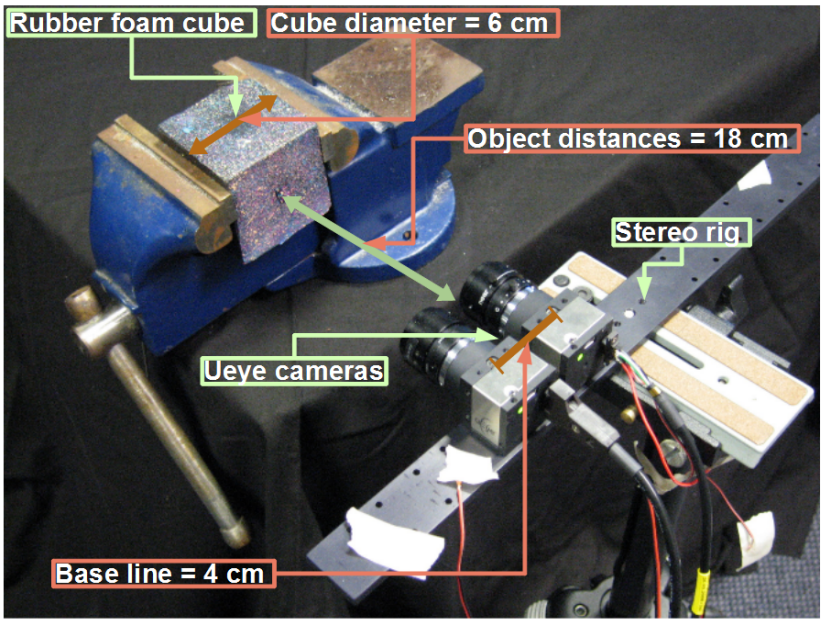


Figure 6.20: **Ruber-foam test setup** - Stereo system setup to study the deformation of the rubber-foam box under stress. The system is set up with a small baseline to be able to focus on small changes on the box surface.

The frame images are corrected to exclude lens distortion [99, 100] and get rectified stereo pairs. The depth information is computed with a simplified version of the belief propagation method. The stereo-vision system equipment and methodology used to solve this problem is described in [101]. The results of the stereo reconstruction are shown in Figure 6.21, the surface of the cube is deformed as the external forces act over the object.

The stereo pairs shown in Figure 6.21 used the grey-map scale values to relate a pixel intensity with a depth value. Before the analysis of the data is performed the histogram of all the frames is normalized

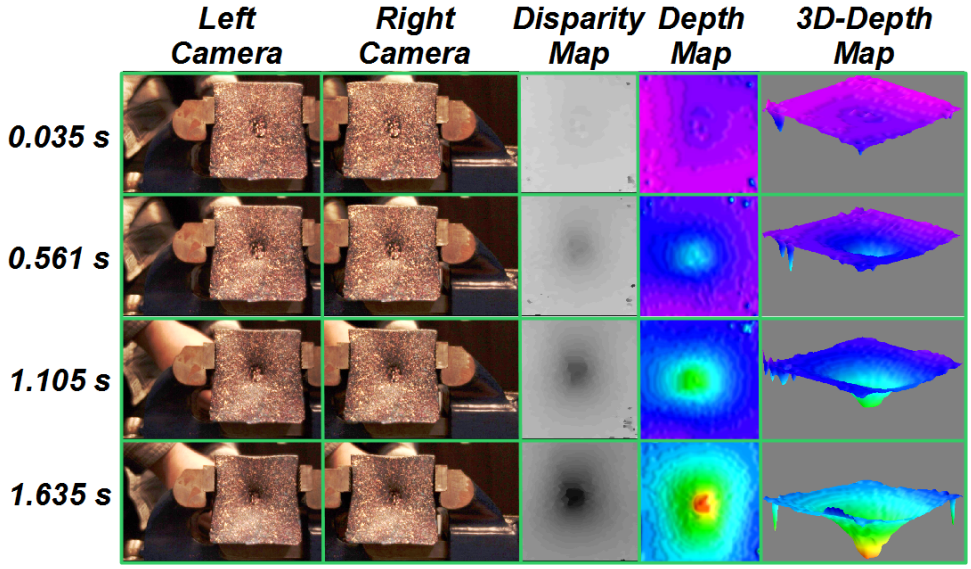


Figure 6.21: **Ruber-foam deformation results** - Stereo measurement results for the rubber-foam deformation test. As the forces act over the cube the depth map information provides a pixel by pixel measurement of the deformation. The figure shows the cube deformation from top (underformed) to bottom (maximum deformation)

and the average histogram of the database is used to obtain the grey-value per pixel for the whole stack of frames, Figure 6.22 shows the resulting frames with the centre profile on the x and y axes.

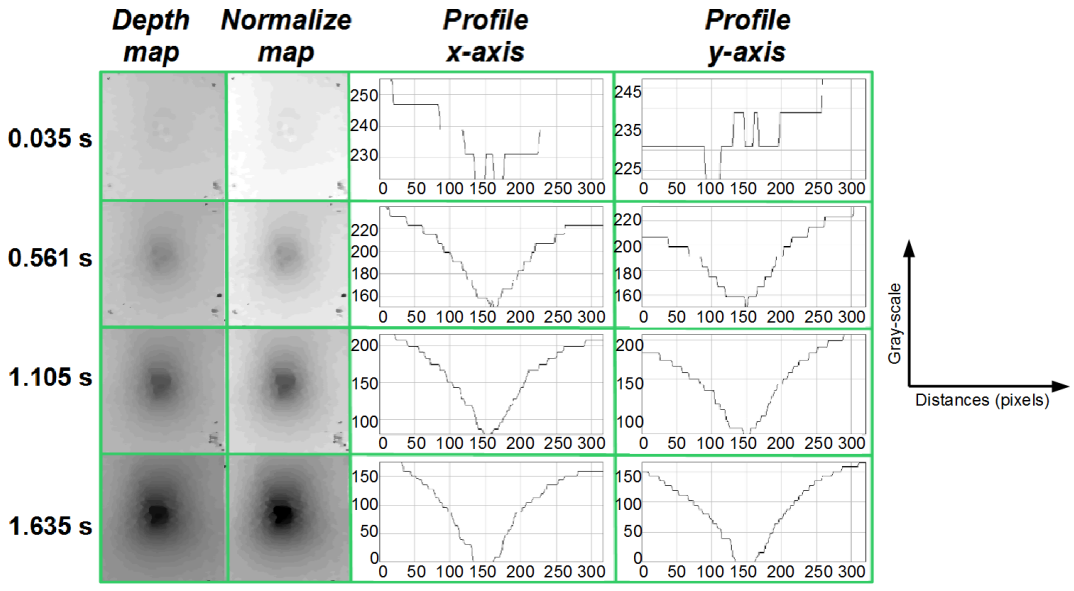


Figure 6.22: **Stereo depth maps profiles** - The figure shows the x and y centre profile on the normalised histogram of the frames obtained by the stereo-setup. The first map shows an undeformed cube, the profile is used to eliminated any bias introduced by the stereo setup since the result is not a flat surface.

In the paper [102] the strain of the object is obtained between two frames, the initial undeformed state and the final deformation. To obtain the value of the deformation a triangle mesh is built and the distortion of the triangles is measured as the strain value of the deformation. Using the stereo-video approach,

instead of measuring two stages of triangle deformation, the vector displacement between different time steps can be used to obtain how a pixel position changes with respect to their neighbourhoods. Figure 6.23 shows how the vector relation between pixels is used to obtain the deformation relation between two frames.

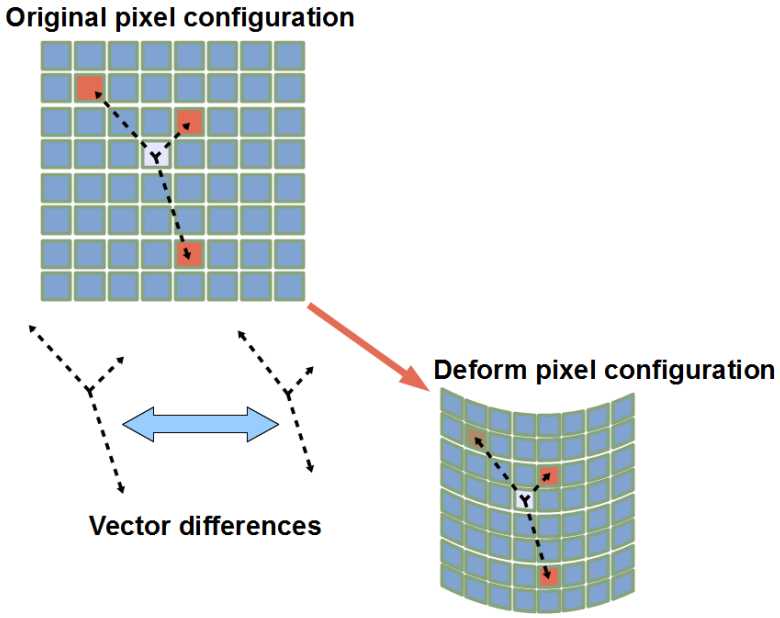


Figure 6.23: **Stereo pixel deformation** - The changes in the distances relation between pixels is used as the measurement of the pixel by pixel deformation. The forces used are selected so the change between frames is small, the limitation on the forces are the material properties and the framerate of the cameras.

During the experiment some conditions are maintained in order to reduce the number of variables in the problem:

- 1: The forces are applied at the centre of the object over a small area, for the rubber-foam test the area the forces were applied to was 0.5 cm (8.3% of the surface area)
- 2: The box boundary walls are fixed so the centre of the box changes the least possible amount during the test.
- 3: The forces are applied perpendicular to the surface xy - plane the box.
- 4: Force acceleration is maintained, so a changed pixel is kept inside the image resolution of 752 by 480 pixels. The box occupied an area of 317 by 323 pixels with a side size of 6 cm. The resolution is 0.0189 cm per pixel in the x direction and 0.0186 cm per pixel in the y direction. The minimum resolution is selected and the maximum change of a pixel between frames should be less than or equal to 0.0189 cm with a dt of 0.0175 s (obtained from the frame rate of the cameras) so the acceleration should be less than or equal to 61.5 cm/s^2, if the acceleration is maintain equal or lower to this value the pixel changed will approximated a linear behaviour.

Point 4 provides the theoretical limit to the movement, if this condition is maintained during the experiment then a pixel position between two frames can only be the same as or one of the pixels forming the eight-connected vicinity, this will help the optical flow measurements computational time. The small deformation theory analysis approach is followed.

After the experiment is finalised a series of space positions per pixel object are obtained. The pixel objects are defined with respect to their original position at time step 0 and how they changed during the experiment is tracked. Figure 6.24 shows a vector plot displaying some of the pixels as they change over time.

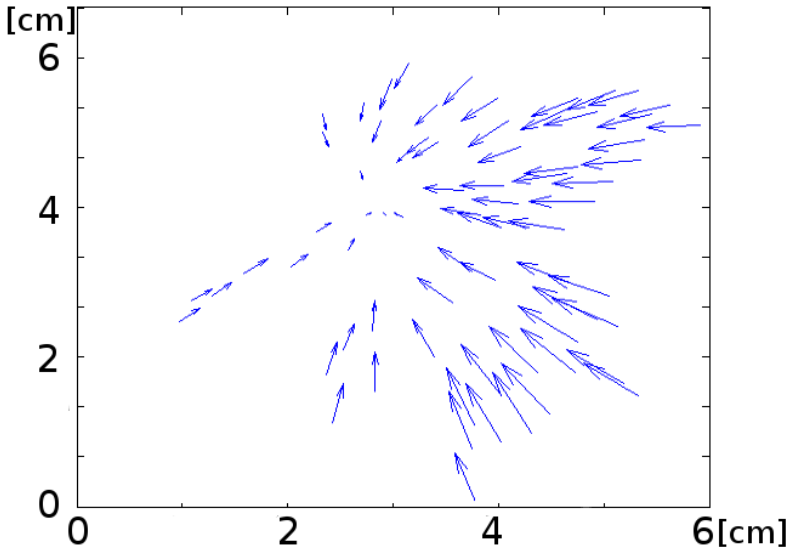


Figure 6.24: **Stereo pixel displacement field** - From the stereo data a discrete displacement field can be defined using the changes in position of the pixels.

The pixel positions and the time step value are used in the strain and stress module of the SPH simulation. A solid particle box is defined, the top pixels are related to a pixel in the stereo images, so the resolution of the SPH problem is equal to the pixel resolution of the experiment. The object is considered to be viscoleastic since the foam inside the object will prevent perfect elastic behaviour.

Using the stereo-vision video database our known variables are:

- Force acceleration and area of effect of the forces, the mass of the area is obtained from the SPH discretisation.
- Pixel (particle) position at the different time steps.
- Time differential dt between steps.

The unknown variables that can be obtained by direct computation using the SPH solution are:

- Displacement vector per particle $\mu\ m$.
- Strain per time step ϵ (dimensionless).

Finally the unknown variables that need a new module are:

- Stress $\sigma\ Pa$.
- Elastic coefficient $E\ Pa \cdot s$.
- Viscosity coefficient $\eta\ Pa \cdot s$.

Algorithm 6.6: Internal force calculation in the SPH implementation for solid objects.

```
//the stre[i] matrix (stress) for viscoelastic solid is defined as:
stre[i] = E*ep + n*epdt;
//Where ep is the strain and epdt the strain rate
void pair_force(pointd* dvxdt, matrixd *stre, int i, int j){
  fe.x = (stre[i].xx/rhoi^2 + stre[j].xx/rhoj^2 ) * node_pairN->particle_dwdxij.x
      +(stre[i].xy/rhoi^2 + stre[j].xy/rhoj^2 ) * node_pairN->particle_dwdxij.y
      +(stre[i].xz/rhoi^2 + stre[j].xz/rhoj^2 ) * node_pairN->particle_dwdxij.z;
  fe.y = (stre[i].yx/rhoi^2 + stre[j].yx/rhoj^2 ) * node_pairN->particle_dwdxij.x
      +(stre[i].yy/rhoi^2 + stre[j].yy/rhoj^2 ) * node_pairN->particle_dwdxij.y
      +(stre[i].yz/rhoi^2 + stre[j].yz/rhoj^2 ) * node_pairN->particle_dwdxij.z;
  fe.z = (stre[i].zx/rhoi^2 + stre[j].zx/rhoj^2 ) * node_pairN->particle_dwdxij.x
      +(stre[i].zy/rhoi^2 + stre[j].zy/rhoj^2 ) * node_pairN->particle_dwdxij.y
      +(stre[i].zz/rhoi^2 + stre[j].zz/rhoj^2 ) * node_pairN->particle_dwdxij.z;

  dvxdt[i].x = dvxdt[i].x + (fe.x*particle_mass[j]);
  dvxdt[j].x = dvxdt[j].x - (fe.x*particle_mass[i]);
  dvxdt[i].y = dvxdt[i].y + (fe.y*particle_mass[j]);
  dvxdt[j].y = dvxdt[j].y - (fe.y*particle_mass[i]);
  dvxdt[i].z = dvxdt[i].z + (fe.z*particle_mass[j]);
  dvxdt[j].z = dvxdt[j].z - (fe.z*particle_mass[i]);
}
force_call(){
  //for all the pairs i and j on the system call:
  pair_force(dvxdt, stre,  i, j)
}
```

The Stress is related to the particle displacement. The acceleration of the particles depends on the internal and the external forces. if the external acceleration is known then the rest of the acceleration is only due to internal forces, the internal forces are obtained as a summation of the interaction of all the particles in the neighbourhood list (e.g. see Algorithm 6.6).

Where $particle\_mass$, $node\_pairN.particle\_dwdxij$, and rho are known variables in the SPH numerical solution that depend on the original shape of the object been model. $dvxdt$ is the known internal acceleration on the time step for each particle.

The unknown variable is the stress matrix $stre$, using the strain relation for the stress it can be defined as: $stre[i] = E * ep + n * epdt$, with this definition the two unknown variables are the elastic coefficient E and the viscous coefficient η, so we have a three equation system that can be solved and the variables E and η defined.

Since stereo vision is only an approximation of the problem, the final E and η are obtained after averaging the values obtained for the particles on the surface and over the different time steps, the final values are:

$$E = 5.00 \times 10^3 \text{ with an std of } 1.31 \times 10^1$$
$$n = 1.09 \text{ with an std of } 1.48 \times 10^{-1}$$

For other constitutive equations with more material variables more pairs could be used to obtain their definitions.

After the material variables are obtained, we have all the variables needed for a complete simulation, so a new simulation with the same conditions is performed. Figure 6.25 shows the result vector field of

the simulation of a box model using the found material variables E and η and with diameter equal to 10 cm .

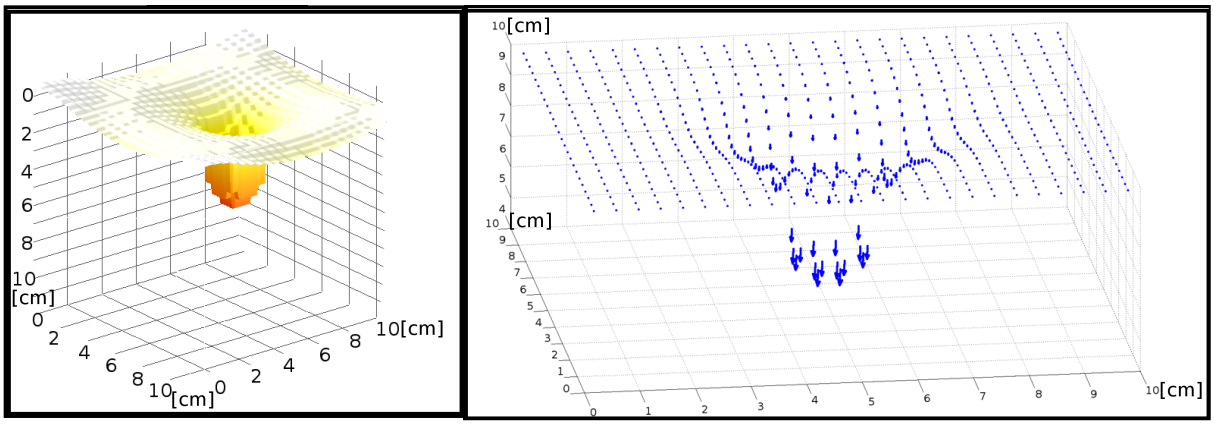


Figure 6.25: **SPH simulation of a Ruber-foam material** - After obtaining the model material values, the SPH implementation is used to simulate an object undergoing a deformation similar to the one in the experiment.

6.6 Smoothing Particle Hydrodynamics implemented on CUDA results

The test and CUDA development were implemented in the Intelligent Vision Systems New Zealand laboratory (www.ivs.auckland.ac.nz). I was in charge of the development of the parallel implementation and the project was completed with the coding assistances of the 780 project student Sina Masoud-Ansari and the PhD candidates Edwin Chan and Minh Nguyen. In this section of the Thesis the results of the implementation and a discussion of the improvements to the computational solution are presented.

The performance of the parallel solution of the SPH implementation depends on several factors:

- Host - Device memory transfer: Communication between the host(CPU) and the device(GPU) requires a memory transfer process. The relation between the number of memory transfers and their size will determine the total cost of the communication

- Serial component of the kernel: The parallel approach to the SPH implementation uses a CUDA thread inside a kernel to solve the numerical approximation of the behaviour of one particle. Inside the thread all the necessary operations to obtain the numerical result of the particle are perform in serial.

  The serial operations performed for each particle should be as similar as possible, with the lest numbers of if statements, so the CUDA performance is maintained. Reducing branch paths inside a kernel will increase the performance.

- Coalesced memory access: The use of coalesced memory increased the CUDA performance since the access to the data is more thread efficient. SPH used different types of memory arrays to represent the computational variables, these arrays do not follow the data rules for coalesced memory access, a padding approach is used to solve this problem.

- Data precision: The CUDA architecture is still under development and double precision capabilities are limited. The computational solution for doubles cost twice the clock cycles of the single precision solution.

In this section the results of the CUDA test will be presented.

6.6.1 CUDA memory management

The biggest limitation on the CUDA implementation is the available memory, The test were made on a GPU NVIDIA Geforce GTX 580 with a memory size of 1536 MB. This limit makes it impossible for a solution for a system bigger than 100000 particles to have a complete representation in the device memory.

In cases where the problem can't reside completely in the device a memory a memory transfer stage between the host and device is needed. The Figure 6.26 shows the computational time cost of transfering the same amount of memory in different fragment sizes.

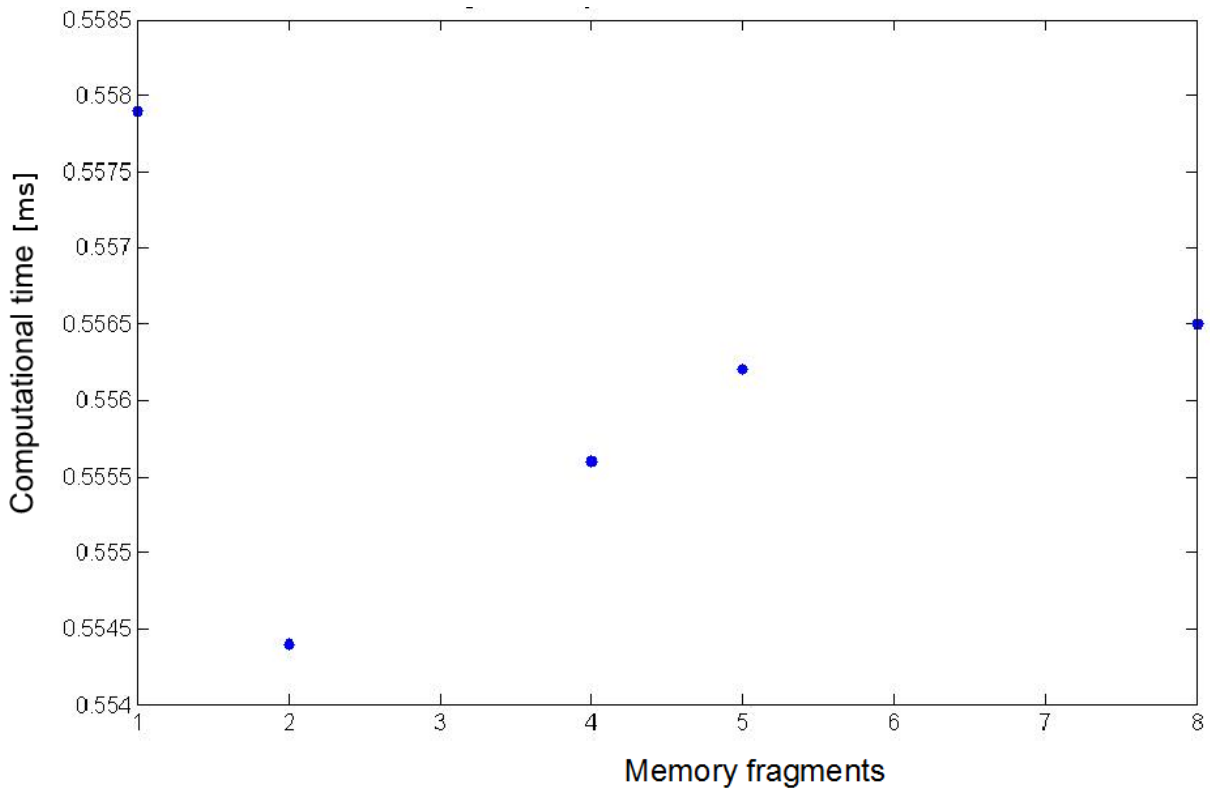


Figure 6.26: **CUDA memory transfer time** - The Memory transfer process from host to device and device to host is an important consideration when building the parallel implementation of SPH.

From the results of the memory test, the best performance was achieved by transfering 500 Mb at a time, this result could be associated with the architecture of the 500 gtx GPU generation. The average memory transfer time is 556 ms with a standard deviation of 1.26 ms

The first approach to managing the amount of memory transfer was to modify the SPH serial implementation. The particle structure was divided into individual arrays, where each array provides the memory for a physical variable. With this approach each one of the kernels on CUDA is required to call the amount of variables needed to solve the physical component needed by the kernel reducing the amount of memory transferred to the device.

After this new SPH code has been implemented there are three different problems types that can appear:

- 1: The whole problem fits into the gpu memory, in this case there is only one memory transfer from

the host to the device at the beginning of the simulation. Texture memory is used to connect with OpenGL for the display.

- 2: The memory require to solved the problem is bigger than the gpu capability, a memory transfer between host and device for each kernel module is needed and at the end of the time step a transfer to the host is done to update the particles properties with the new values. The host is used as the control environment providing the particles states for each time step.
- 3: The worst case scenario with this approach is when a kernel needs more memory than the capabilities of the GPU, for this problems multiple memory transfers will be needed so the kernel solved the particle system in segments.

Cases 1 and 2 are solved by using an array per variable but case 3 needs a different approach, in Section 6.7 an implementation to solve this problem is presented.

6.6.2 CUDA implementation results

To test the computational time improvement with the use of the parallel SPH solution the solid-fluid test shown on Figure 6.2 was implemented in parallel. The test was implemented on a computer with a GTX-580 GPU and a Pentium i7 CPU running on a 64 bit system (Microsoft Windows 7).

Test1:

The first test consists of a solid object with a particle dimension of $[5, 5, 5]$ particles and a fluid object with $[10, 10, 10]$ particles, the test is run during a time large enough to have a collision interaction between the objects, Figure 6.27.

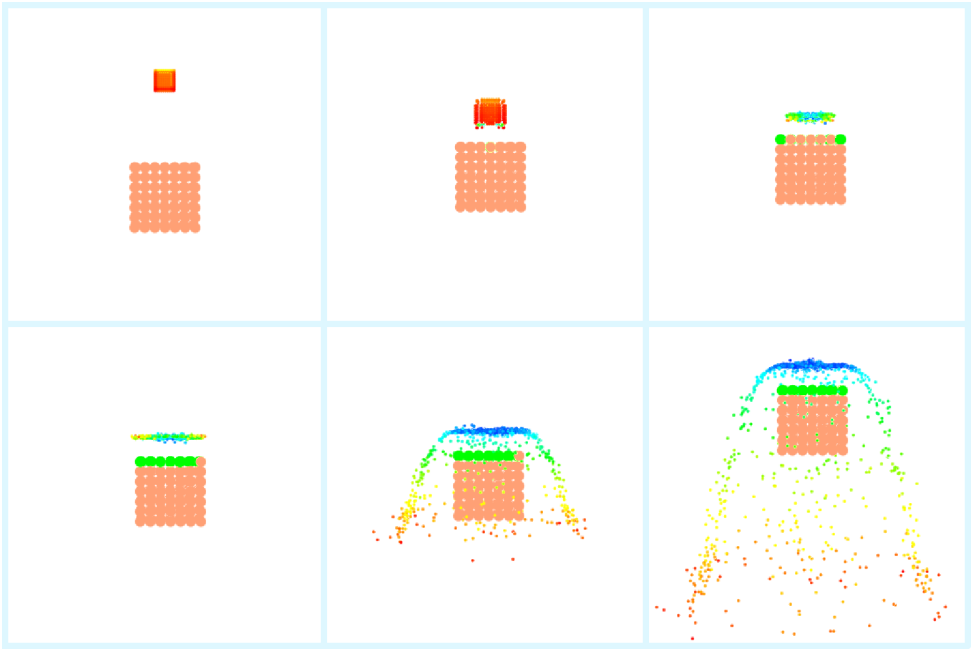


Figure 6.27: **Parallel - serial test** - The simulation is run for different particle number, all the simulations start with both objects separate by the same distances, the fluid is pull by the gravitational force and the simulation is run until the interaction solid - fluid is solved.

The result for the test is shown in Figure 6.28. For the first test the serial implementation took 16.5 s to solve 4000 steps and the parallel solution took 49.4 s for the same amount of steps. When the number of particles is low, the computational time taken to transfer memory is greater than the improvement obtained with the parallel solution.

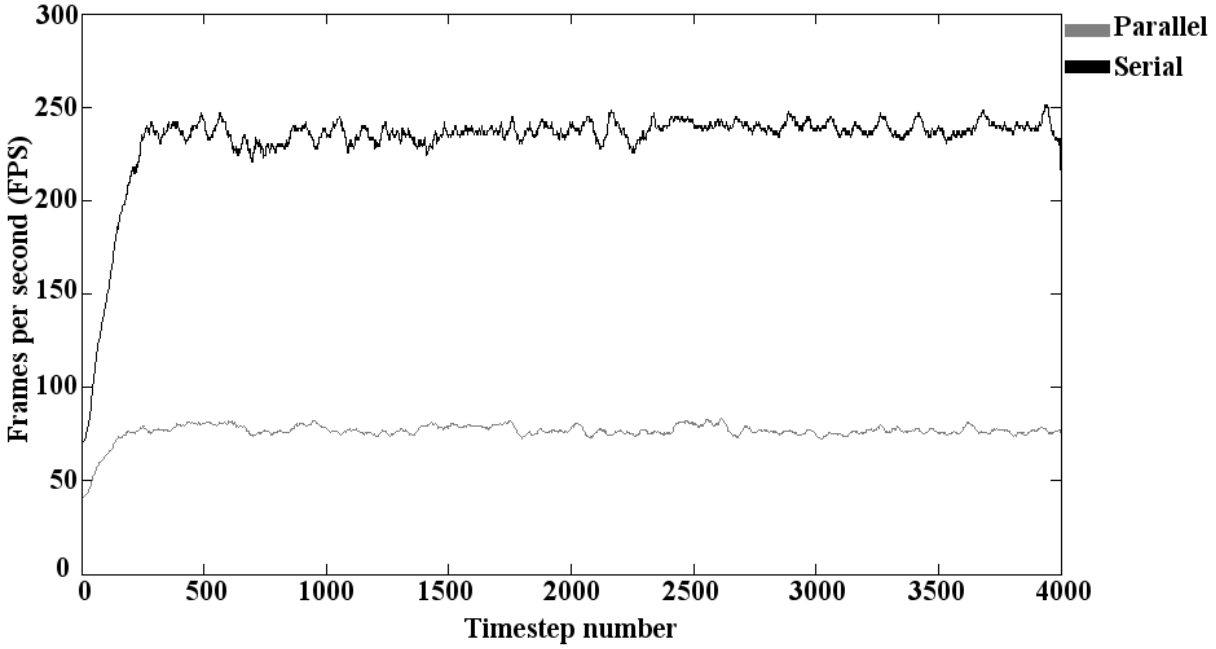


Figure 6.28: **Parallel - serial test 1** - The result shows that the serial performed better for 1125 particles when memory transfer is considered.

Test2:

The next test increases the dimensions of the fluid particles to $[20, 20, 20]$, in this case the parallel performance is better than the serial. The decline in performance between 500 and 1000 time steps is due to the solid - fluid collision and serves to demonstrate how even during these conditions the parallel implementation suffers less computational cost than the serial one. Figure 6.29 shows a plot of the fps for the test. For test 2 the computational time to solve 4000 timesteps were 110.52 s for the serial implementation and 78.50 s for the parallel.

Test3:

For the final test the fluid particles dimension increases to $[30, 30, 30]$ giving a total of 27125 particles. At this level of resolution for the fluid box the serial implementation suffers the most in the simulation. In Figure 6.30 it can be seen how for this test the parallel implementation outperforms the serial solution. The final computational time for the 4000 timesteps are 159.40 s in parallel and 415.70 s in serial.

6.7 Octree memory management

Following the process presented in Section 5.3, the Tree of tree structure can be used to solve the problem where the SPH implementation requires more memory than is available in the computer RAM or the

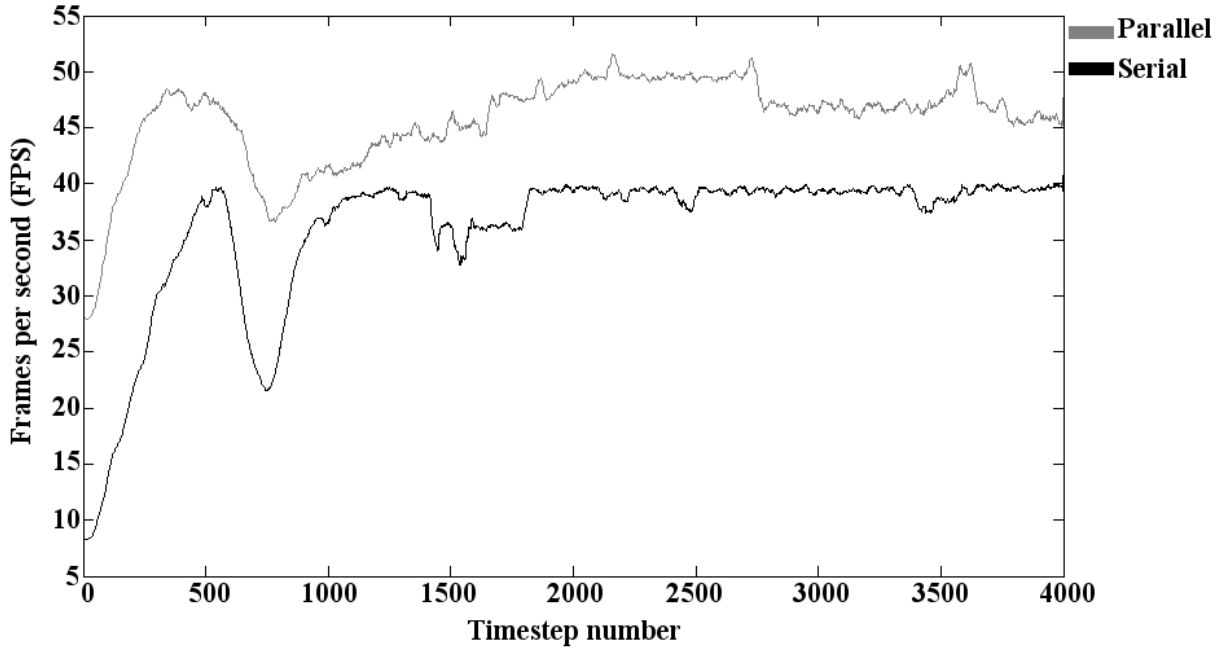


Figure 6.29: **Parallel - serial test 2** - The parallel implementation perform better for 8125 particles even when memory transfer cost between the PC and the GPU is consider in the computational time of the simulation.

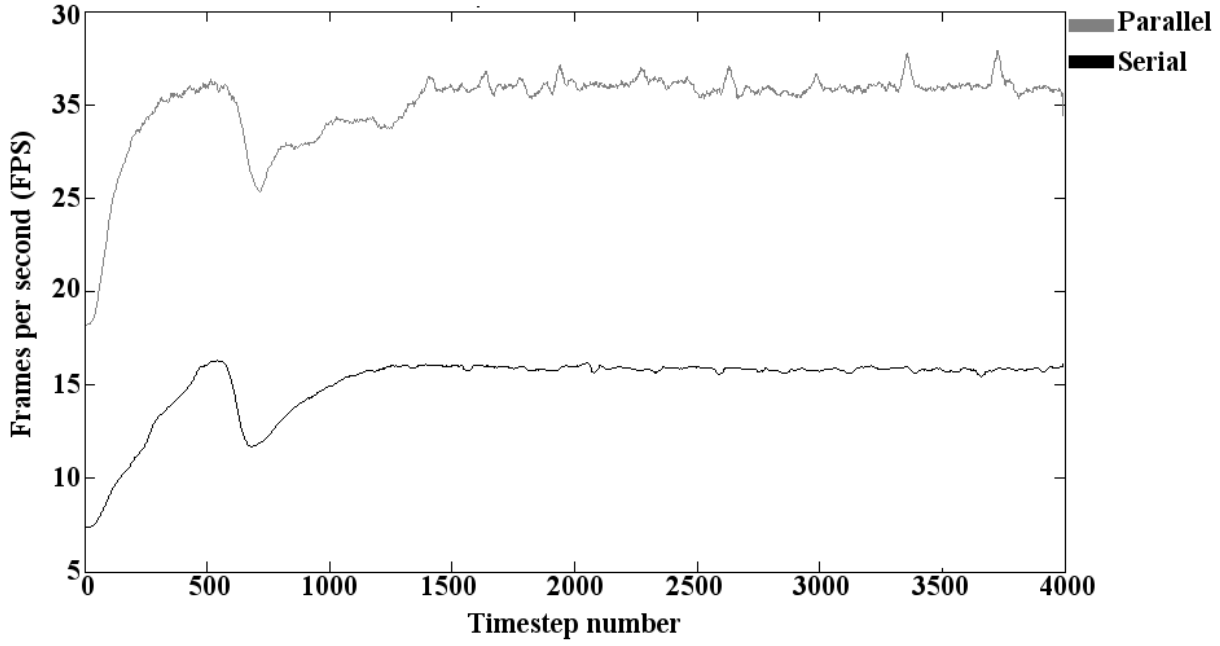


Figure 6.30: **Parallel - serial test 3** - The parallel implementation perform better for 27125 particles, the high number of operations needed make the parallel solution a better selection to solved the problem.

GPU memory.

To solve the problem of memory limitations an octree is implemented to subdivide the problem volume at each one of the leaf nodes (following the same rules as the tree of trees) will solve the particles belonging to it. This procedure requires a solution to the new conditions on the tree traversal:

- 1: The leaf tree should contain the necessary particles to solve all the interactions.
- 2: The number of particles inside leaf trees should approximate the memory limit to reduce the quantity of leaf trees needed to solve the volume problem.
- 3: At each time step all the leaf nodes that suffers an update need to save the updated state of the particles and free the memory so the next leaf can use it. In the next time step the leaf that was updated needs to read back the new particles inside its volume and solve the time step.

Figure 6.31 shows the computational time that the serial and the parallel implementations need to populate a tree.

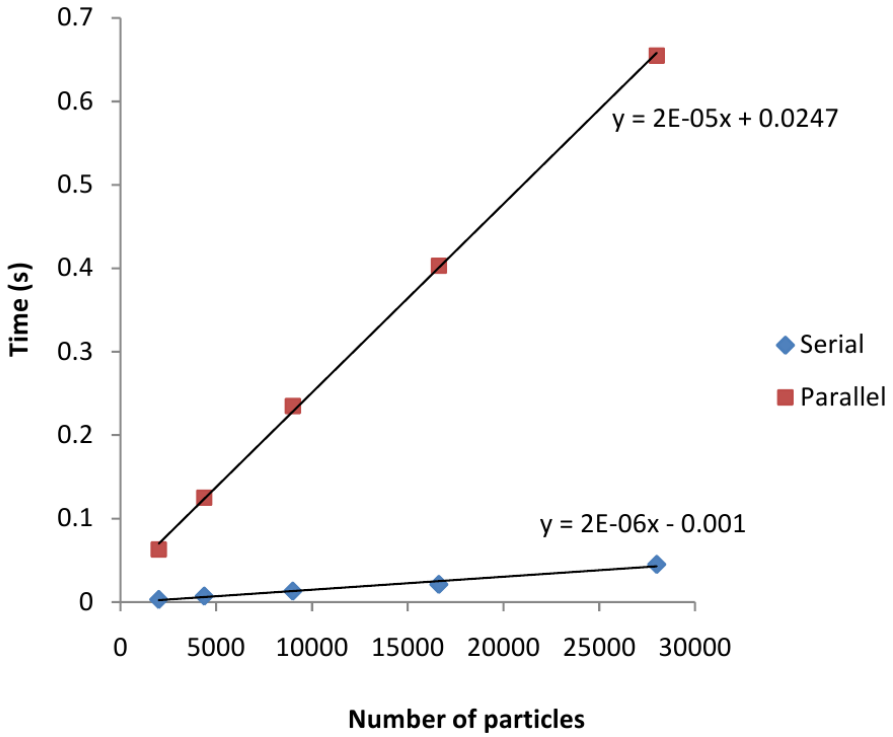


Figure 6.31: **Parallel - serial tree creation** - Computational cost of creating and populating a tree in the serial and parallel implementations, the computational cost of creating several trees at each time step is too high to be considered a proper solution.

Since a direct approach to the creation of the trees from data on hard - disk results in a high computational cost, a new approach was created, in this method the general tree Morton key, internal Morton key, index of the particles, and trees structure are kept in memory at all times.

A new kernel search is defined, in the case of the nodes the kernel is scaled to be equal to Equation 6.4:

$$Tt = \left(\sqrt{\frac{node_{width}}{2}^2 + \frac{node_{height}}{2}^2 + \frac{node_{depth}}{2}^2} \right) + (k * 1.2) \tag{6.4}$$

Tt scale factor is constant per simulation and is used to created a list of nodes that are neighbours of the focus node. The neighbour list for all the nodes and the Morton keys are saved in the node structure (the list is only created at the beginning of the simulation), this node structure serves to find the fastest way in which a node's location and the neighbourhood of a particle i inside the node can be found.

Next, a new flag is created by the particle, this flag tells the implementation when a particle changes a sub-tree, this keeps track of when the sub-tree system need to be updated.

Finally, using this new tree structure, the physic problem is solved from the top nodes in the space to the bottom, one row at a time.

In an 8 sub-tree problem the computational time to solve each node with a particle size of 1500 is equal to 24.0 ms per time step and the tree population per sub-tree in the node cost 2.0 ms for creating a new sub-tree. The read process from hard-disk to memory of the physical variables per particle using the particles index cost 5.0 ms per sub-tree for 1500 particles.

Figure 6.31 shows the current advantage of our serial implementation and at this point tree creation and population is kept outside of the parallel environment.

6.8 Conclusions

The SPH implementation was coded at the Intelligent Vision Systems New Zealand laboratory, the original code was based on the solution presented in [5] but due to the nature of the experiments conducted it was necessary to code the SPH implementation with a new method. The equations that form the solution of SPH are based on the work presented in [6] and are introduced in Section 2.1 and Chapter 3, during the development of the SPH implementation the following modules were added to extend and add new capabilities to the solution:

- In the implementation presented in [5] a pair-wise solution is used, this is the common approach to the SPH implementation. Since the serial SPH implementation is connected to the CUDA implementation the pair-wise solution was removed from the SPH implementation since in the CUDA solution each thread is in charge of updating one particle and accessing memory from a particle on another thread is not possible without using atomic operators.

  Atomic operations are special operations on the CUDA architecture that are performed without interference from any other threads. These operations are used to avoid the race condition (a race condition is an error on the process output and/or result due to the unexpectedly and critically dependence on the sequence of other events on the algorithm).

  One common example of a race condition on CUDA is presented when two different threads $thread_1$ and $thread_2$ need to increased the value of a memory location, for example the memory location 00012 with value equal to 7. In order for the threads to be able to update the value they will need first to read the value at the memory location 00012 and there is a possibility that both threads read the value at the same time, so both threads $thread_1$ and $thread_2$ read the value 7, then both threads add the value 1 to increase the value in the memory location, at this point the value in $thread_1$ will be 8 and the value in $thread_2$ will also be 8 so both threads will write to the memory location 00012 the value 8, this is incorrect the value was only incremented once when in reality two different threads operate over it, this is what is called the race condition.

  When using atomic operators it can be guarantee that when a thread is operating on a specific memory location the other threads on the kernel will not operate on it until the original thread finished using the memory location.

The cost of using atomic operators can be seen on the performance of the parallel implementation, since the threads need to wait for a memory location to be free before continuing making operations with the values on that location. To reduced the number of atomic operators the pair-wise approach to solved the particles interactions is substitute by an octree solution and a linked-list method to provide each individual particle with the necessary information to solved its behaviour.

- The solid implementation used part of the work presented in [103], the solution is extended to solve visco-elastic solids and an inverse kinematic method is introduced to connect the implementation with the Stereo-vision system. This produce a new tool that can be use to describe solid materials.

- The Morton key octree implementation for neighbourhood search is based on the work published in [8]. The solution was extended to implement a Tree of trees solution and the parallel solution implemented in this Thesis used a new method based on the introduced ghost nodes and CUDA architecture to solve the parallel problem.

- The dynamic update of the tree, neighbourhood, and collision list are part of the most computationally expensive modules of the implementation. The kinetic flags based on the Morton key of the particles and tree nodes are used as discriminants in the SPH implementation to track the necessary updates.

- The CUDA parallel implementation was developed at the Intelligent vision systems New Zealand laboratory as a method to obtained the fastest computational time for the soil experiments since the number of particles needed to solve the problem made the serial computational time for the solution too high.

  The serial implementation was modified to enable a more efficient connection between the serial and parallel solution and reduce, where possible, the cost of the memory transfer.

This chapter presented the different performance enhancements introduced with each one of the methods implemented in the SPH solution.

An important point obtained from the performance study is that the different modules: linear serial solution, linear octree solution, CUDA parallel linear solution and the parallel octree, have different advantages and depending on the number of particles and the scale factors, the SPH implementation will use the appropriate method for each problem.

The computational efficiency of the SPH implementation is only one side of the problem, the most important consideration is how accurate and flexible the solution is. The chapter discussed how the SPH implementation can be connected to problems of fluid flow inside solid boundary walls and how the computational simulation provided an efficient method to obtain solutions for different conditions, the application of this method is presented in Chapter 7 where the input solid boundaries are obtained from CT-Scan data.

Another example of the flexibility of the implementation is the ability to use the SPH algorithm to solve the inverse kinematics of the strain data provided by a stereo-vision solution, with this method it is possible to obtain the material properties of the object under study.

Finally the chapter discussed the memory management limitations with the different methods. One of the advantages of using an octree solution for problems with memory requirements greater than the available memory is that it allows us to pre-compute a divided volume problem that can be solved independently of the general solution for a set of time-steps.

The final SPH solution is being used in the Intelligent Vision Systems New Zealand laboratory as a computational method to solved physical problems. The advantage of this collaboration is that vision implementations and image processing solutions provide a database and structural information for different problems and materials.

Chapter 7

PORE EXPERIMENT

7.1 Introduction

Studies of soil core poral structure is important for simulating preferential flows, which is essential for understanding the leaching of chemical compounds in underground aquifers. It is particularly important in the management of water resources and the monitoring of water quality.

Solute transport (through rain and/or irrigation water) through soils is widely researched and there is strong public interest in monitoring water resource contamination by chemicals from agricultural, pharmaceutical and industrial sources [104].

Most solute transport models assume a simplistic homogeneous soil structure where the morphological component of the boundary condition is not taken into account. In reality, the fluid flux is distributed heterogeneously and is dependent on the non-uniformity of soils poral structure [105].

In order to obtain the three-dimensional model of a core sample (Morphological properties) an X-ray computed tomography(CT-Scan) is used to obtain a database containing the 2D-cut images. The database is processed using a set of imaging techniques to label and segment the 3D porous structure of the sampled core.

When the pore structure is obtained the SPH Navier-Stokes implementation is used to model the water transport in the reconstructed 3D pore network.

7.2 Data acquisition and processing

The 3D description of the soil characteristics is represented by a soil porous network computational model. The data for the model is obtained using a CT-Scanner to obtain two-dimensional cuts from the undisturbed soil cores of the porous structure geometry [106].

The CT-Scan delivers the internal structure of the scanned object in a series of 2D slices encoded as 16 bits per pixel data (often with 10 to 14 effective bit encoding per pixel).

The pixel grey value represents the attenuation of the X-ray beam from the CT-scanner. The amount of attenuation is related to the density of the material components traversed by the ray. Each pixel of the image represents a discrete value of the total attenuation reconstructed from a set of X-ray beams [107, 108].

In the soil samples the principal material components are void, plastic resin, stones, and aggregates. The different on the densities of the different material components allows the classification of the material

on the CT-scan images using a grey-scale value representation. One example cut resulting from a CT-scan of a poral structure is shown in Figure 7.1.

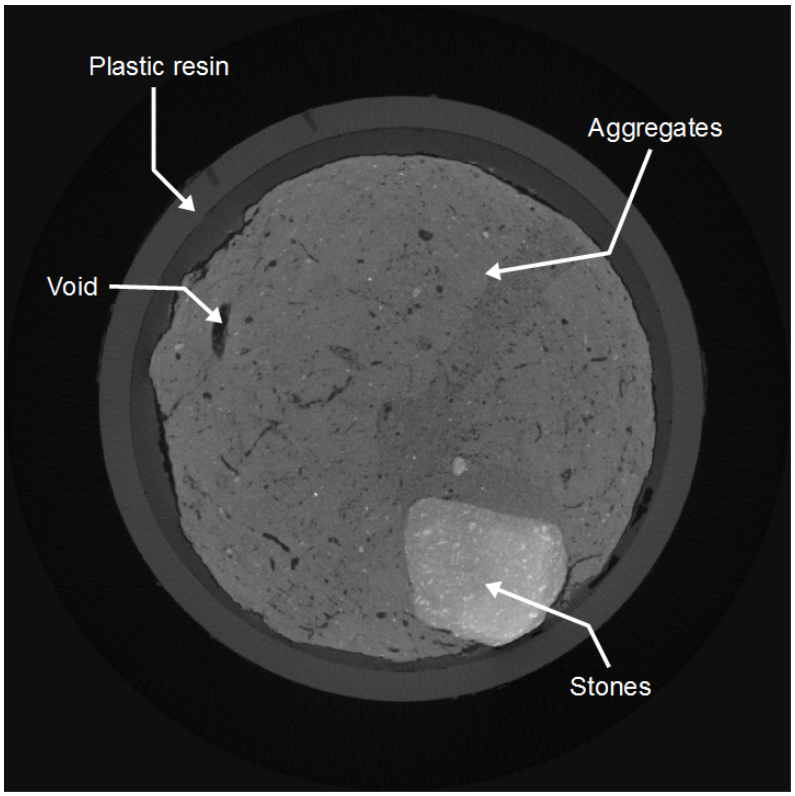


Figure 7.1: **CT-scan cut with sample components** - Selected two-dimensional cut from the database showing the components of the core with different grey scale levels.

7.3 Soil characteristics

The studied site for the obtention of the samples and obtain the results for the water and fertilisers transport study was an elementary water catchment called la Loma in Valle de Bravo watershed (Central Mexico) located in the Trans- Mexican Volcanic Belt [109]. The soil was characterised and classified as a Pachic Andosol by [110]. In Table 7.1 is presented the physical and chemical characteristics found on [110] for the soil at a depth level of 80$[cm]$.

Andosol characteristics at 80 cm depth							
Sand %	Silt %	Clay %	pH	ρ_d $\frac{g}{cm^3}$	SOC $\frac{g}{kg}$	Ks $\frac{m}{s}$	η_p %
22	68	10	6.4	0.6	51	1.89×10^{-5}	0.75

Table 7.1: Main physical and chemical characteristics of the Andosol at 80 cm depth.

Where ρ_d is the bulk density, SOC the soil organic carbon, Ks the saturated hydraulic conductivity, η_p the total porosity respectively.

Disc infiltrometer measurements (0.2 m diameter) were performed in situ under unsaturated steady state under tension varying from -0.1 to -0.01 m. A hole was dug to reach the bottom layer, and a thin

sand layer on top of the soil ensured good contact between the soil and the base of the infiltrometer. We assumed that flux (q, mm/s) under the disc is given by the Wooding equation [111] and that the hydraulic conductivity K Equation 7.1 follows an exponential function with tension [112].

$$K = K_s e^{P_{head}\alpha_{soil}} \tag{7.1}$$

where K_s is the saturated conductivity mm/s, P_{head} the pressure head mm and α_{soil} a soil-dependent parameter mm^{-1}. Infiltration data were analysed to determine K_s (Table 7.1)

7.3.1 Soil database

The soil core was scanned by X-ray tomography with the CAT scanner at the 3SR laboratory in Grenoble. The CT-scanner generates multi-energy X-rays with energy between 40 and 230 kV, the distance source-detector (SDD) of the scanner is 0.752 m and the distance source-object (SOD) was 0.347 m.

The scan of the core sample took approximately 45 minutes. The final database is called Loma-80 and is formed by 1669 2D image cuts. The cuts in the database are named numerically from P23-80_00121 to P23-80_01789. The spatial resolution per voxel is (8.7×10^{-5} m, 8.7×10^{-5} m, 8.7×10^{-5} m), each pixel value is stored as a 16 bit entity with an effective 14 bit dynamic range (The final 2 bits are reserved as free space for labelling), the 14 bit dynamic range provides 16384 different grey values.

RX solution was set with a spatial resolution (pixel size) of 87 μm. With these parameters, the images obtained showed a satisfying contrast between voids (air or water) and soil solid matrix. Once scanning completed, the 3D soil column data set is available as a series of 1500 2D images. Image pixels are stored as 16 bit entities with an effective 16 bit dynamic range, the total number of grey values is:

$$\text{grey levels} = 2^{16} = 65536 \tag{7.2}$$

The histogram is shown in Figure 7.2. The histogram shows the full use of the dynamic range, with the problem neither normalised or equalised. This shows that the range of grey values (45746 to 65535) that forms 30.2% of the dynamic range values only have a count of 60 pixels (0.00261% of the total number of pixels), this problem will be addressed latter in the pre-segmentation process.

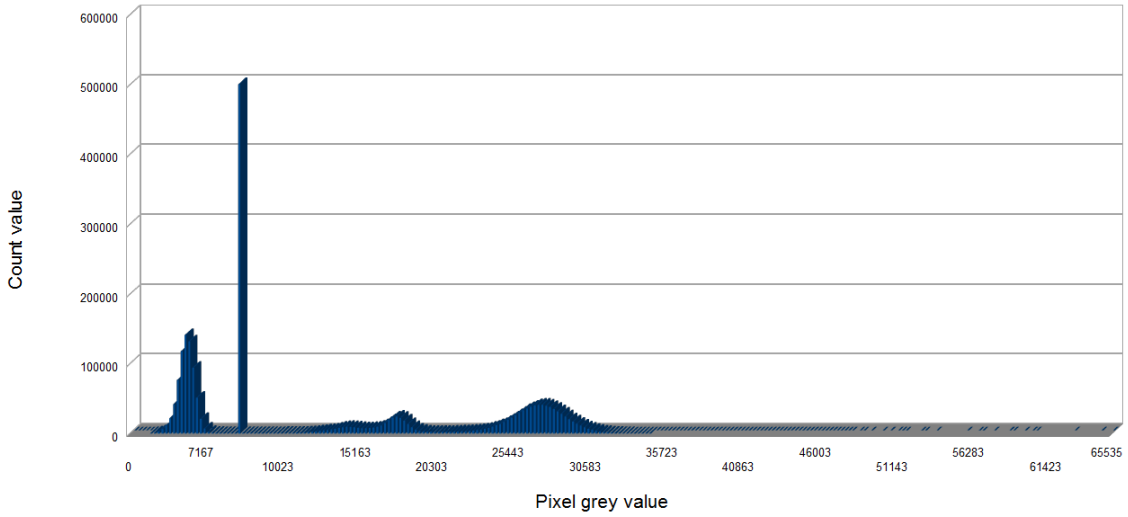


Figure 7.2: **CT-scan cut histogram** - The histogram from a cut sample of the database shows the use of the full dynamic range from 0 (black) to 65535 (white).

7.3.2 Soil cut image processing

In order to generate the soil porous network computational model, we first need a 3D description of the pore soil boundary characteristics.

The first step consists of segmenting the pore boundary walls from the rest of the image (that will be considered background).

The CT-scanner at the 3SR laboratory in Grenoble is an experimental CT that had not been previously calibrated and suffered from instability in the energy of the beam. This caused problems that required a solution before the analysis could be performed.

The first problem is a difference in the brightness level of pixels with the same density level in the sample, this is due to the inherent shadowing produced by the CT reconstruction and energy fluctuations. This caused the second problem, a change in the dynamic range of the images. Due to these problems a references frame to normalized all the images was selected. The histograms have four principal components defined by peaks (Figure 7.3).

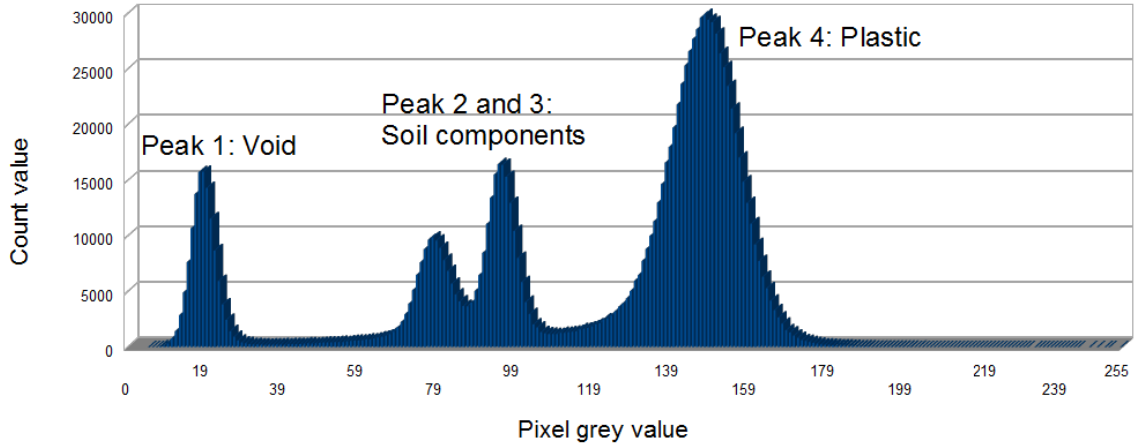


Figure 7.3: **CT-scan cut histogram** - Histogram peaks from a cut sample of the database. The peaks are related to the different components of the image cut. The segmentation requires a clear differentiation of peak 1 from the other peaks.

The intensity values are related to the density of the voxel. The aim of the experiment is to segment the pore wall structures. In perfect conditions the voids inside the pores will be defined by a density equal to 0, but due to noise and resolution errors the intensity values only approximate 0. Figure 7.4 shows the differences in the histograms of a set of images, the histograms are normalised to bins from 0 to 255.

To correct the differences in dynamic range the four peak positions $\mathbf{Pk_i}$ for the Ni images i on the databases are obtained, then the average is calculated to obtain the mean position of the peaks $\overline{\mathbf{Pk_i}}$.

$$\overline{\mathbf{Pk_i}} = \frac{\sum_{j=1}^{Ni} \mathbf{Pk_i}}{Ni} \tag{7.3}$$

The mean histogram $\overline{\mathbf{Pk_i}}$ is used to calibrate the principal peaks of all the histograms with respect to it. Each peak is assumed to represent the same material density in each cut, so the calibration ensures that the same principal density belongs to the same intensity peak for all the images.

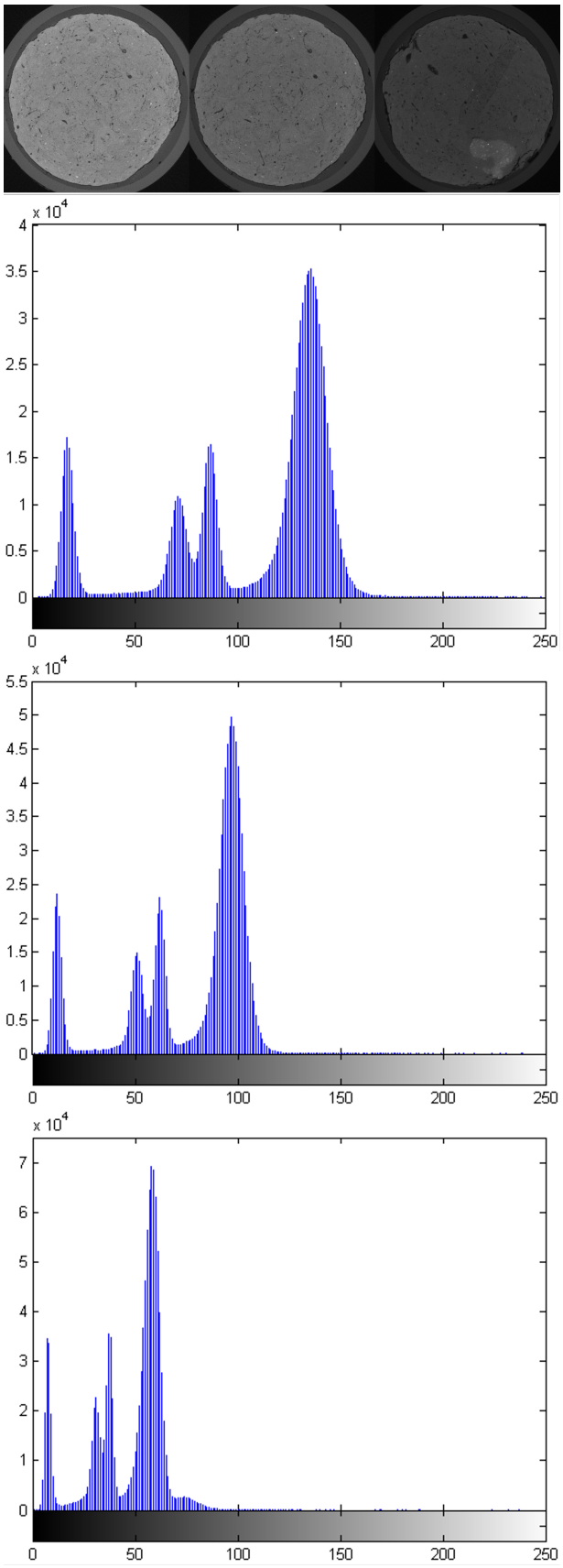


Figure 7.4: **Pore CT-images histogram** - CT-images with their respective histograms. The histograms share the same characteristic peaks but their dynamic ranges are differents.

This technique is only used to segment the void - pore walls from the rest of the image, it does not provide a calibrated density image. The aim of the technique is to fix the dynamic range; the final peaks are not related to real density values, they are only related to overall density differences.

The histograms after rectification are shown in Figure 7.5, in this image the vacuum inside the pore wall is shown in dark blue, light green to dark green is used to represent pore volume, and the red component is vacuum outside the pore volume and plastic.

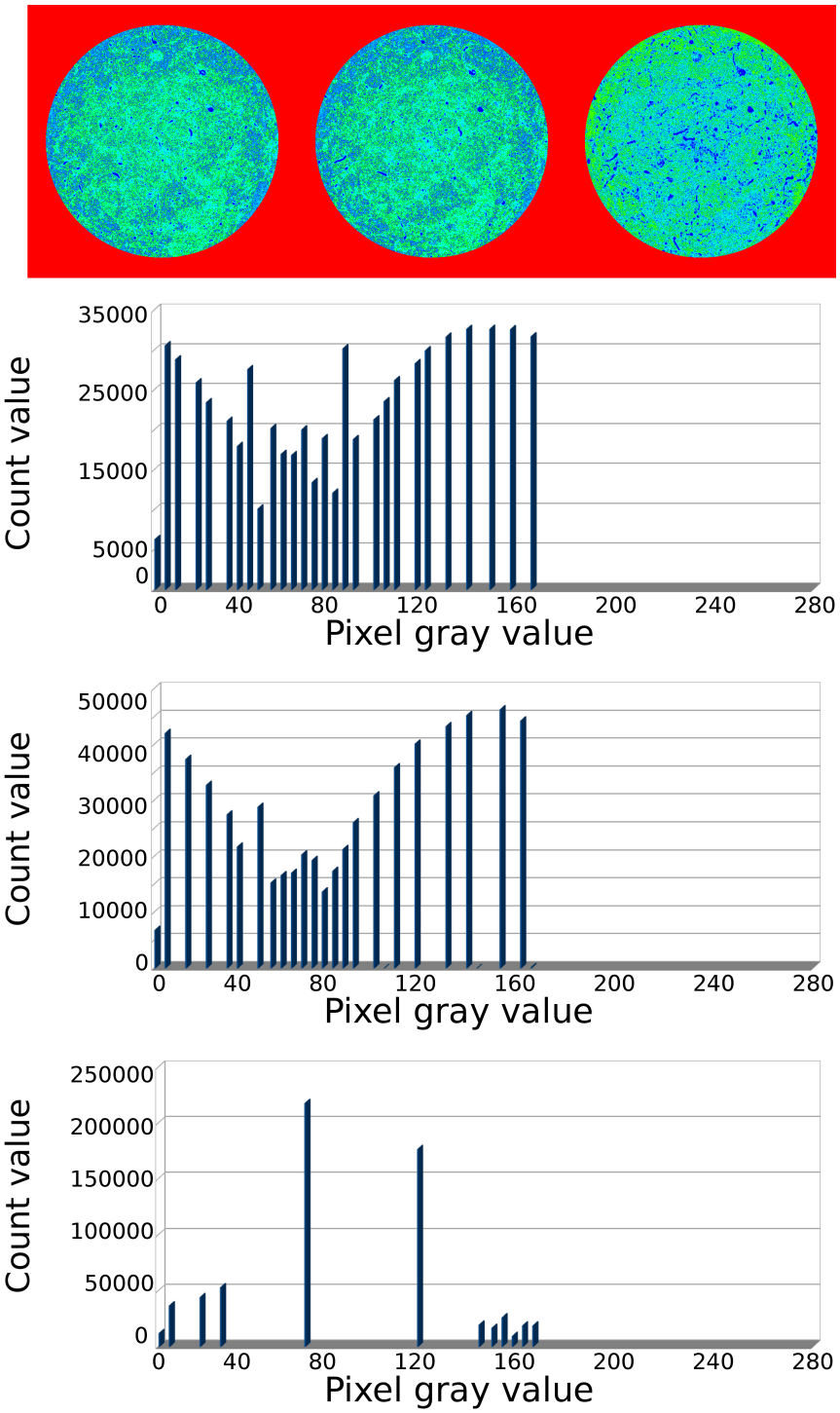


Figure 7.5: **Dynamic range corrected histogram** - The histograms have been calibrated against the mean average. The volume inside the pore walls is described by the blue areas.

The pores are segmented from the resulting images using simple histogram thresholding and the pore walls defined with a Deriche edge extractor [113] (Figure 7.6).

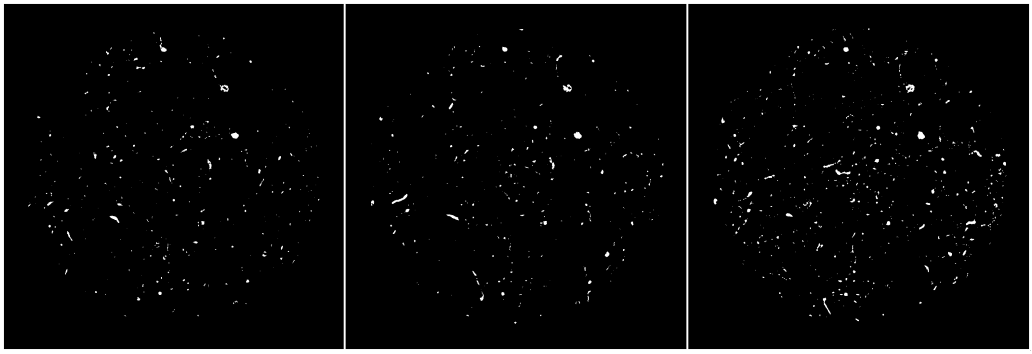


Figure 7.6: **Pore void segmentation** - After segmentation the void inside the pores is shown in white. The 3D boundary model will be obtained from these images.

After all the image histograms have been calibrated and the walls segmented, the database is used to calculated the 3D representative elemental volume (REV) of the core sample.

7.3.3 3D representative elemental volume

In order to obtain the pore structures for the modelling, a representative elemental volume (REV) is chosen from the binary database. The REV volume is defined in the literature as the smallest volume in which the measurements of the physical properties of the pores and flux are representative of the whole volume [114].

In the work presented in [114], the REV is calculated with a two-dimensional mask over a plane. Since the pore network morphology has three-dimensional dependants the REV calculation should be made in this space.

The initial REV centre was first selected by randomly finding centres of cylindrical volumes of at least 6000 voxels. This minimal allowed volume was set (as the average volume of the top 10 largest pores in the soil core) to avoid selecting REV with a high enough porosity but with only very small and/or disconnected pores.

Eliminating centres too close to each other or which generated asymmetric cylindrical structures (e.g. as too close to the soil core boundaries) left about 10 potential centres. Considering the effective 1500 slices of the soil core, the initial REVs centre was positioned at coordinates x: 534, y: 527 and z: 1199 (e.g. at the centre of the 400 slice cylindrical volumes).

The REVs were given cylindrical geometry to cater for the soil core shape. REVs dimension was expanded by a step of 2 pixels in the x, y, and z directions. At each optimisation step, the porosity i.e. the ratio of the number of void pixels with respect to the total number of pixels (total volume) Equation 7.4, for each potential REVs was calculated.

$$Porosity = \frac{pore\_volume}{total\_volume} \tag{7.4}$$

This process was maintained until the REVs porosity reached a constant value which should correspond to the soil core porosity at the CT-scan resolution. This optimisation process took into account REV centre, porosity, pore size distribution and variability in pore characteristics. The selected REV (out of the 10 initial seeds) holds the largest connected pores which will be used for our simulation experiment.

Figure 7.7 presents 10 different seeds with different start positions on x, y, z and shows how the ratio changes as the volume of interest increases.

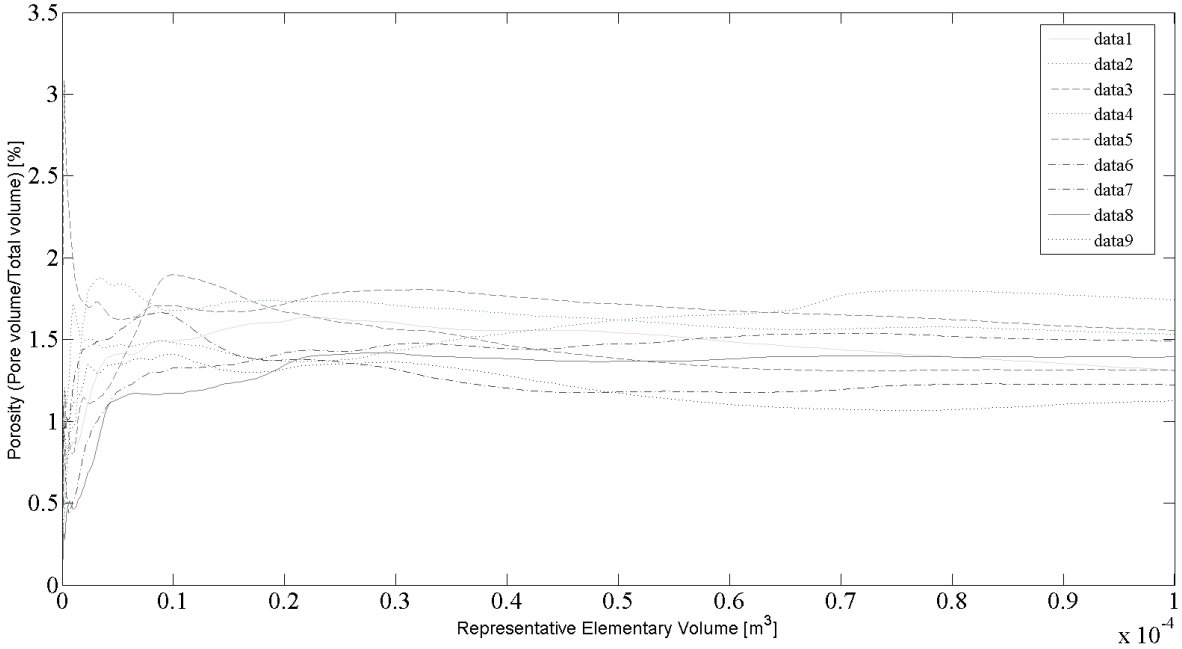


Figure 7.7: **REV graph ratio** - The graph shows the % ratio between the void inside the pore walls and the total volume. After a volume size for the REV is reached the behaviour of the ratio tends to 0.

The data point properties are presented in Table 7.2:

Figure 7.7 shows how the volumetric porosity (at 0.087 mm resolution) varies when increasing the volume of interest. The porosity becomes constant (at a value of 1.5%) for a cylindrical volume with height and diameter of 400 pixels ($= 3.48 \times 10^{-2}$ m for a pixel of 8.7×10^{-5} m), i.e. a volume of $3.031 \times 10^{-5}[m^3]$.

The new REV database is formed by binary images with xy-dimension (209 by 209) and with 400 cuts in z (Figure 7.8).

The new REV database contains the voxels that will be used to obtain the 3D pore volume.

Seed initial properties for databased P23-80				
Legend	x centre coordinate	y centre coordinate	z centre image	Diameter range
1	334	334	1199	d:2:402 z:00999 to 01398
2	334	534	1199	d:2:402 z:00999 to 01398
3	344	734	1199	d:2:402 z:00999 to 01398
4	534	334	1199	d:2:402 z:00999 to 01398
5	534	534	1199	d:2:402 z:00999 to 01398
6	534	734	1199	d:2:402 z:00999 to 01398
7	734	334	1199	d:2:402 z:00999 to 01398
8	734	534	1199	d:2:402 z:00999 to 01398
9	734	734	1199	d:2:402 z:00999 to 01398

Table 7.2: REV initial seed data points properties, 10 seeds are planted in the plane of image 1198

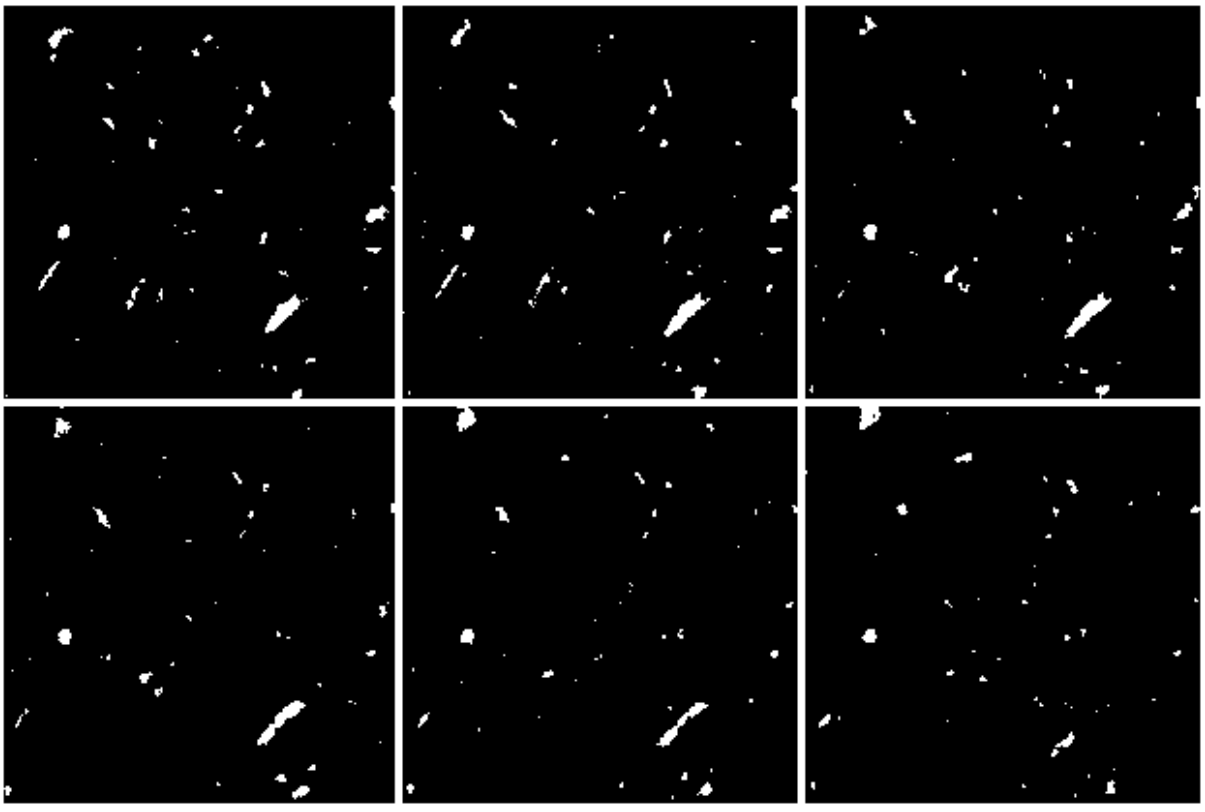


Figure 7.8: **REV binary images database** - The database is used to build the 3D pore structure.

7.3.4 3D pore structure

Using the REV volume database and a marching cuber algorithm [115] the REV 3D pore representation is obtained. Figure 7.9 shows the REV pore structure from different view angles.

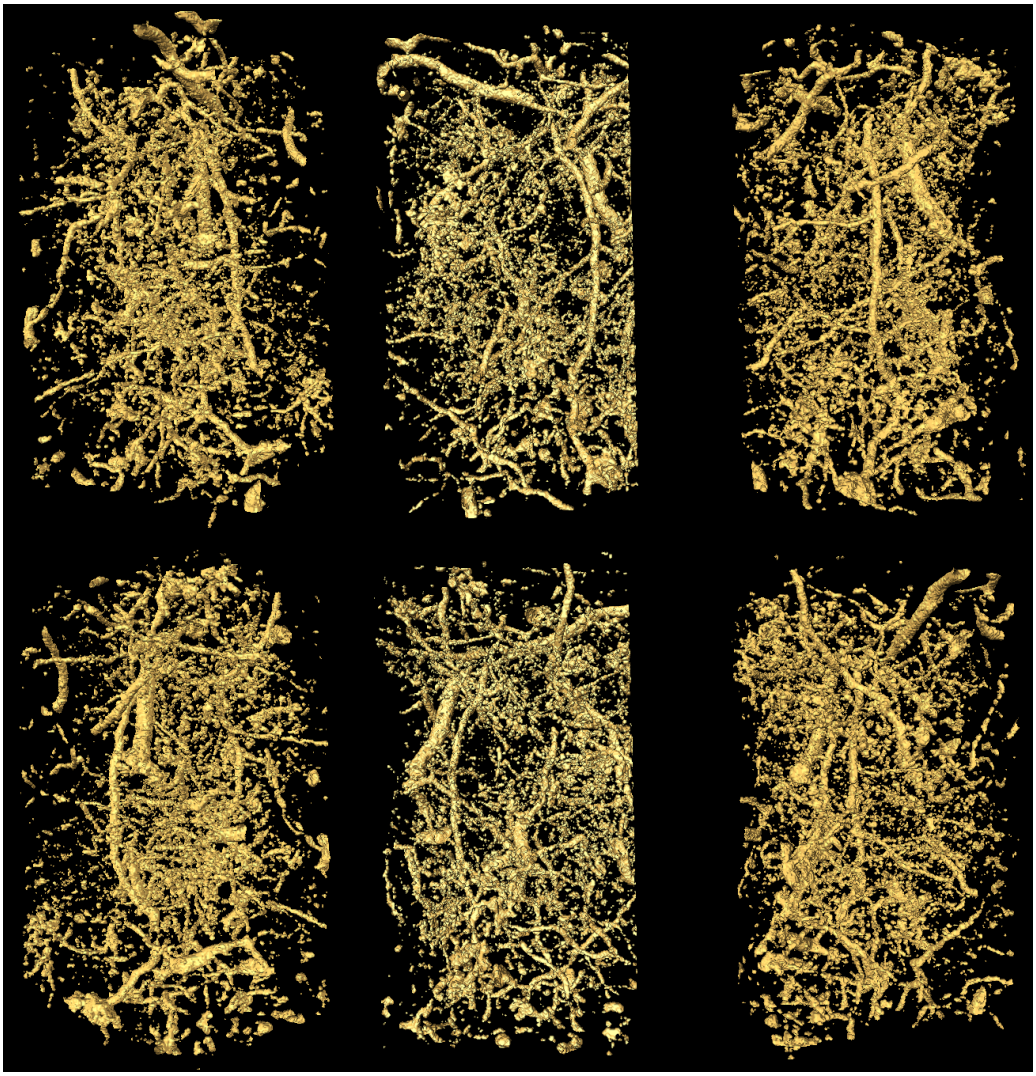


Figure 7.9: **3D pore network of the REV** - The 3D pore network representation contains the walls of all the pore components, the figures shows the 3D REV from different angles.

The 3D pore structure inside the REV is used to select the pores to be studied.
A pore of interest is selected using the following set of conditions:

- **Voxel Volume Size:** The pores should have a number of voxels greater than 6000. This is done to remove all the small pores and non connected components in the database (formed by only a few voxels).
- **Pore Diameter:** The second condition is the diameter of the pores. Inside the REV there are pores with closed pathways where flux is impossible. The pores selected have a complex internal structure with diameters that are not constants.
- **Tortuosity:** Pores with different levels of tortuosity are selected to analyse the different effects on the flux of the fluid.

For the experiment, 3 pores were selected and named pore-3, pore-6, and pore-9. The pores were labelled with the aid of the Amira multi-physics ver. 5.3.3 (http://www.amira.com/) program.

The labelling process is based on the 3D connected components technique. Figure 7.10 shows a visual representation of the pore selection and labelling.

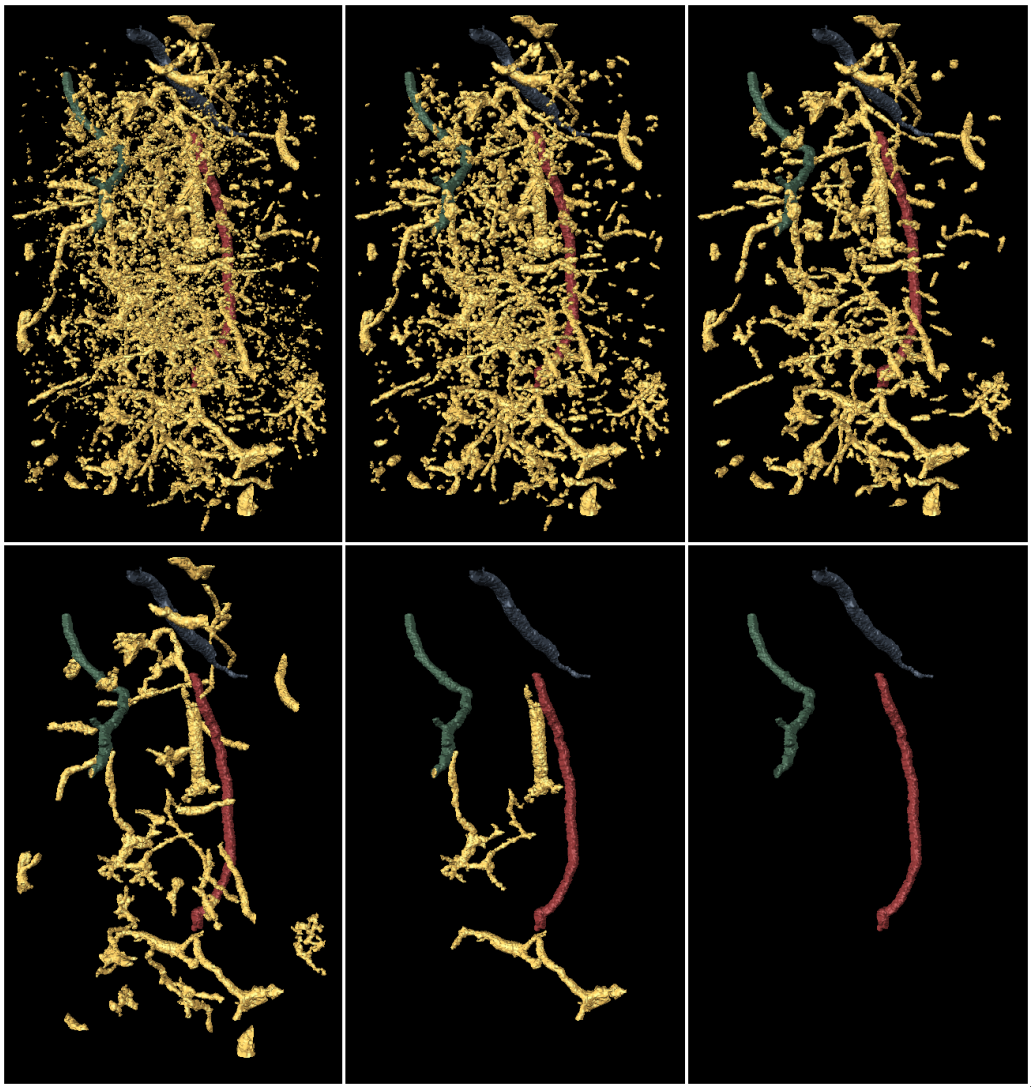


Figure 7.10: **3D pores selection** - 3D pore selection with the aid of the connected components techniques that segments the pores with the bigger voxel volume from the rest. The final three are selected considering the requirements for the simulation. The red pore is pore-3, the green is pore-6, and the blue one is pore-9.

7.3.5 3D pores morphological and geometric properties

The selected pore structures need to be analysed for their morphological and geometric properties.

In the following subsections the following pore properties will be presented for each one of the pores:

- **Pore skeleton:** The pore skeleton Po_{ς} is the Euclidean distance line that transverses all the centroids of the pore from the top cut to the bottom. The skeleton will be used to determine the tortuosity of the pore.

Po_ς is obtained using the original binary images. The centroid for each cut is obtained and then the distances between adjacent cuts are calculated.

$$Po_\varsigma = \sum_{i=1}^{i=c_N-1} d\left(\bar{\mathbf{c}}_i, \bar{\mathbf{c}}_{i+1}\right) \tag{7.5}$$

where c_N is the total number of cuts, $\mathbf{c_i}$ is the centroid value of the cut i and the operator $d\,()$ is the Euclidean distance.

- **Pore length:** Po_ℓ This is the lineal distance between the top voxel (voxel closest to the corner $(0,0,0)$) and the bottom voxel (voxel closest to the corner (max_x, max_y, max_z)).

- **Tortuosity:** The tortuosity Po_τ is defined as the ratio between the longest path within a pore, often referred as the geodesic length, and the pore length, i.e. the Euclidean distance between the pore extremities. The pore geodesic length is obtained by computing the pore skeleton length [116].

$$Po_\tau = \frac{Po_\varsigma}{Po_\ell} \tag{7.6}$$

- **Pore inlet area:** Po_α is the total area of the flow entrance cut. The pore inlet is defined by the number of pixels in the binary top cut image multiplied by the pixel area.

$$Po_\alpha = Pc_\alpha \cdot pixel\_area \tag{7.7}$$

Where Pc_α is the number of pixels in the inlet cut.

- **Pore outlet area:** Po_β is the total area of the flow exit cut. The outlet pore area is defined by the number of pixels in the binary bottom cut image multiplied by the pixel area.

$$Po_\beta = Pc_\beta \cdot pixel\_area \tag{7.8}$$

Where Pc_β is the number of pixels in the outlet cut.

- **Pore area distribution:** Besides tortuosity, another important flow restriction is how the area of the pore changes with depth. The pore area distribution is obtained using the area per cut for all the cuts that form the pore. The mean pore area and the standard area deviation is obtained from the distribution.

- **Pore diameter distribution:** The pore diameter distribution provides the effective per cut diameter. The value of the diameter for each cut is obtained by averaging the distances from the cut centroid to all the pixels of the contour.

- **Hydraulic radius:** The mean pore hydraulic radius of each pore was calculated as :

$$R_h = \frac{\bar{A}_s}{\bar{P}_s} \tag{7.9}$$

Where $\bar{A}_s$ is the mean pore area and $\bar{P}_s$ is the mean perimeter of each slice.

Pore-3

In this section the Pore-3 geometric and morphological properties are presented.

- **Pore skeleton:** The effective length of the skeleton is: $Po_{\varsigma} = 0.026\ m$. Shown in blue in Figure 7.11.
- **Pore length:** The pore length is: $Po_{\ell} = 0.022\ m$. Shown in red in Figure 7.11.

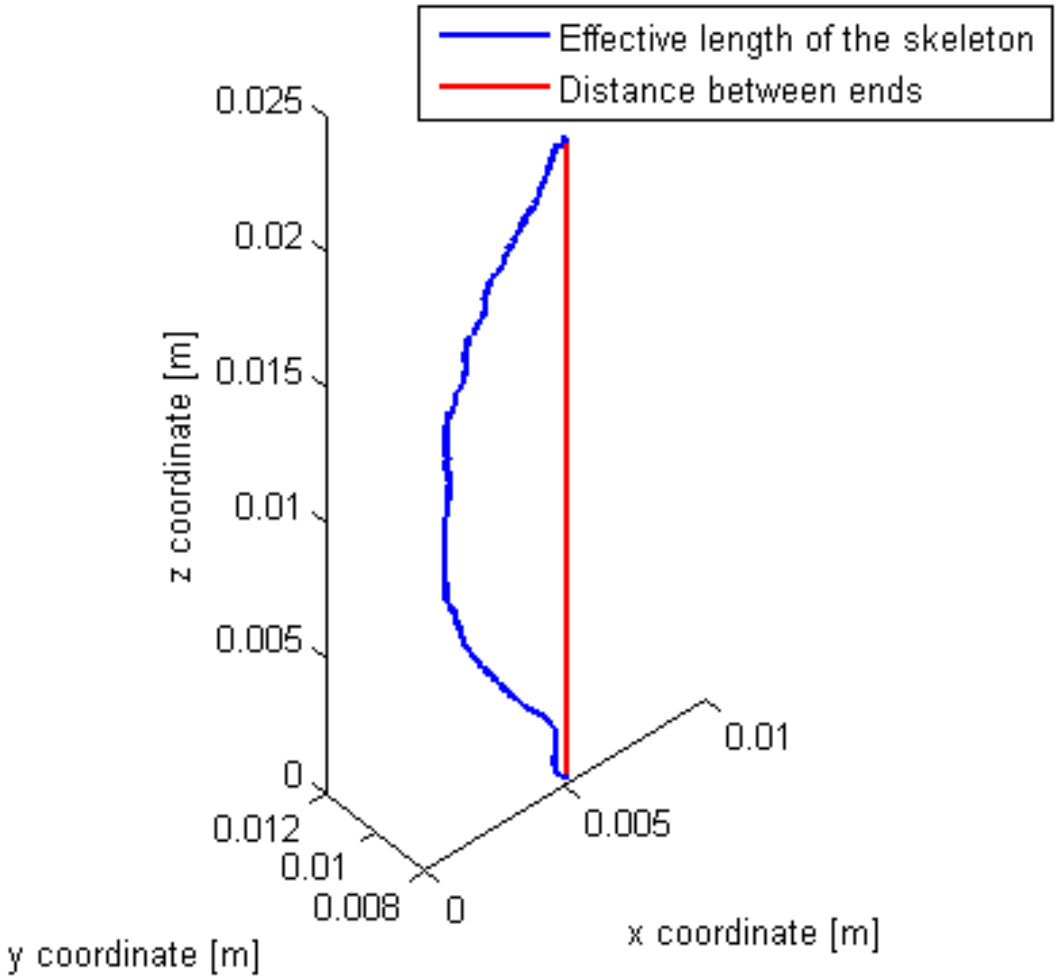


Figure 7.11: **Pore-3 Skeleton and length** - Pore-3 Skeleton and length. The ratio between these values gives us the tortuosity of the pore.

- **Tortuosity:** The tortuosity is equal to:

$$Po_{\tau} = \frac{Po_{\varsigma}}{Po_{\ell}} = \frac{0.026m}{0.022m} = 1.16 \tag{7.10}$$

- **Pore inlet area:** The inlet area is equal to

$$\begin{aligned} Po_{\alpha} &= Pc_{\alpha} \cdot pixel\_area \\ &= 18 \cdot \left(8.7 \times 10^{-5}m * 8.7 \times 10^{-5}m\right) \\ &= 1.36 \times 10^{-7}m^2 \end{aligned} \tag{7.11}$$

The top binary cut with the inlet area is shown in Figure 7.12.

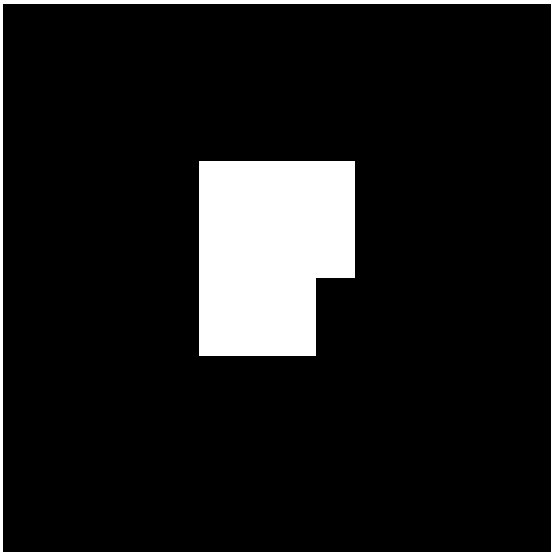


Figure 7.12: **Pore-3 inlet area** - Pore-3 binary cut showing the inlet area.

- **Pore outlet area:** The outlet area is equal to

$$\begin{aligned} Po_\beta &= Pc_\beta \cdot pixel\_area \\ &= 9 \cdot \left(8.7 \times 10^{-5} m * 8.7 \times 10^{-5} m\right) \\ &= 6.81 \times 10^{-8} m^2 \end{aligned} \tag{7.12}$$

The bottom binary cut with the inlet area is shown in Figure 7.13.

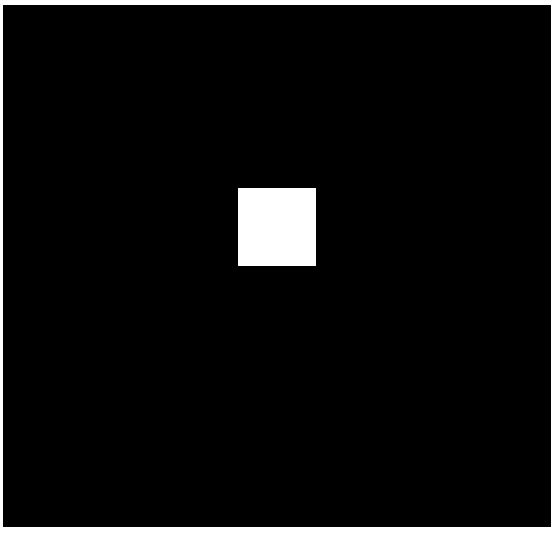


Figure 7.13: **Pore-3 outlet area** - Pore-3 binary cut showing the inlet area.

- **Pore area distribution:** Pore-3 mean area is 5.22×10^{-7} m with a std of 1.17×10^{-7} m

  The top image in Figure 7.14 shows the relation between area and depth, and the bottom image represents the frequency of areas.

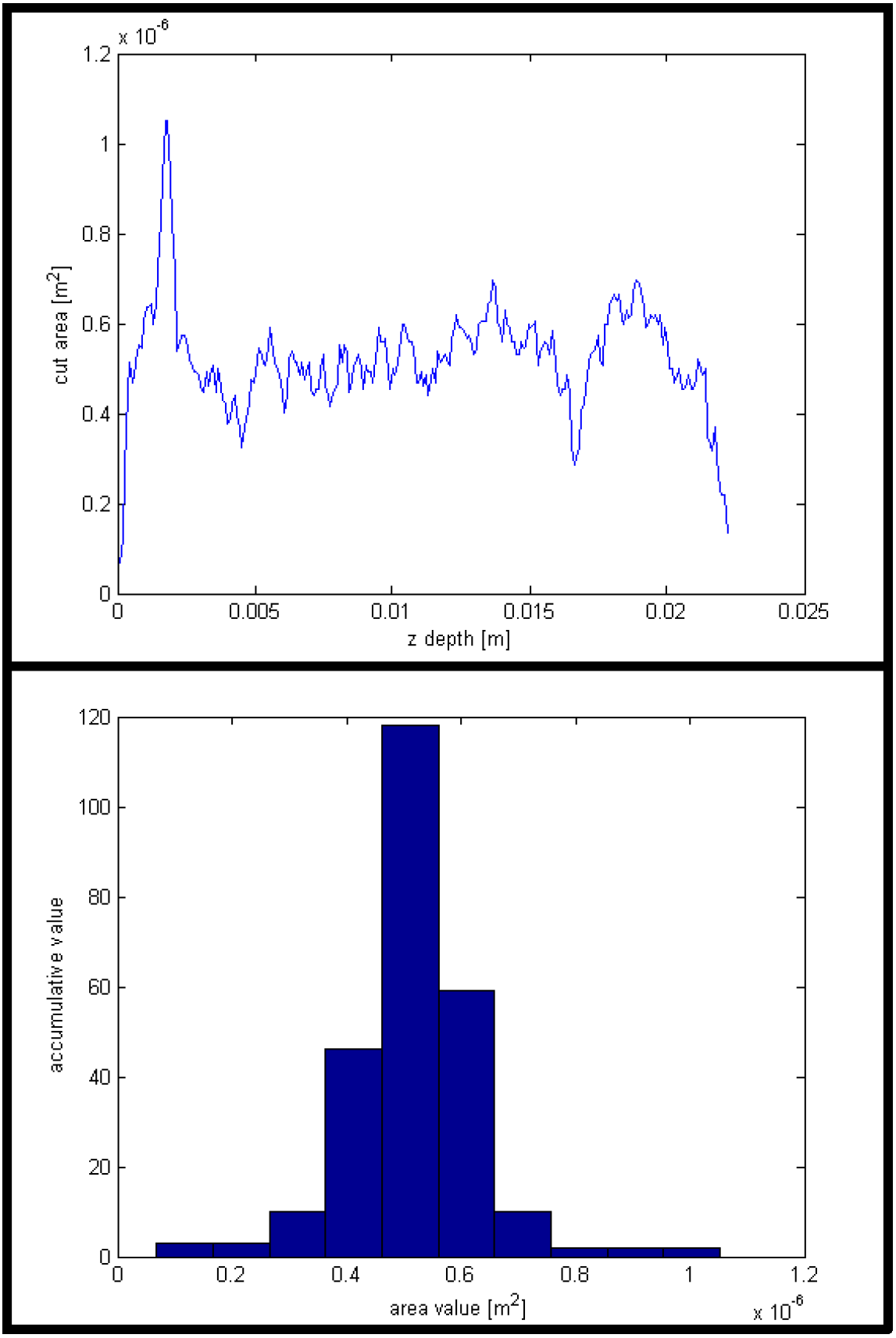


Figure 7.14: **Pore-3 area distribution** - Pore-3 area distribution with depth is show in the top image, the area per cut is lowest at the outlet.

- **Pore diameter distribution:** Pore-3 mean diameter is 3.67×10^{-4} m with a std of 4.97×10^{-5} m

The top image in Figure 7.15 shows the relation between diameter and depth, and the bottom image represents the frequency of diameter.

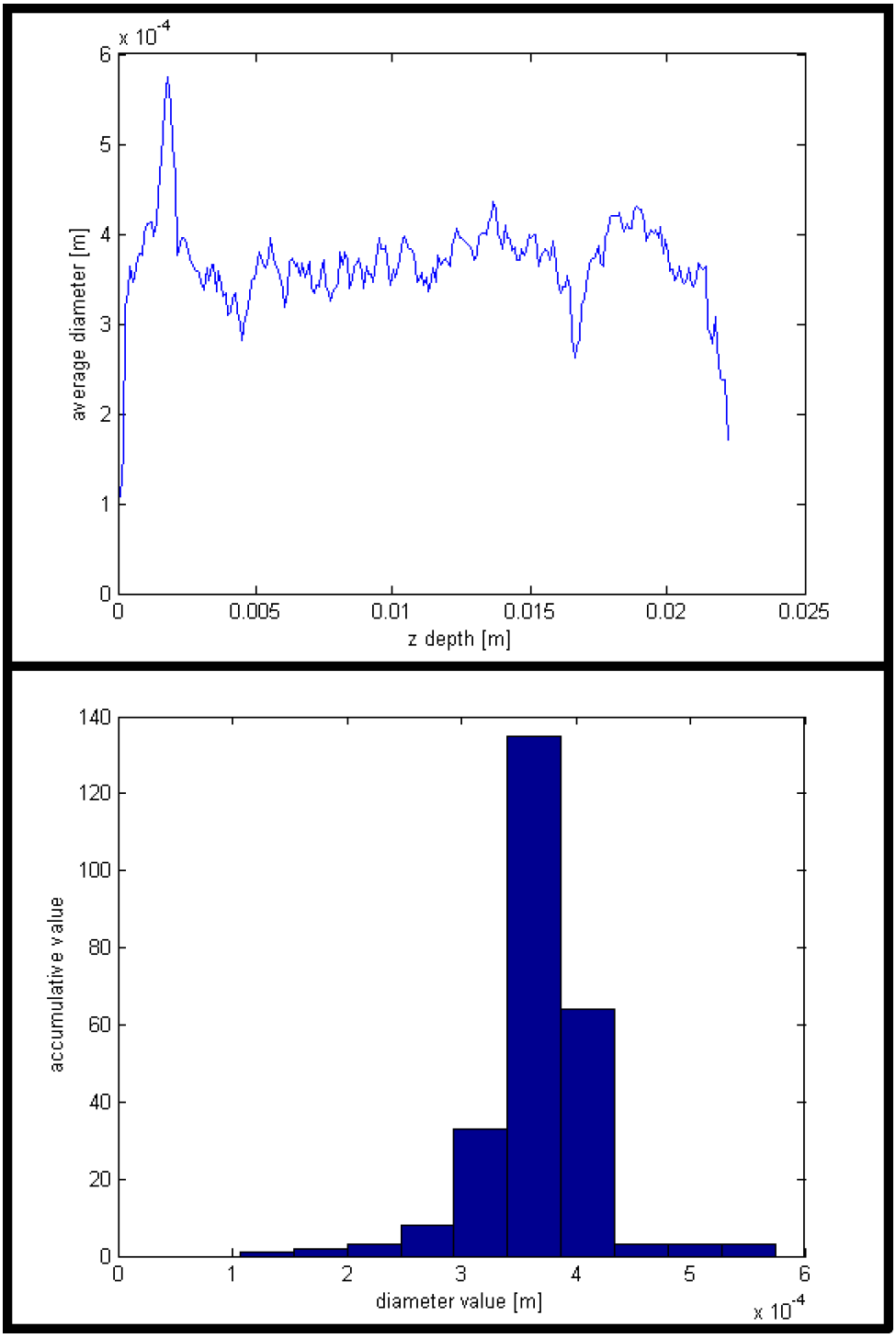


Figure 7.15: **Pore-3 diameter distribution** - Pore-3 diameter distribution with depth, as show in the top image the diameter per cut is lowest at the outlet.

Pore-6

In this section the Pore-6 geometric and morphological properties are presented.

- **Pore skeleton:** The effective length of the skeleton is: $Po_{\varsigma} = 0.021\ m$. Shown in blue in Figure 7.16.
- **Pore length:** The pore length is: $Po_{\ell} = 0.017\ m$. Shown in red in Figure 7.16.

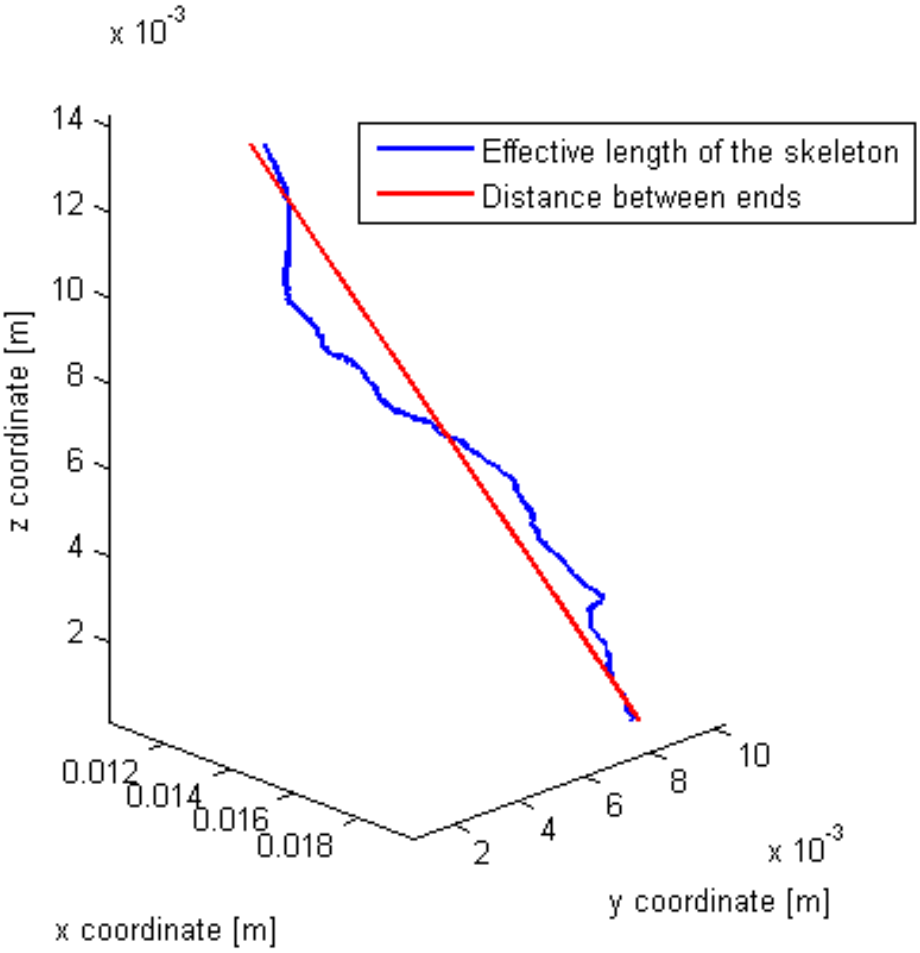


Figure 7.16: **Pore-6 Skeleton and length** - Pore-6 Skeleton and length. the ratio between these values gives us the tortuosity of the pore.

- **Tortuosity:** The tortuosity is equal to:

$$Po_{\tau} = \frac{Po_{\varsigma}}{Po_{\ell}} = \frac{0.021m}{0.017m} = 1.26 \tag{7.13}$$

- **Pore inlet area:** The inlet area is equal to

$$\begin{aligned} Po_{\alpha} &= Pc_{\alpha} \cdot pixel\_area \\ &= 40 \cdot \left(8.7 \times 10^{-5}m * 8.7 \times 10^{-5}m\right) \\ &= 3.0276 \times 10^{-7}m^2 \end{aligned} \tag{7.14}$$

The top binary cut with the inlet area is shown in Figure 7.17.

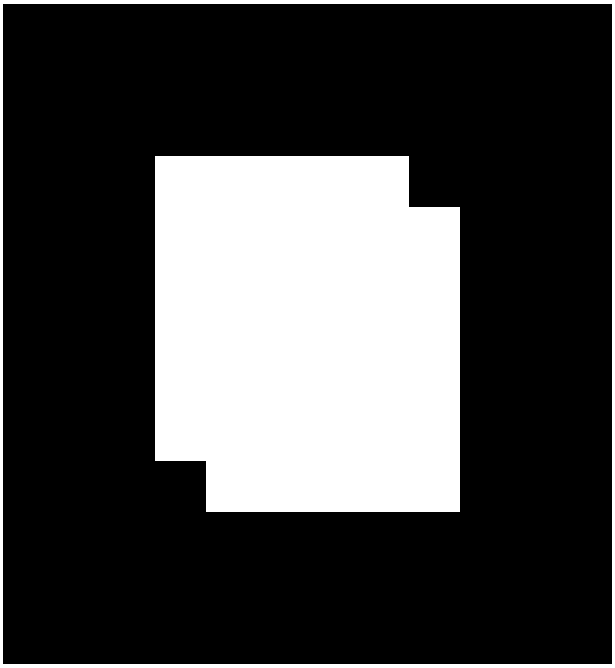


Figure 7.17: **Pore-6 inlet area** - Pore-6 binary cut showing the inlet area.

- **Pore outlet area:** The outlet area is equal to

$$\begin{aligned} Po_\beta &= Pc_\beta \cdot pixel\_area \\ &= 48 \cdot \left(8.7 \times 10^{-5} m * 8.7 \times 10^{-5} m\right) \\ &= 3.63 \times 10^{-7} m^2 \end{aligned} \tag{7.15}$$

The bottom binary cut with the inlet area is shown in Figure 7.18.

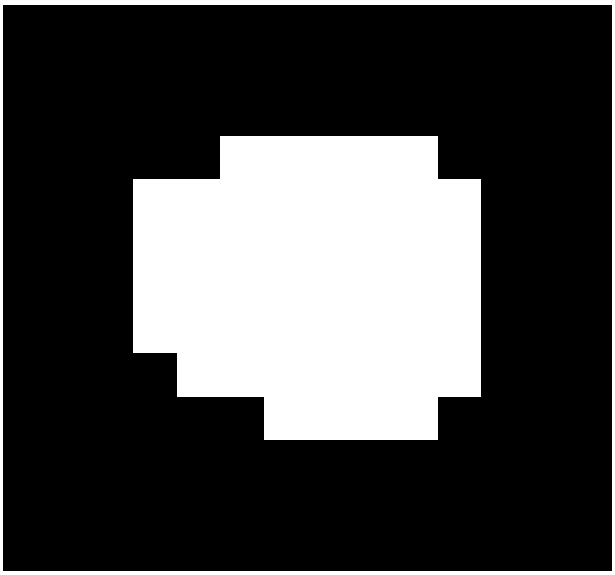


Figure 7.18: **Pore-6 outlet area** - Pore-6 binary cut showing the outlet area.

- **Pore area distribution:** Pore-6 mean area is $5.79 \times 10^{-7}\ m$ with a std of $2.022 \times 10^{-7}\ m$

  The top image in Figure 7.19 shows the relation between area and depth, and the bottom image represents the frequency of areas.

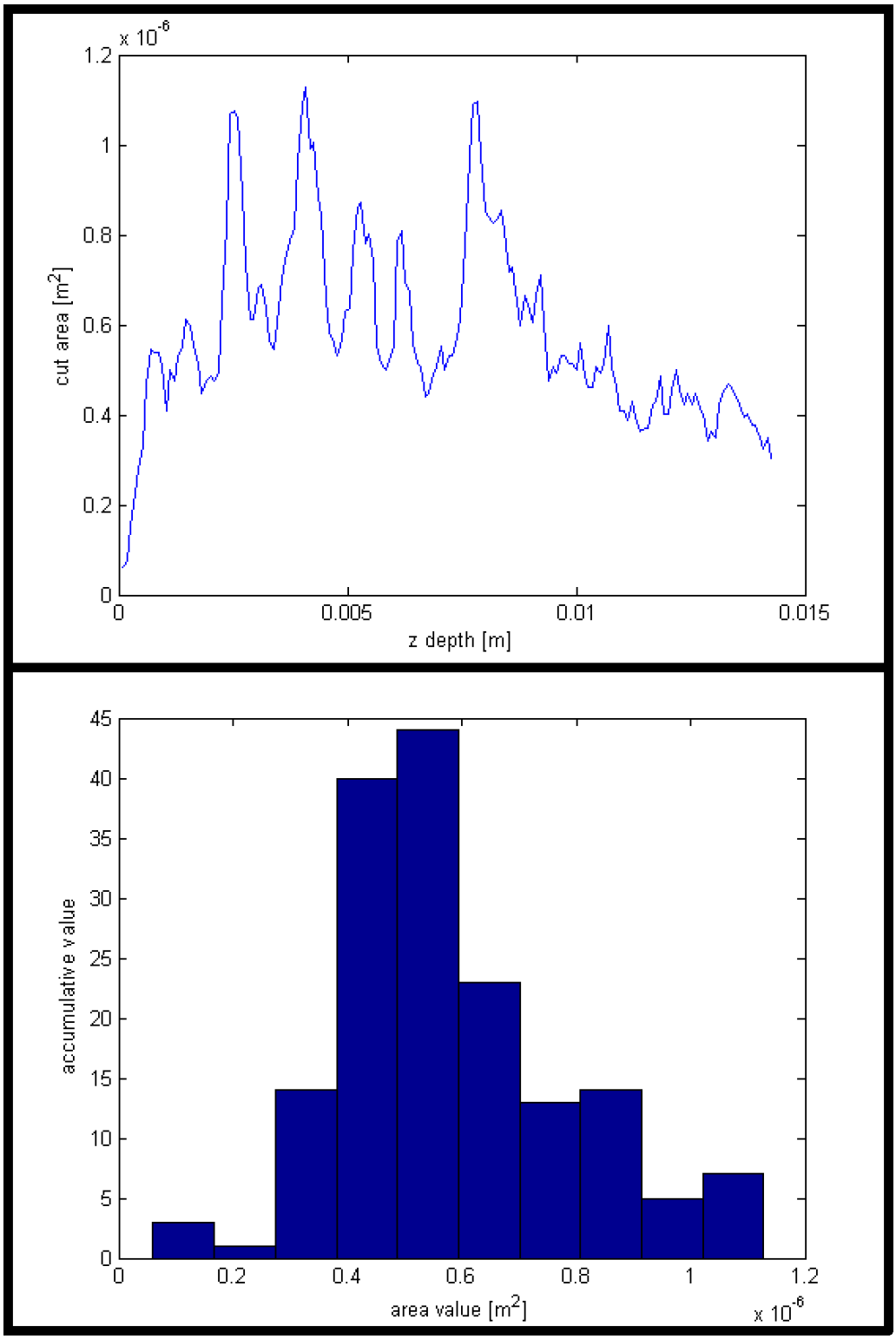


Figure 7.19: **Pore-6 area distribution** - Pore-6 area distribution with depth.

- **Pore diameter distribution:** Pore-6 mean diameter is $3.92 \times 10^{-4}\ m$ with a std of $8.55 \times 10^{-5}\ m$

The top image in Figure 7.20 shows the relation between diameter and depth, and the bottom image figure represents the frequency of diameter.

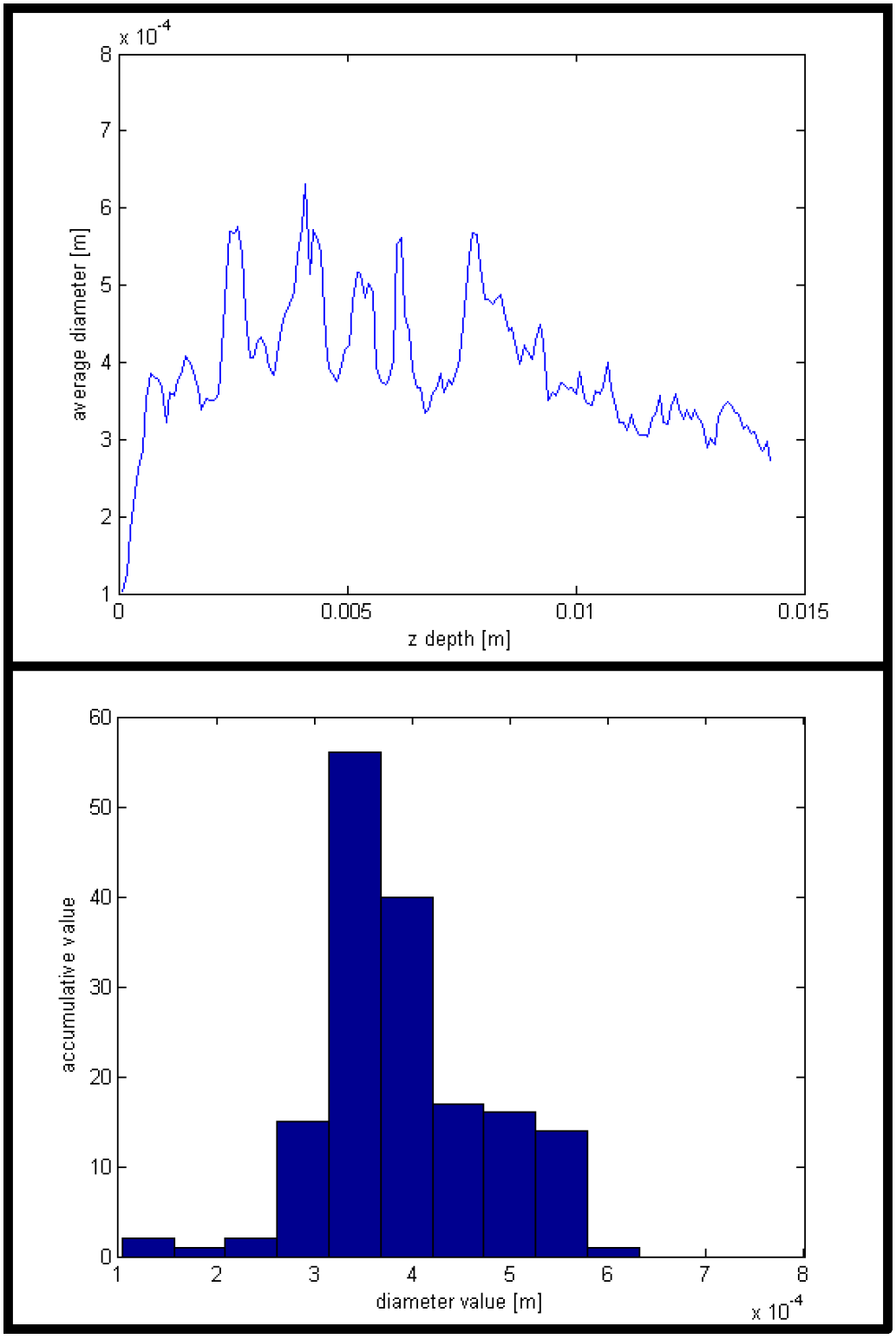


Figure 7.20: **Pore-6 diameter distribution** - Pore-6 diameter distribution with depth.

Pore-9

In this section the Pore-9 geometric and morphological properties are presented.

- **Pore skeleton:** The effective length of the skeleton is: $Po_{\varsigma} = 0.015\ m$. Shown in blue in Figure 7.21.
- **Pore length:** The pore length is: $Po_{\ell} = 0.013\ m$. Shown in red in Figure 7.21.

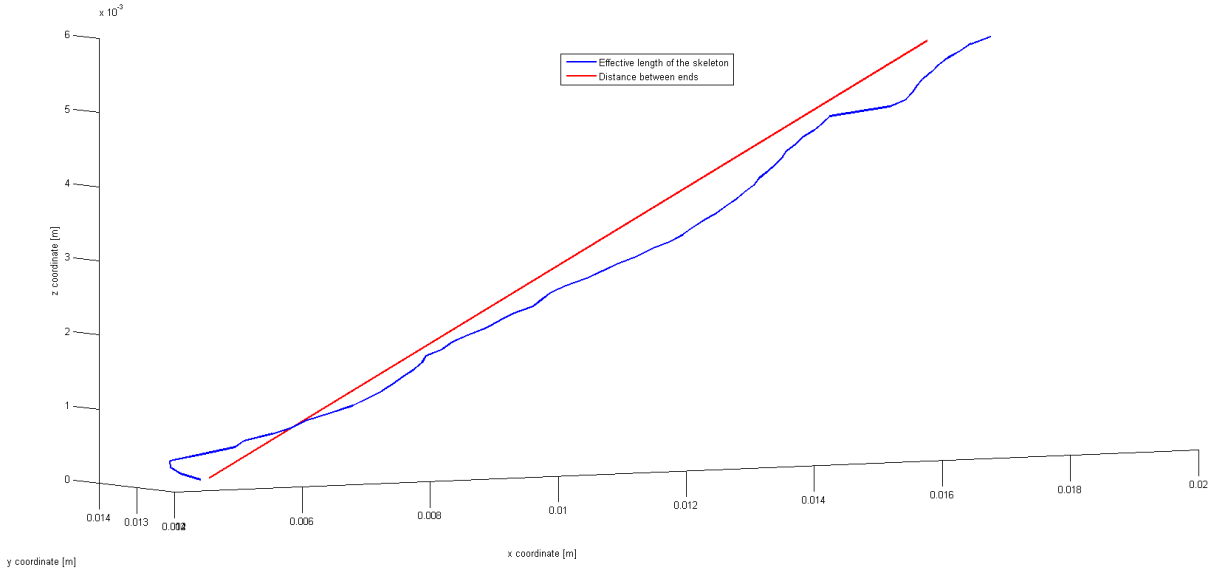


Figure 7.21: **Pore-9 Skeleton and length** - Pore-9 Skeleton and length. The ratio between these values gives us the tortuosity of the pore.

- **Tortuosity:** The tortuosity is equal to:

$$Po_{\tau} = \frac{Po_{\varsigma}}{Po_{\ell}} = \frac{0.015m}{0.013m} = 1.21 \tag{7.16}$$

- **Pore inlet area:** The inlet area is equal to:

$$\begin{aligned} Po_{\alpha} &= Pc_{\alpha} \cdot pixel\_area \\ &= 197 \cdot \left(8.7 \times 10^{-5}m * 8.7 \times 10^{-5}m\right) \\ &= 1.49 \times 10^{-6}m^2 \end{aligned} \tag{7.17}$$

  The top binary cut with the inlet area is shown in Figure 7.22.

- **Pore outlet area:** The outlet area is equal to:

$$\begin{aligned} Po_{\beta} &= Pc_{\beta} \cdot pixel\_area \\ &= 55 \cdot \left(8.7 \times 10^{-5}m * 8.7 \times 10^{-5}m\right) \\ &= 4.16 \times 10^{-7}m^2 \end{aligned} \tag{7.18}$$

  The bottom binary cut with the inlet area is shown in Figure 7.23.

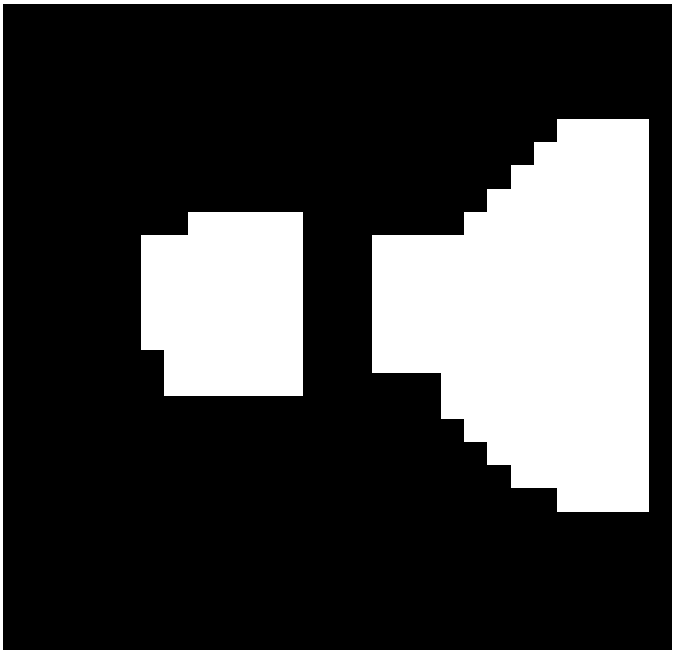


Figure 7.22: **Pore-9 inlet area** - Pore-9 binary cut showing the inlet area.

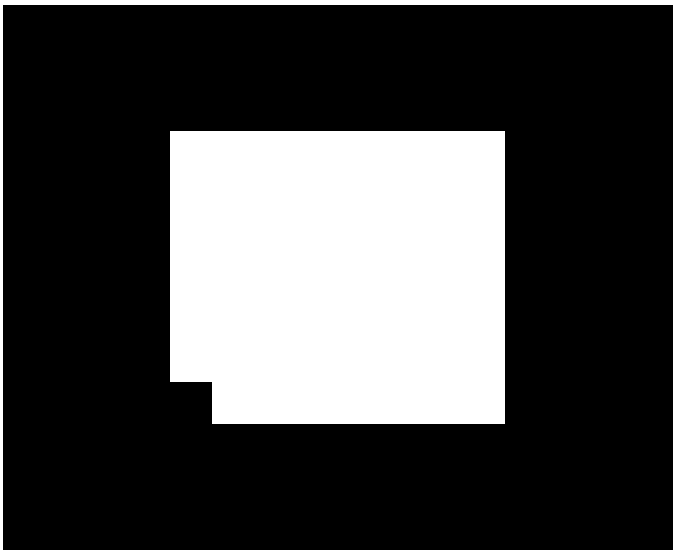


Figure 7.23: **Pore-9 outlet area** - Pore-9 binary cut showing the outlet area.

- **Pore area distribution:** Pore-9 mean area is 2.62×10^{-6} m with a std of 1.28×10^{-6} m

  The top image in Figure 7.24 shows the relation between area and depth, and the bottom image represents the frequency of areas.

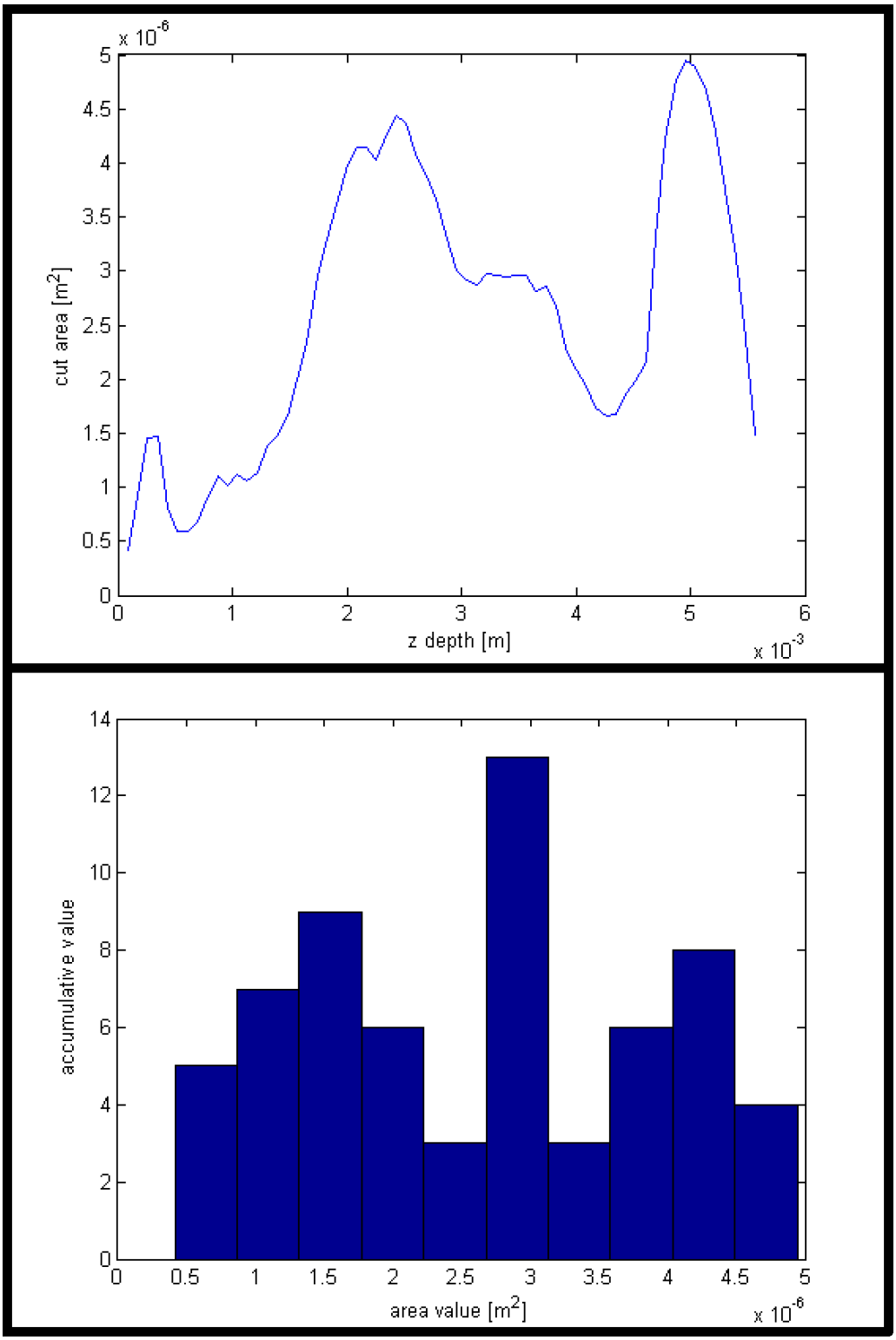


Figure 7.24: **Pore-9 area distribution** - Pore-9 area distribution with depth.

- **Pore diameter distribution:** Pore-9 mean diameter is 9.13×10^{-4} m with a std of 2.64×10^{-4} m

  The top image in Figure 7.25 shows the relation between diameter and depth, and the bottom image represents the frequency of diameter.

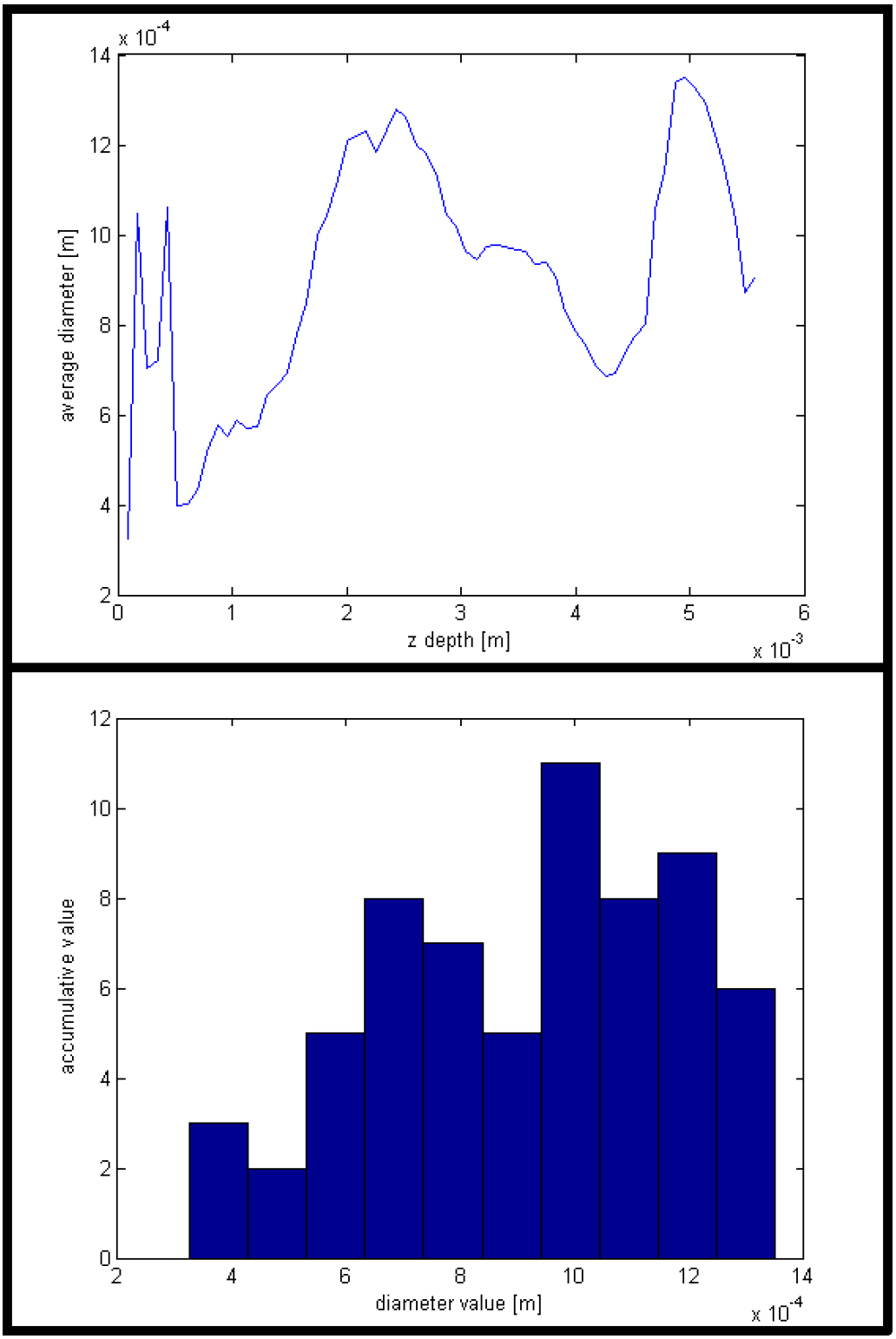


Figure 7.25: **Pore-9 diameter distribution** - Pore-9 diameter distribution with depth.

Table 7.3 shows the morphometric and geometric properties of each pore. Volume ratios between each pore volume and total VER volume are 1.37%, 0.98%, and 3.28% for pore 3, 6 and 9 respectively. Pore 3 is the most vertical and straight pore, having the lowest tortuosity, and the skeleton length as the closest value to the vertical length. Pore 3 also has the lowest mean hydraulic radius, meaning that the channel is the most efficient for conducting the flow. Pore 6 is the smallest pore in volume, and has the highest tortuosity as it contains an angle. Pore 9 is the biggest pore, with the highest mean diameter and total volume. It also shows a big variation in area and has the lowest vertical slope and the highest mean hydraulic radius. It is thus the pore the less efficient to conduct the flow.

Pore morphometric and geometric properties									
Pore #	Po_{ς} $\times 10^{-3}$ m	Po_{ℓ} $\times 10^{-3}$ m	R_h $\times 10^{-3}$ m	Po_{τ} m	Po_{α} $\times 10^{-6}$ m^2	Po_{β} $\times 10^{-6}$ m^2	Mean area $\times 10^{-6}$ m^2	STD $\times 10^{-6}$ m^2	Vertical length $\times 10^{-3}$ m
$Pore-3$	25.6	22.1	194.2	1.16	0.14	0.068	0.52	0.12	22.09
$Pore-6$	20.9	16.6	198.4	1.26	0.30	0.36	0.58	0.20	14.00
$Pore-9$	15.2	12.6	315.7	1.21	0.88	0.55	2.60	1.27	5.48

Table 7.3: Morphometric and geometric properties of each pore, these properties together with the boundary conditions determine the behaviour of the fluid flow inside the pores.

7.4 Computational modelling

After the pores have been selected and the morphological measurements obtained, the 3D model of the soil wall is used as the boundary conditions for the computational modelling of the fluid flow.

The fluid solution is obtained either using the SPH implementation or with the FEM program COMSOL. Since COMSOL is a mesh based algorithm two different 3D database formats were constructed:

- **3D Mesh-free database:** The 3D model consists only of points $\mathbf{E_p}$ with components (x, y, z), all the 3D points belong to the wall (void boundary). The SPH implementation takes the position of these points and creates a set of SPH particles $\mathbf{P}$, Figure 7.26 shows the particle system that forms the boundary wall for the SPH solution.

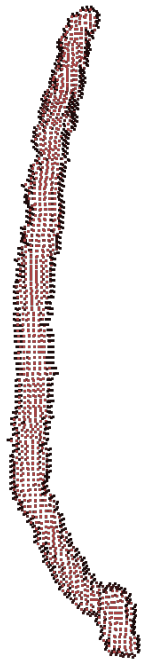


Figure 7.26: **Pore-3 particle system** - Pore-3 particle system defining the boundary conditions of the SPH solution.

- **3D Mesh database:** COMSOL requires either a surface mesh (formed by 2D objects such as triangles or square objects) or a full tetrahedral mesh. During this experiment COMSOL was used to build a tetrahedral mesh.

  The 3D model consists of the set of points $\mathbf{E_p}$ with components (x, y, z) and a set of triangles, index $\mathbf{T_i}$ containing the three point components that form the mesh components.

  The 3D object passed to COMSOL contains the triangles forming the problem walls, COMSOL uses this mesh to build a full 3D tetrahedra where the interior triangles will be used to simulate the fluid. Figure 7.27 shows the triangle mesh and the resulting tetrahedral mesh.

The computer used to solve both the COMSOL and SPH solutions has the following components:

- **CPU:** Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz 5.80GHz
- **RAM:** 16.0 GB

Figure 7.27: **Pore-3 triangle mesh - Left:** shows the Pore-3 Triangle mesh surface. **Right:** shows the resulting tetrahedral mesh produced by COMSOL. The COMSOL internal mesh creator was used in order to prevent external triangulation errors that could compromise the computational solver.

- **Graphic card:** GeForce GTX 580 @ 772 MHz 1544 MHz @ 1536 MB GDDR5 memory
- **System Type:** 64-bit

The next step is the definition of the computational problem and the simulation. The first section will describe the COMSOL procedure and then the SPH solution will be presented.

7.4.1 COMSOL solution

COMSOL is proprietary software with no source code available to the user. All the operations are performed using the solvers and options available.

The first step was to prepare the 3D model of the problem. As described in Section 7.4 the internal mesh generation is used.

The steps taken when modelling using COMSOL were:

I **Mesh generator:** The quality of the solution is directly related to the quality of the Mesh provided and built by COMSOL. COMSOL is a FEM Euler based solution system. FEM systems(as described in Section 1.7.1) require fixed points where the solution to the problem will be obtained.

The more vertex and tetrahedral objects provided the more points COMSOL will have to evaluate the solution. Table 7.4 presents the object triangle properties for the 3 pores.

COMSOL Pores Mesh Properties				
Pore number	Triangle vertices	Triangle faces	Tetrahedral mesh elements	Tetrahedral mesh quality
3	9002	18000	296998	0.7936
6	9002	18000	296998	0.7935
9	3559	14652	61384	0.7682

Table 7.4: The table presents the pore mesh properties introduced into the COMSOL solver.

The quality of an element is a double value from 0 to 1, where 0.0 represents a degenerated element and 1.0 represents a completely symmetric element. Due to memory management limitations it is not possible to obtain a value of 1 so the best approximation is used in all cases.

II **Model material:** The material for the object is set to the predefined COMSOL water. The following parameters are defined:

Density 1000 kg/m^3
Viscosity 0.003 $Pa \cdot s$
Inlet pressure: 8.7×10^{-6} Pa
Outlet pressure: 0 Pa

III **Physic Model:** COMSOL allows for multiple types of physical models. To solve the water flow through the tetrahedral mesh the "Laminar Flow Single-Phase Flow (spf)" module was selected.

COMSOL Navier-Stokes equation (3.3) and the spf module was configured with the following options:

Compressibility Incompressible flow
Turbulence None
Initial values: Velocity and initial pressure are set to 0

IV **Boundary conditions:** In all pores there are three boundary conditions, wall, inlet and outlet:

Wall The wall is formed by all the boundary elements, excluding the inlet and outlet definition.

Inlet Top part of the pore. The inlet is used to define the starting position of the flow and the inlet pressure.

Outlet: Bottom part of the pore. The outlet is used to define the exit plane of the flow and the outlet pressure.

V **Mesh quality:** Depending on the imputed triangle mesh, different qualities of tetrahedral mesh element size can be selected in COMSOL. Table 7.4 shows the resulting mesh quality per pore. When selecting the element size for mesh building "finer quality" was picked from the element size options.

VI **Study:** COMSOL provides a module to define the solver to be used for the physical simulation. For the pore simulation the following setup was choosen:

Study step A stationary solver was selected. The stationary solver is used to solve a steady-state problem where all time derivatives vanish.

Solver An iterative solver using the (generalised minimum residual) GMRES method was choosen [117] with a maximum number of interactions equal to 50.

Preconditioner: The Geometric Multigrid method for elliptic and parabolic models was selected. This method used a coarser mesh when possible.

After configuration, COMSOL obtained the computational solution.

7.4.2 SPH solution

The simulation of the SPH model requires the use of a set of particle systems, one system to define the fluid and the other for the soil boundary particles.

I **Particle system:** In SPH the quality and precision of the solution is directly related to the number of particles per volume in the model. Table 7.5 gives the number of particles per pore for both components of the problem.

SPH particle systems		
Pore number	Fluid particles	Solid particles
3	10164	12814
6	10164	12814
9	18530	12535

Table 7.5: Number of particles per object in the model.

II **Particles material:** The particles are separated into two systems (fluid and soil). For the fluid particles the SPH initial values are:

Density	1000 kg/m^3
Viscosity	0.003 $Pa \cdot s$
Velocity	0.0 m/s
Acceleration	0.0 m/s
Smoothing length	0.000181 m
Inlet pressure:	8.7×10^{-6} Pa
Outlet pressure:	0 Pa

The soil particles are initialised with the following values:

Density	1650 kg/m^3
Velocity	0.0 m/s
Acceleration	0.0 m/s
Smoothing length	0.000181 m

The soil particles will always maintain a velocity of 0.0 and an acceleration of 0.0. During the collisions the fluid particles bounce off the soil ones with some loss in kinetic energy.

III **Physic Model:** The SPH computational solution used to solve the problem is based on the Navier-Stokes approach with viscosity and smoothing length evolution (e.g. see Section 3.4). SPH configuration is similar to the COMSOL one:

Compressibility	Incompressible flow
Turbulence	None
Initial values:	Velocity and initial pressure of the fluid are set to 0

IV **SPH Boundary conditions:** The boundary condition is provided by the soil particle system, the inlet and outlet pressure are solved in the external force component of the simulation. The boundary conditions for the SPH solutions are:

Wall	The wall is provided by the soil particles.
Inlet	Top part of the pore. The inlet is used to define the starting position of the flow and the inlet pressure. SPH particles are reinitialised at this point after leaving through the outlet.
Outlet:	Bottom part of the pore. The outlet is used to define the exit plane of the flow and the outlet pressure. When a particle reaches the outlet it is reinitialised to be used in the next time step from the top at the inlet. The particles are reinitialised by assigned to them an initial velocity and a random position on the inlet surface

Figure 7.28 shows Pore-3 with the boundary conditions defined.

V **SPH Solution definition:** The SPH implementation has a module to define how the soil will be solved and the time step resolution:

Time step	A delta time $dt = 0.00015$ was selected
Simulation conditions	Navier-Stokes approach with viscosity and smoothing length evolution.

7.5 COMSOL pores computational modelling results

After the setup is fed into COMSOL, the system starts computing the solution. The solution to the problem depends on the mesh quality, solver type, and number of interactions.

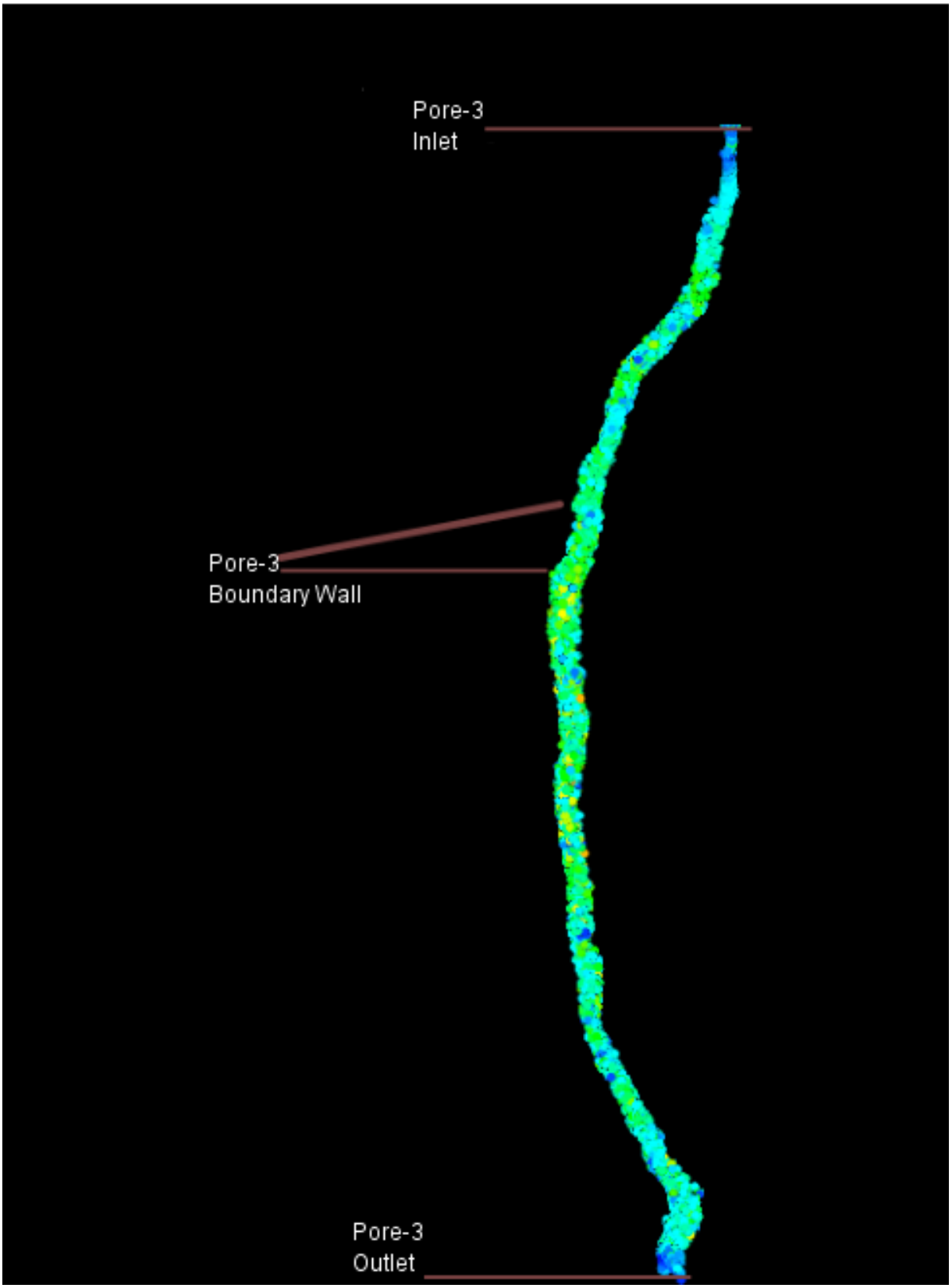


Figure 7.28: **Pore-3 SPH boundary conditions** - The figure shows the components that the SPH solution used to simulated the flux flow.

The mesh quality is a parameter that COMSOL used to determined the resolution of the tetrahedral elements forming the problem volume, as the mesh quality increased the resolution of the elements and the number of elements also increase, there are some limitations with respect to how many elements COMSOL can solved and the highest mesh quality (extra fine and extremely fine) options made the system crash and COMSOL fails to obtain the solution.

In the following sections the results for the three pores will be presented.

7.5.1 COMSOL Pore-3 solution

Since COMSOL is a mesh based algorithm it is important to see how the fluid flow behaves where the mesh quality is the lowest. The solution presented by the GMRES method is important for visualizing the relation between the solution and the morphologic structure of the soil boundary. Figure 7.29 shows the isosurface of the flow and mesh boundary structure.

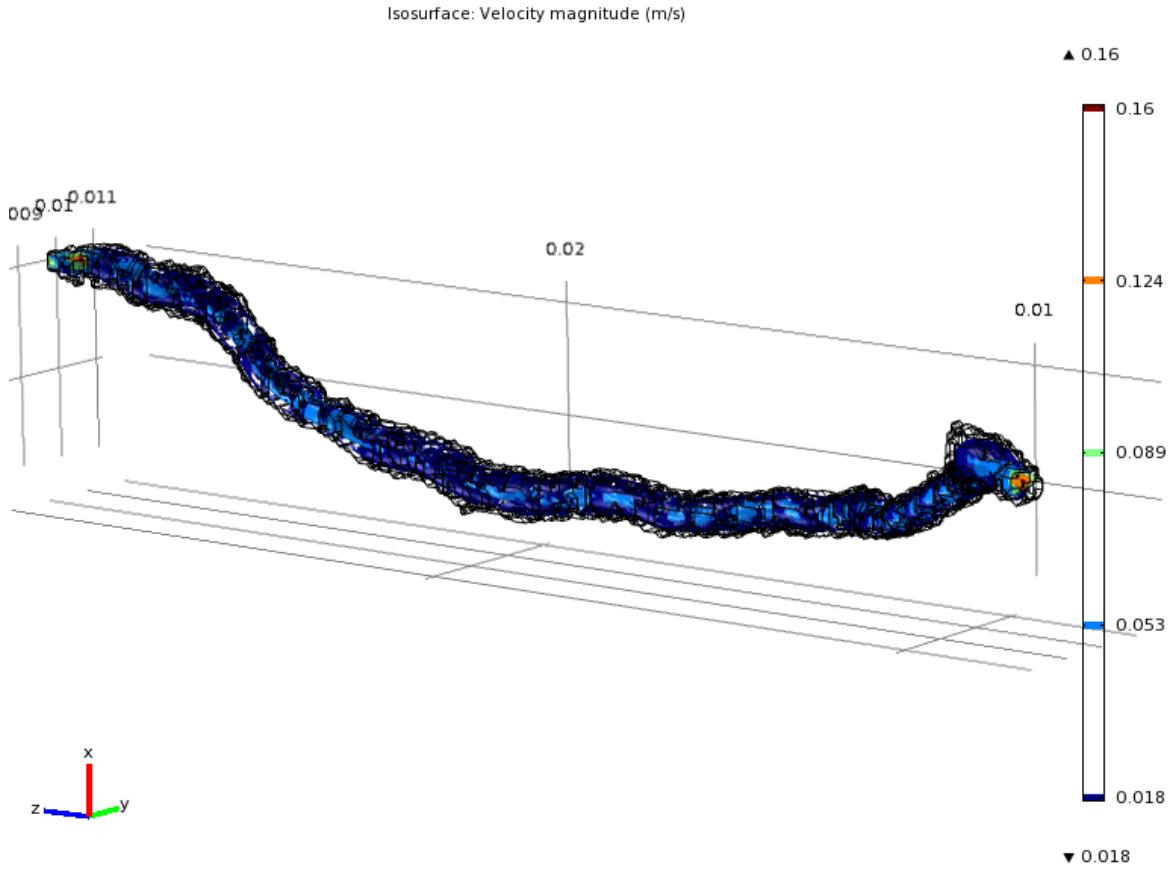


Figure 7.29: **Pore-3 isosurface** - Pore-3 isosurface describing the behaviour of the fluid flow inside the boundary. The elliptical approximation of the solution can be seen in the image.

In Figure 7.29 it can be seen that the solution of the fluid velocity has an elliptic behaviour with a centre inside the area cut. The ellipsoid solution approaches the roughness of the Pore-3 for flow values greater than 0. The no-slip condition on the boundary caused that the velocity of particles at the boundary tends to 0.

Figure 7.30 shows Pore-3 COMSOL results for different area planes in the xy-direction at different levels along the z coordinate (depth).

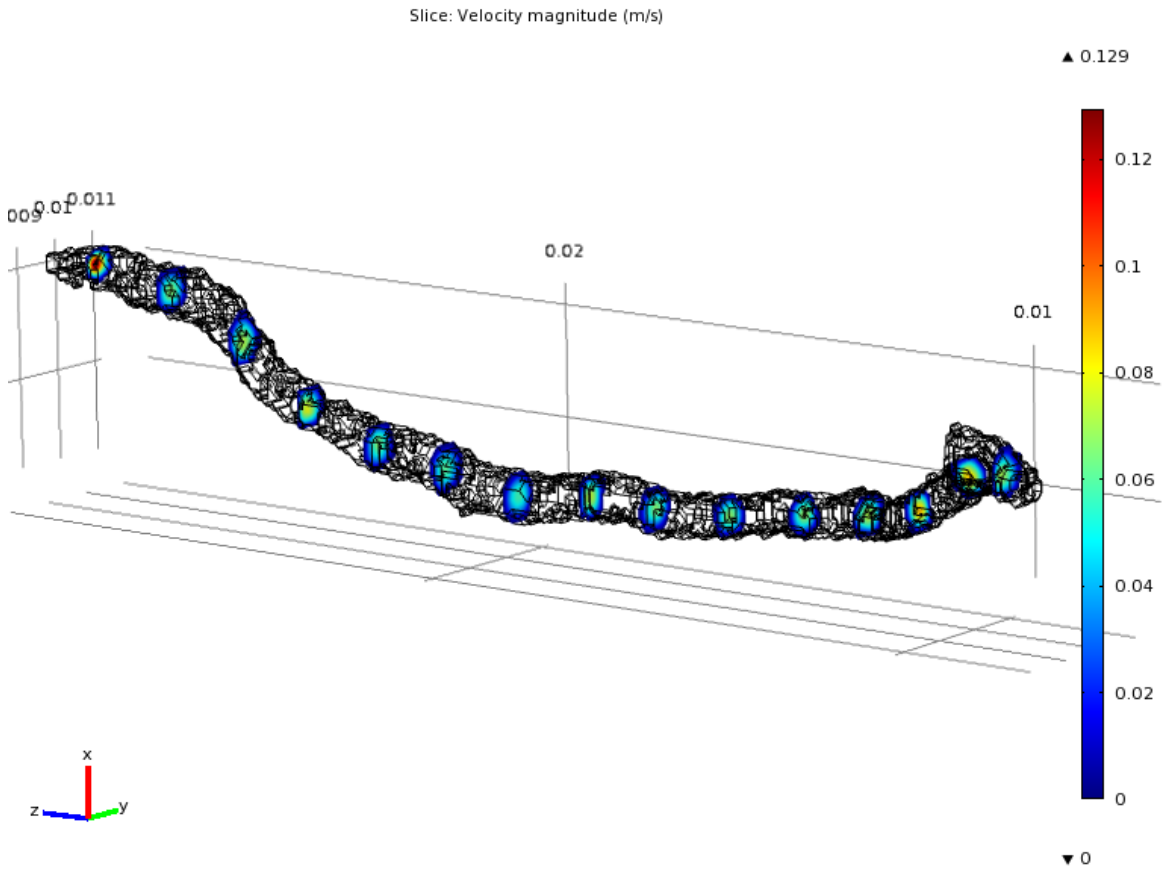


Figure 7.30: **Pore-3 xy-area result** - Pore-3 results of the simulation. COMSOL provides a set of area solutions in the xy-direction of the flow behaviour at different depths of the pore structures.

COMSOL allows to study the volumetric flow at the outlet and at the inlet of the pores, the values are used to described the behaviour of the fluid flow. Using the differences between the inlet and the outlet volumetric flow is possible to check if the steady-state flow is preserved.

COMSOL Pore-3 inlet

Figure 7.31 shows the inlet area cut with the centre profile line.

Figure 7.31 shows a small interpolation mistake produced in the inlet area cut, the boundary condition is not preserved and the isosurface is not centred with respect to the boundary. This will produced an incorrect flow amount and a higher volumetric flow velocity for this cut.

COMSOL Pore-3 outlet

In Figure 7.32 the flow velocity for the outlet is described in the top and in the bottom the centre horizontal profile line is shown. As in the other problems there are parts of the solution where the boundary condition is roughly approximated.

The Darcy flow and volumetric flow at the outlet is:

$$Q = 4.2 \times 10^{-14} m^3/s \tag{7.19}$$

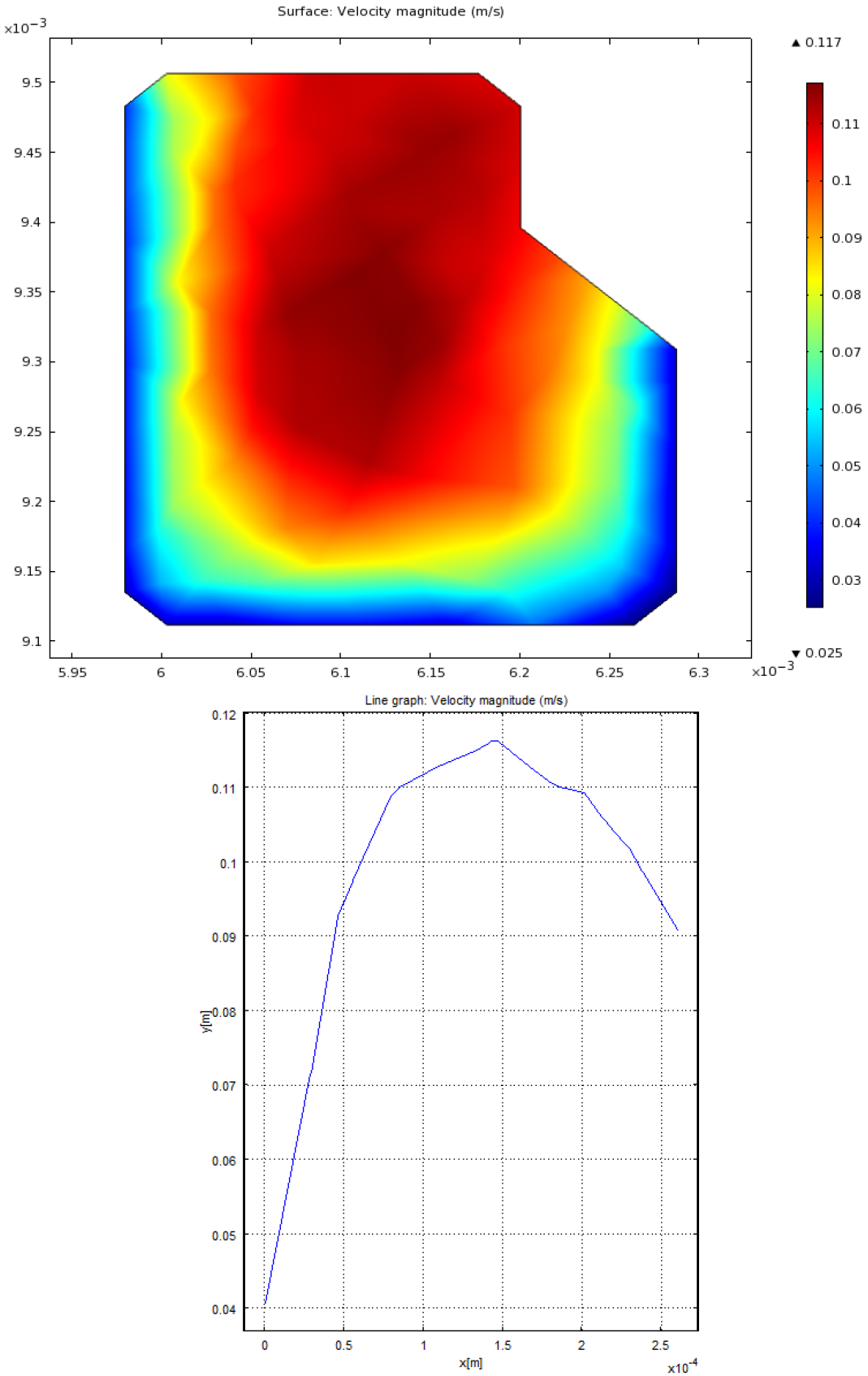


Figure 7.31: **Pore-3 inlet flow** - Pore-3 inlet velocity magnitude of the flow. The cut provides the solution to how much fluid is entering the pore. As can been seen, there is an error in the solution approximation for the inlet cut, this will produce a small component of the solution where the boundary conditions are not preserved.

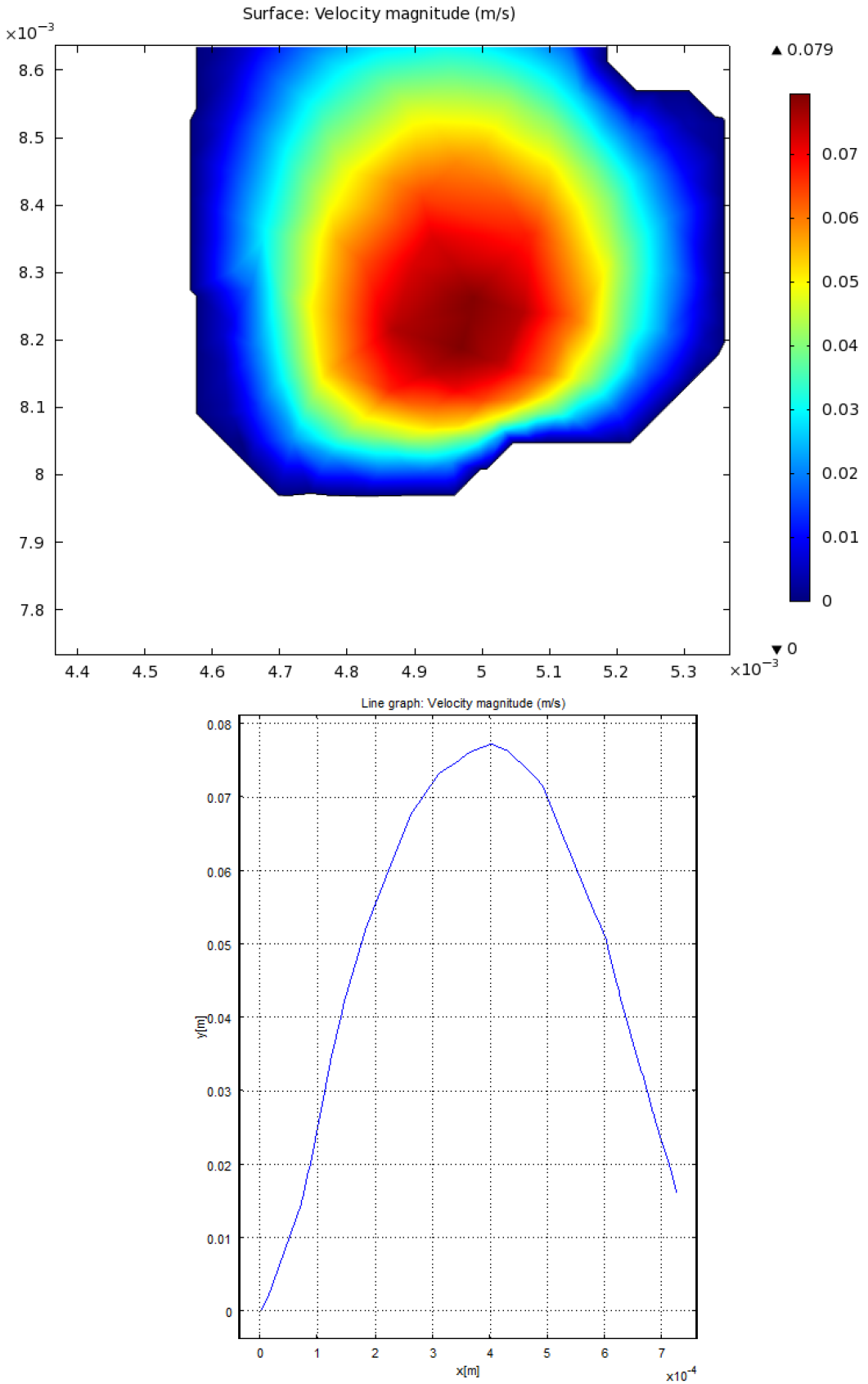


Figure 7.32: **Pore-3 outlet flow** - Pore-3 outlet flow, the cut provides the solution for how much fluid is leaving the pore. At the exit plane a better solution of the velocity at the boundary conditions can be seen.

$$v = 8.6 \times 10^{-9} m/s \tag{7.20}$$

COMSOL Pore-3 results

From the simulation results and the geometric and morphological analysis of the pores some physical properties to describe the flow in the pore are obtained.

The first description is the pore permeability (see Equation 6.2). For Pore-3 the permeability is equal to:

$$\kappa = 1.9 \times 10^{-8} m^2 \tag{7.21}$$

7.5.2 COMSOL Pore-6 solution

Figure 7.33 shows the isosurface of the flow and the mesh boundary structure for Pore-6.

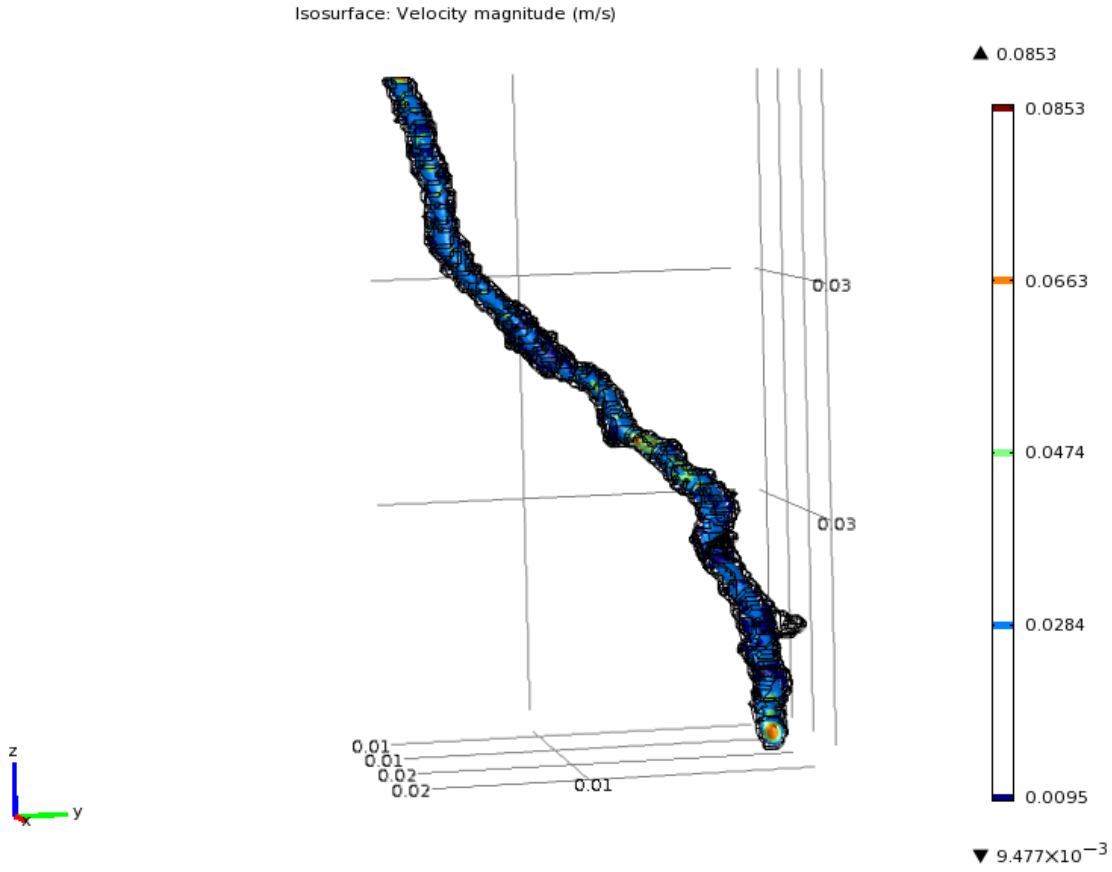


Figure 7.33: **Pore-6 isosurface** - Pore-6 isosurface describing the behaviour of the fluid velocity. Problems when approximating the boundary walls can be seen in the isosurface components.

Figure 7.34 shows the Pore-6 COMSOL results for different area planes in the xy-direction at different levels along the z coordinate (depth). Since the outlet cut is in the plane yz-direction the plane will be redefined for the calculations.

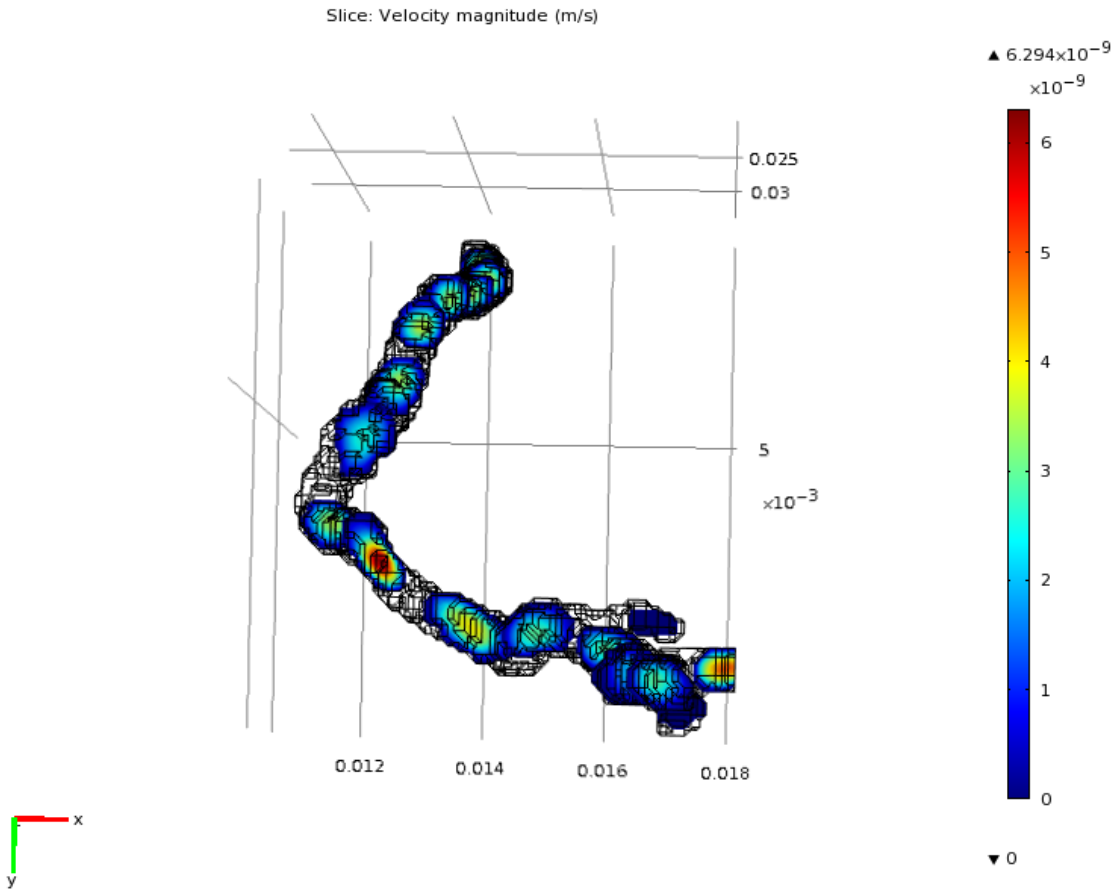


Figure 7.34: **Pore-6 xy-area result** - Pore-6 results of the simulation.

COMSOL Pore-6 inlet

Figure 7.35 shows the inlet area cut with the centre profile line. The inlet flow in COMSOL is not equal to the outlet and this will caused a difference in the result compared to SPH.

COMSOL Pore-6 outlet

Figure 7.36 shows the outlet area cut with the centre horizontal profile line.

The Darcy flow and volumetric flow at the outlet is:

$$Q = 5.6 \times 10^{-15} m^3/s \tag{7.22}$$

$$v = 1.5 \times 10^{-10} m/s \tag{7.23}$$

COMSOL Pore-6 results

For Pore-6 the permeability is equal to:

$$\kappa = 2.2 \times 10^{-10} m^2 \tag{7.24}$$

The permeability of pore-6 is lower than for pore-3; in the morphological and geometrical section it was already noted that the tortuosity and general area will reduce the fluid flow.

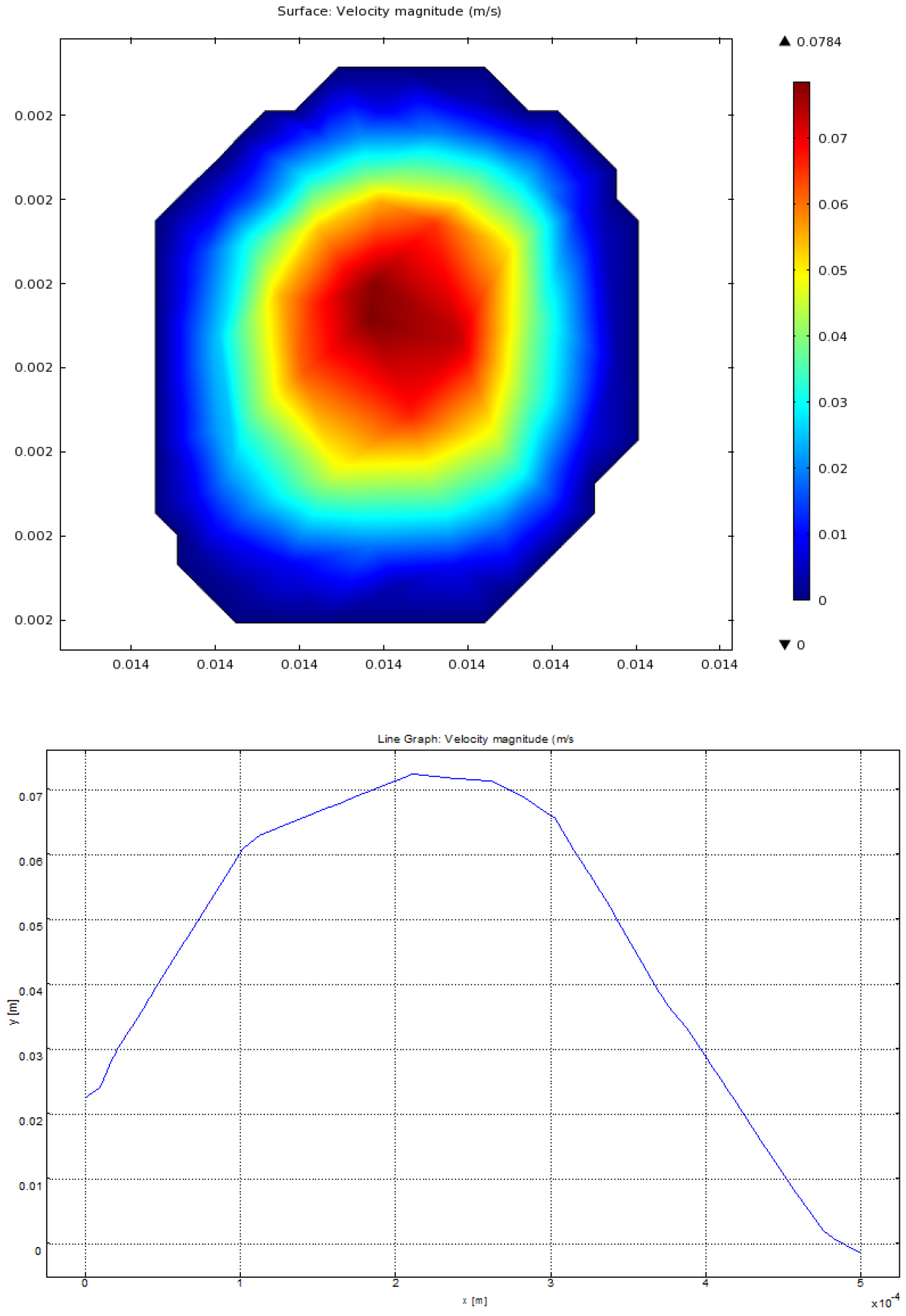


Figure 7.35: **Pore-6 inlet flow** - Pore-6 inlet flow, the cut gives the solution for how much fluid is entering the pore. Due to the morphology of the top of the Pore-6 object COMSOL provided a better aproximation than for Pore-3.

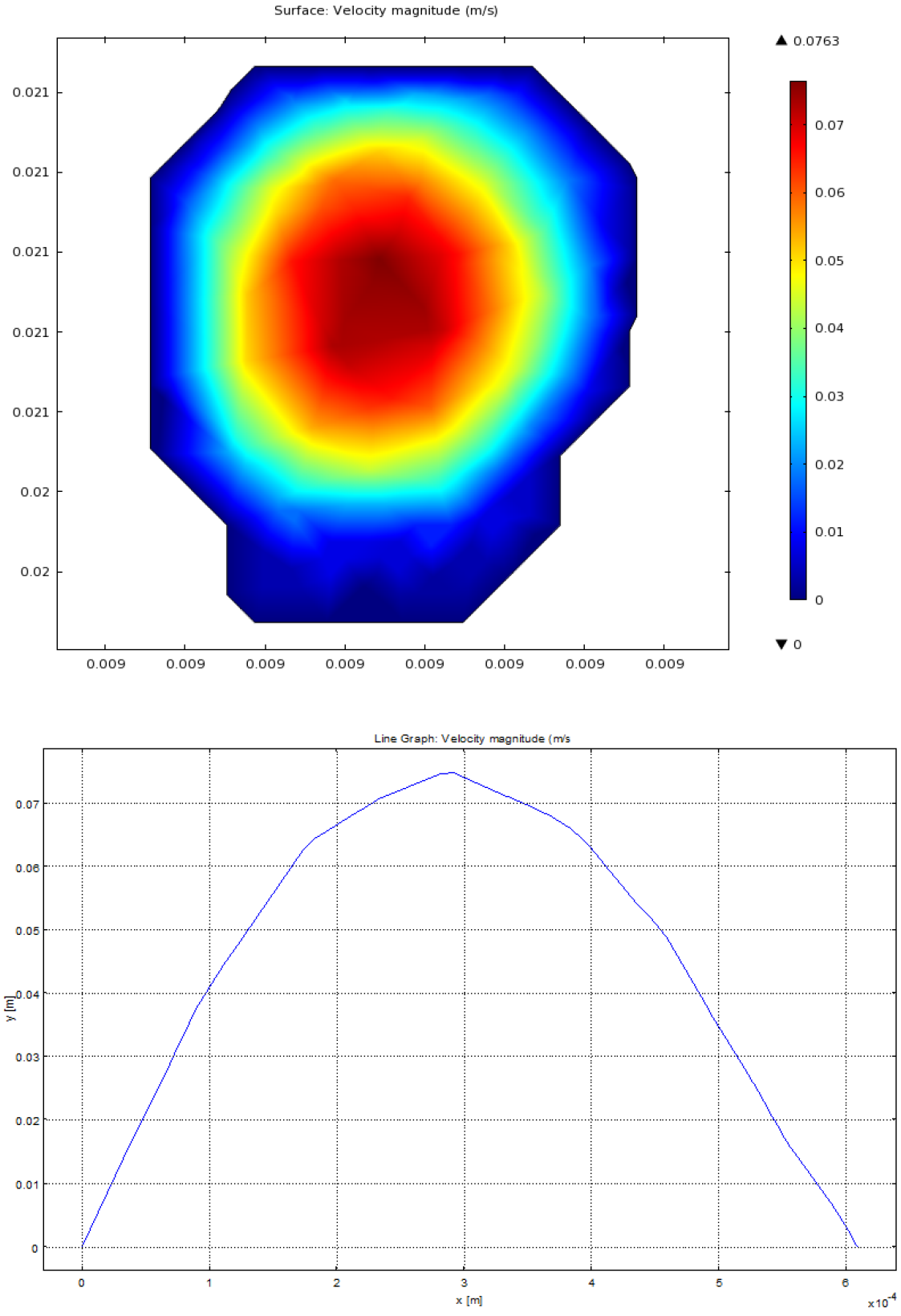


Figure 7.36: **Pore-6 outlet flow** - Pore-6 outlet flow, the exit plane was chosen to be on the yz-direction due to the pore morphology.

7.5.3 COMSOL Pore-9 solution

Figure 7.37 shows the isosurface of the flow and the mesh boundary structure for Pore-9.

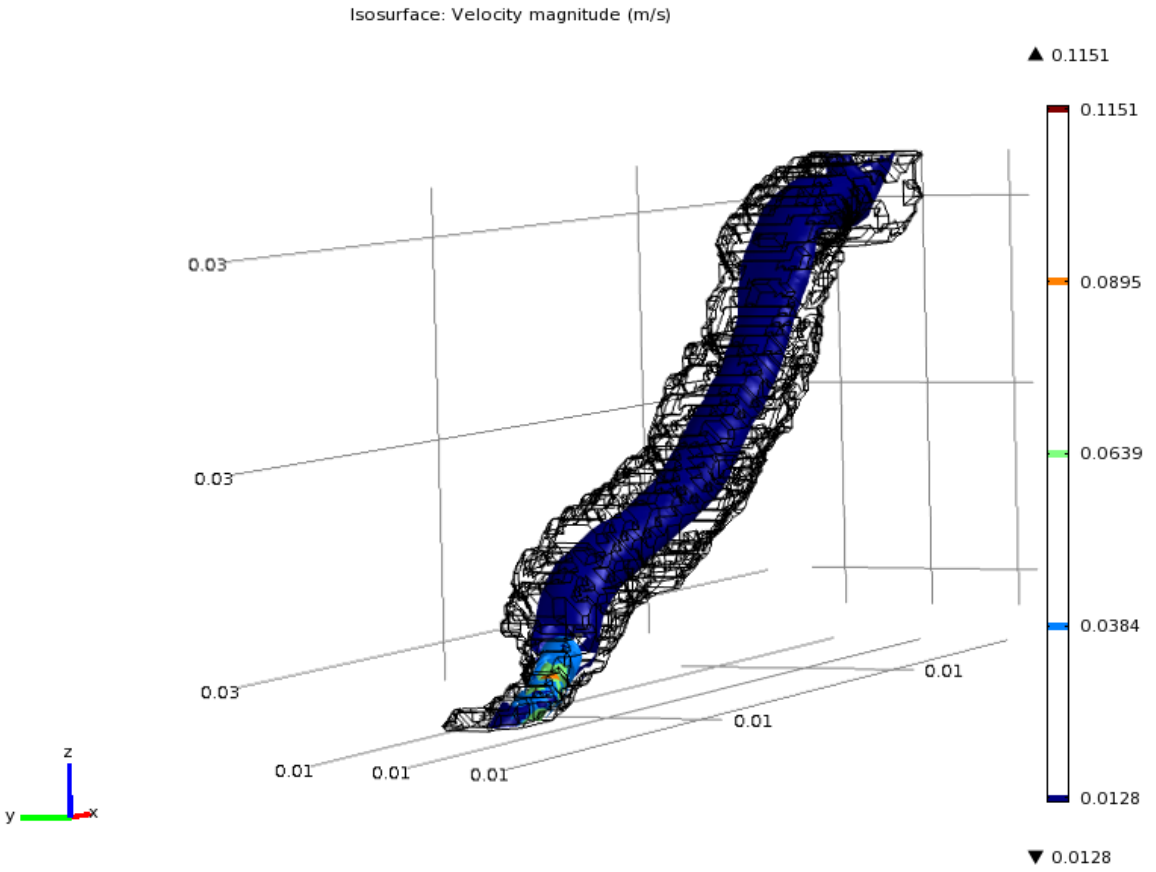


Figure 7.37: **Pore-9 isosurface** - Pore-9 isosurface describing the behaviour of the fluid flow. The resulting COMSOL isosurface description is missing the top part of the pore solution, this could be due to the mesh-quality but the number of mesh elements is limited by the available memory.

Figure 7.38 shows the Pore-9 COMSOL result for different area planes in the xy-direction at different levels along the z coordinate (depth). Since the outlet cut is in the plane yz-direction the plane will be redefined for the calculations.

COMSOL Pore-9 inlet

Figure 7.39 shows the inlet area cut with the centre profile line. The inlet data is provided when a transversal cut is defined at the top of the pore but the problem on the isosurface is still present, and the approximation will deteriorate at the following cuts.

COMSOL Pore-9 outlet

Figure 7.40 shows the outlet area cut with the centre horizontal profile line.

The volumetric flow and the Darcy flow at the outlet is:

$$Q = 8.8 \times 10^{-14} m^3/s \tag{7.25}$$

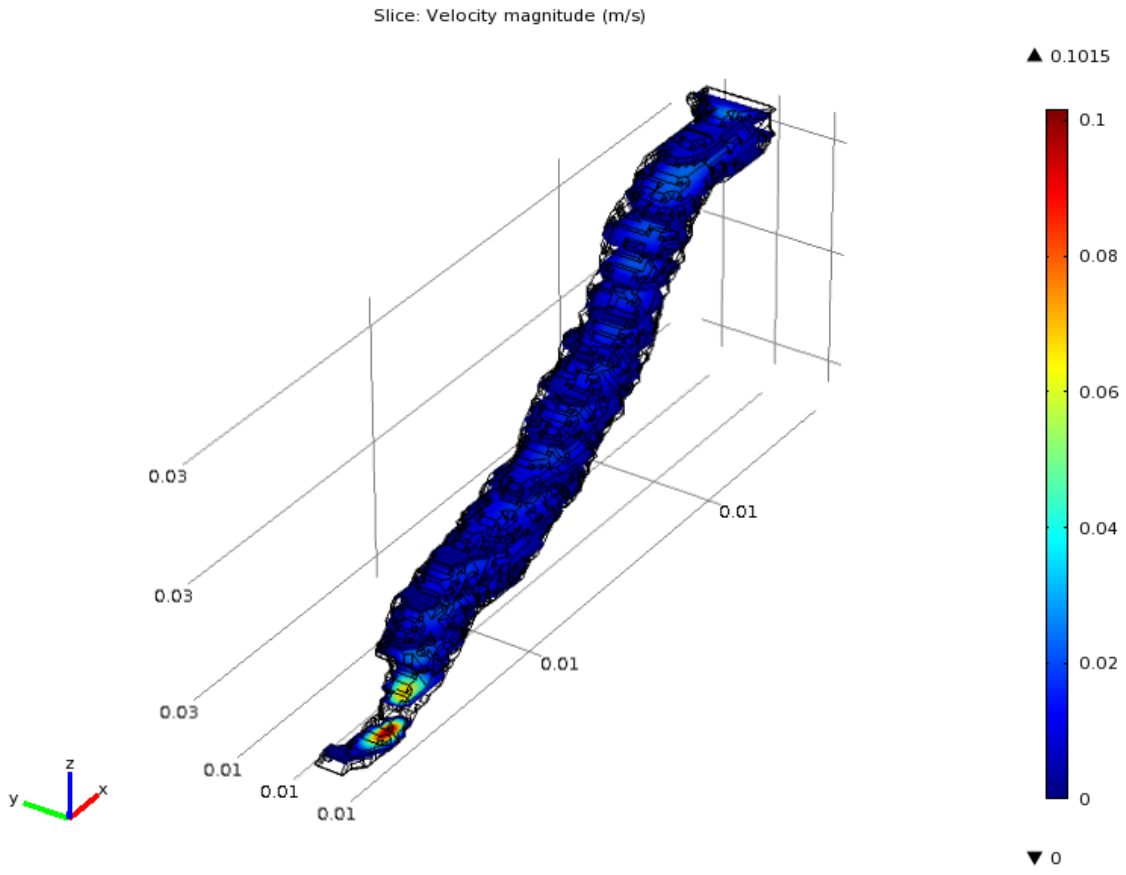


Figure 7.38: **Pore-9 xy-area result** - Pore-9 results of the simulation. A similar problem to that in the case of the isosurface solution is presented, since the isosurfaces elements are not present to obtain the correct volumetric flow most of the components on the top part of the pore have only values related to the boundary conditions and do not represent the fluid flow.

$$v = 1.5 \times 10^{-9} m/s \tag{7.26}$$

COMSOL Pore-9 results

For Pore-9 the permeability is equal to:

$$\kappa = 2.6 \times 10^{-9} m^2 \tag{7.27}$$

Pore-9 has the greatest angle with respect to the vertical this caused a slower flow velocity and less permeability.

7.6 Smoothing Particle Hydrodynamics pores computational modelling results

This section presents the results obtained using the SPH system simulation.

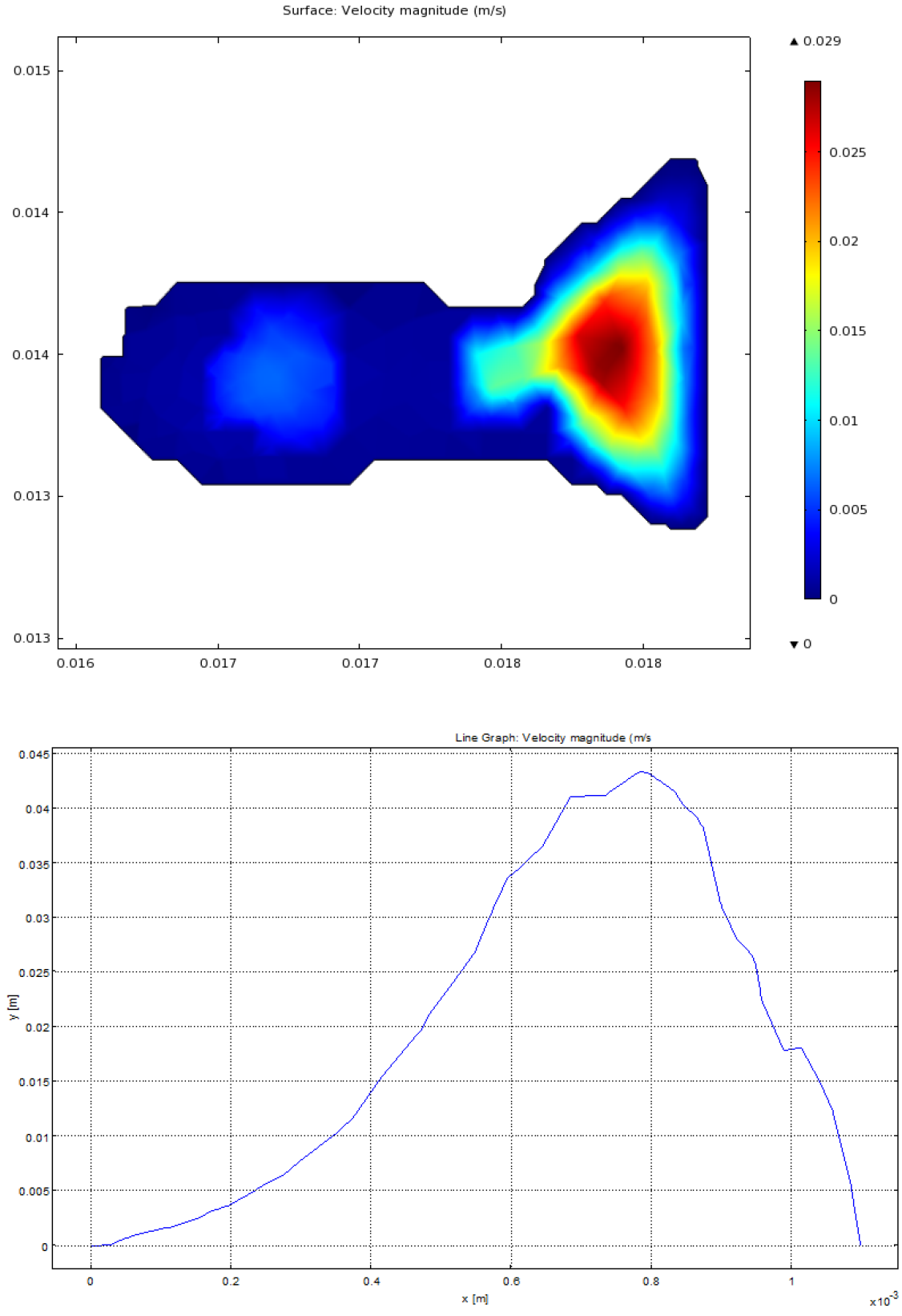


Figure 7.39: **Pore-9 inlet flow** - Pore-9 inlet flow, the cut provides the solution for how much fluid is entering the pore. The inlet of pore-9 is formed by two streams.

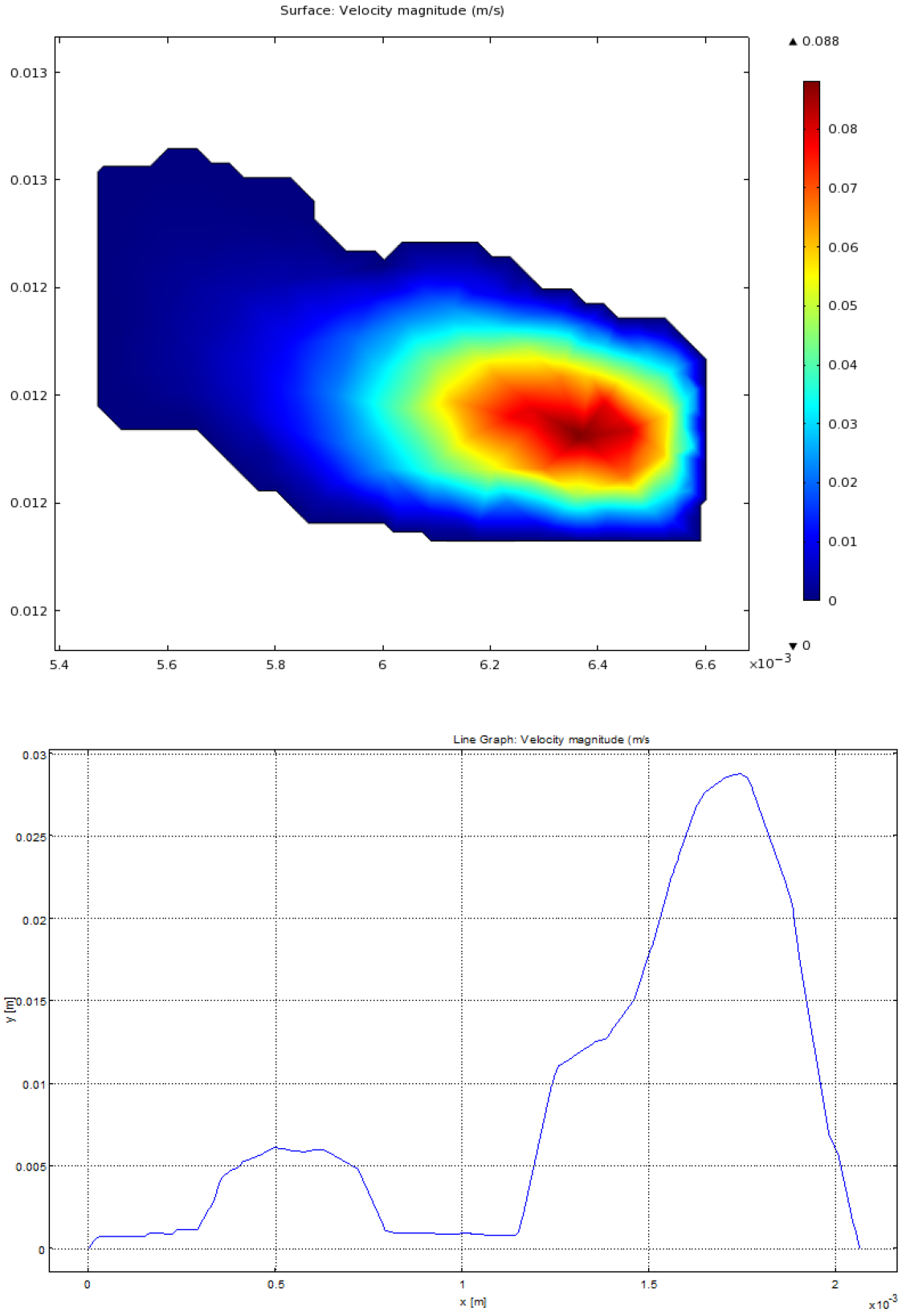


Figure 7.40: **Pore-9 outlet flow** - Pore-9 outlet flow, the exit plane was chosen to be on the yz-direction due to the pore morphology.

The SPH simulation will use the particle based boundary model for the solid particles. The solid boundary particles will not have any deformation and during the simulation the properties of the particles will be constant. They will be used to solve the collision and anti-penetration forces for the fluid particles.

7.6.1 Smoothing Particle Hydrodynamics Pore-3 solution

The SPH solution provides a database containing the physical properties of all the fluid particles at each time step. These properties include the particle velocity, total area velocity, and volumetric velocity, for each defined time step.

Figures 7.41 and 7.42 shows area cuts from different z-depth levels with the velocity of the particle system.

Figures 7.41 and 7.42 also shows how the particles approximate the morphology of the boundary walls of the pores. The boundary condition of no-slip is maintained and the areas of the flow fluid furthest from the wall have the highest velocity.

Since the pore is not formed by perfect elliptical or circular shapes, flow behaviour is affected by the pore shape at different depths. Another important factor of the flow behaviour is the tortuosity of the pore, the highest tortuosity will produce the least uniform flow.

Smoothing Particle Hydrodynamics Pore-3 inlet

The fluid inlet measure in the frontier between the collector exit and the entrance to Pore-3 is shown in Figure 7.43.

The resulting integration of volumetric flow and flow over the inlet area is:

$$Q = 4.39 \times 10^{-13} m^3/s \tag{7.28}$$

$$v = 3.22 \times 10^{-7} m/s \tag{7.29}$$

Smoothing Particle Hydrodynamics Pore-3 outlet

The flow of fluid leaving the pore at the outlet is shown in Figure 7.44.

The volumetric flow velocity and the velocity of the flow at the outlet after the complete SPH simulation are:

$$Q = 4.71 \times 10^{-14} m^3/s \tag{7.30}$$

$$v = 9.4 \times 10^{-9} m/s \tag{7.31}$$

Smoothing Particle Hydrodynamics Pore-3 results

After the SPH solution is obtained, Pore-3 permeability can be calculated from the outlet flow output. Using Equation 6.2:

$$\kappa = 2.1 \times 10^{-8} m^2 \tag{7.32}$$

7.6.2 Smoothing Particle Hydrodynamics Pore-6 solution

Figures 7.45 and 7.46 shows the volumetric velocity flow of the particle system at different z-depth levels.

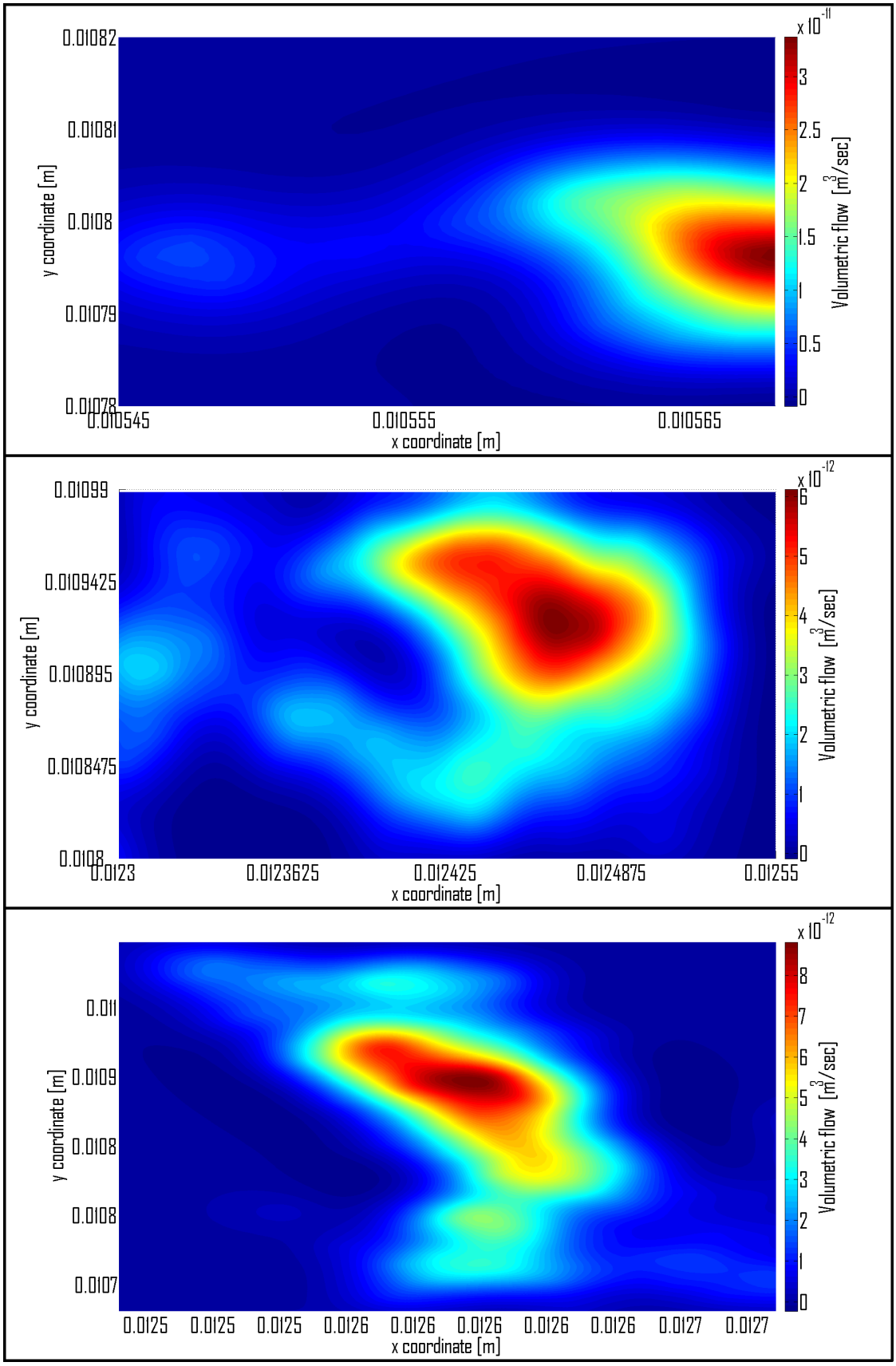


Figure 7.41: **Pore-3 SPH cuts velocity flow** - Volumetric velocity flow plot. The 2D-plot shows discrete position on the surface area and the volumetric velocity flow related to it.

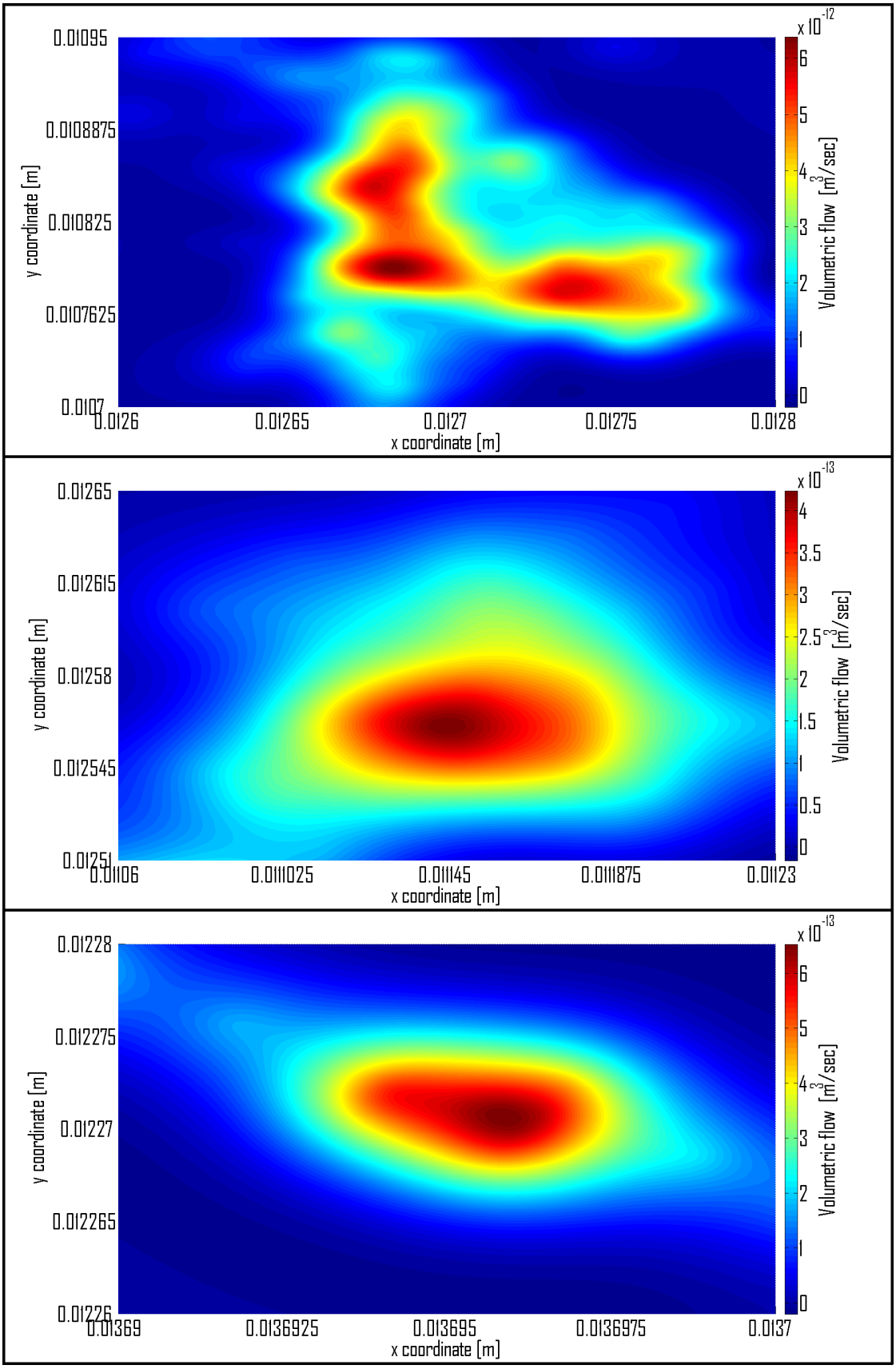


Figure 7.42: **Pore-3 SPH cuts velocity flow** - Volumetric velocity flow plot. The 2D-plot shows discrete position on the surface area and the volumetric velocity flow related to it.

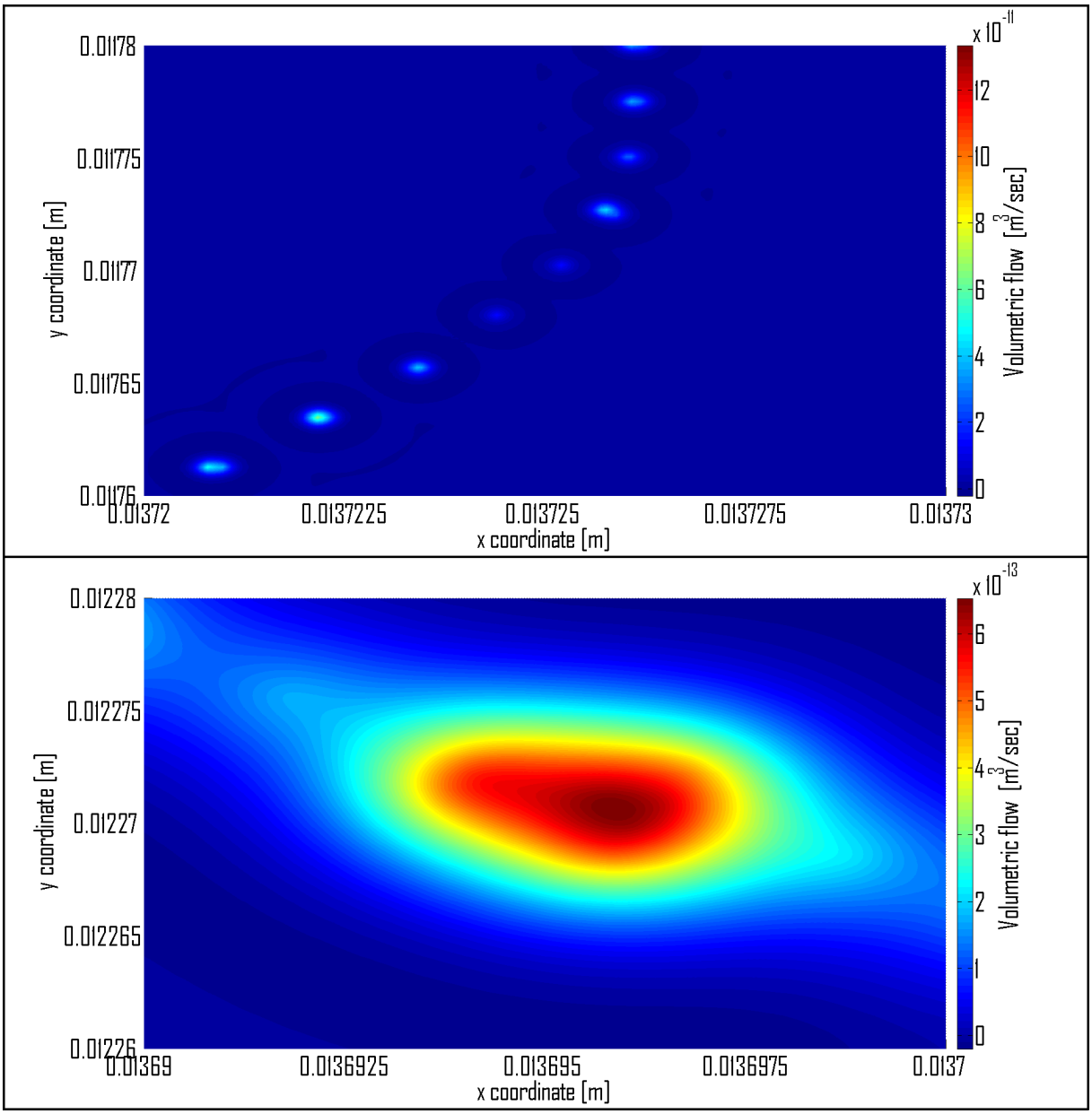


Figure 7.43: **Pore-3 SPH inlet flow - Top:** Pore-3 inlet flow result for the SPH simulation. **Bottom:** The inlet is formed by random seeds and it will not behave as the boundary shape, the simulation will make the fluid behave as the boundary after the inlet.

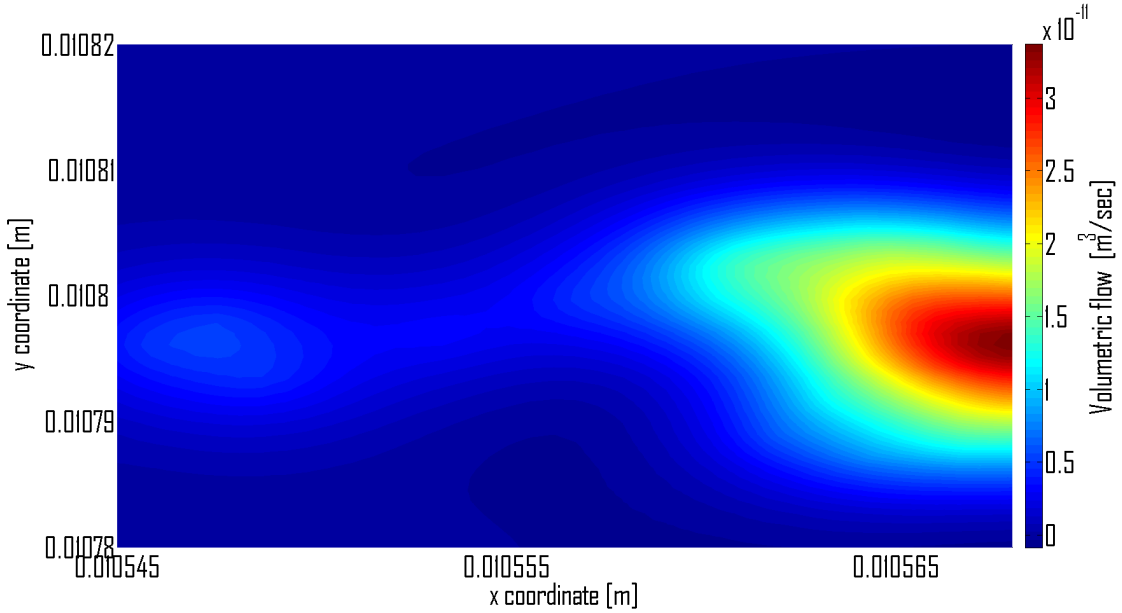


Figure 7.44: **Pore-3 SPH outlet flow** - The cut provides the solution for how much fluid is leaving the pore. In the exit plane a better approximation to the boundary conditions can be seen.

Smoothing Particle Hydrodynamics Pore-6 inlet

The fluid inlet measured between the collector exit and the entrance to Pore-6 is shown in Figure 7.47.

The resulting integration of volumetric flow and flow over the inlet area is:

$$Q = 1.07 \times 10^{-13} m^3/s \tag{7.33}$$

$$v = 3.53 \times 10^{-7} m/s \tag{7.34}$$

Smoothing Particle Hydrodynamics Pore-6 outlet

The flow of fluid leaving the pore at the outlet is shown in Figure 7.48.

The volumetric flow velocity and the velocity of the flow at the outlet after the complete SPH simulation are:

$$Q = 1.1 \times 10^{-13} m^3/s \tag{7.35}$$

$$v = 2.9 \times 10^{-9} m/s \tag{7.36}$$

Smoothing Particle Hydrodynamics Pore-6 results

After the SPH solution is obtained, the Pore-6 permeability can be calculated using the outlet flow output. Using Equation 6.2:

$$\kappa = 4.2 \times 10^{-9} m^2 \tag{7.37}$$

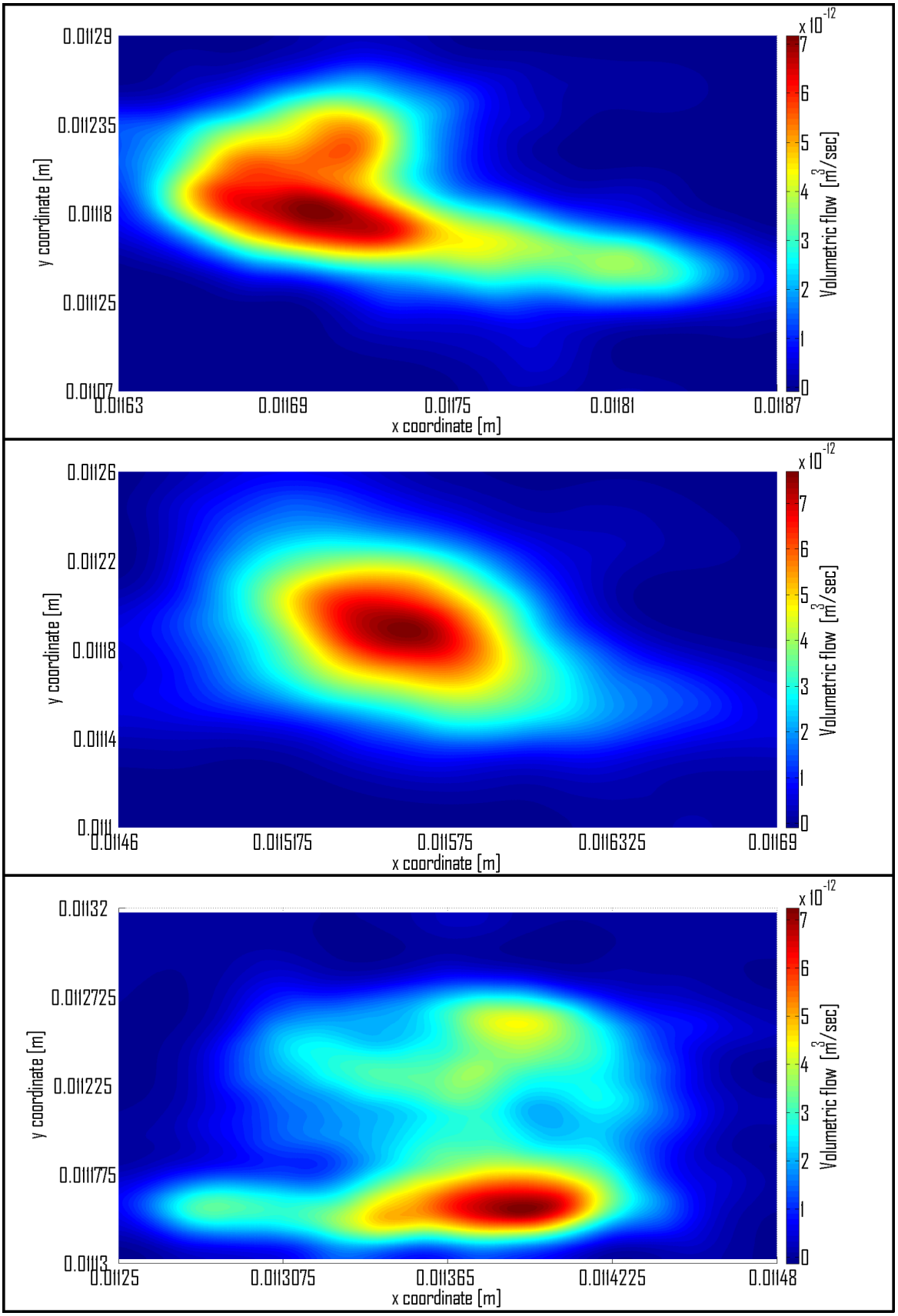


Figure 7.45: **Pore-6 SPH cuts velocity flow** - Volumetric velocity flow plot. The 2D-plot shows discrete position on the surface area and the volumetric velocity flow related to it.

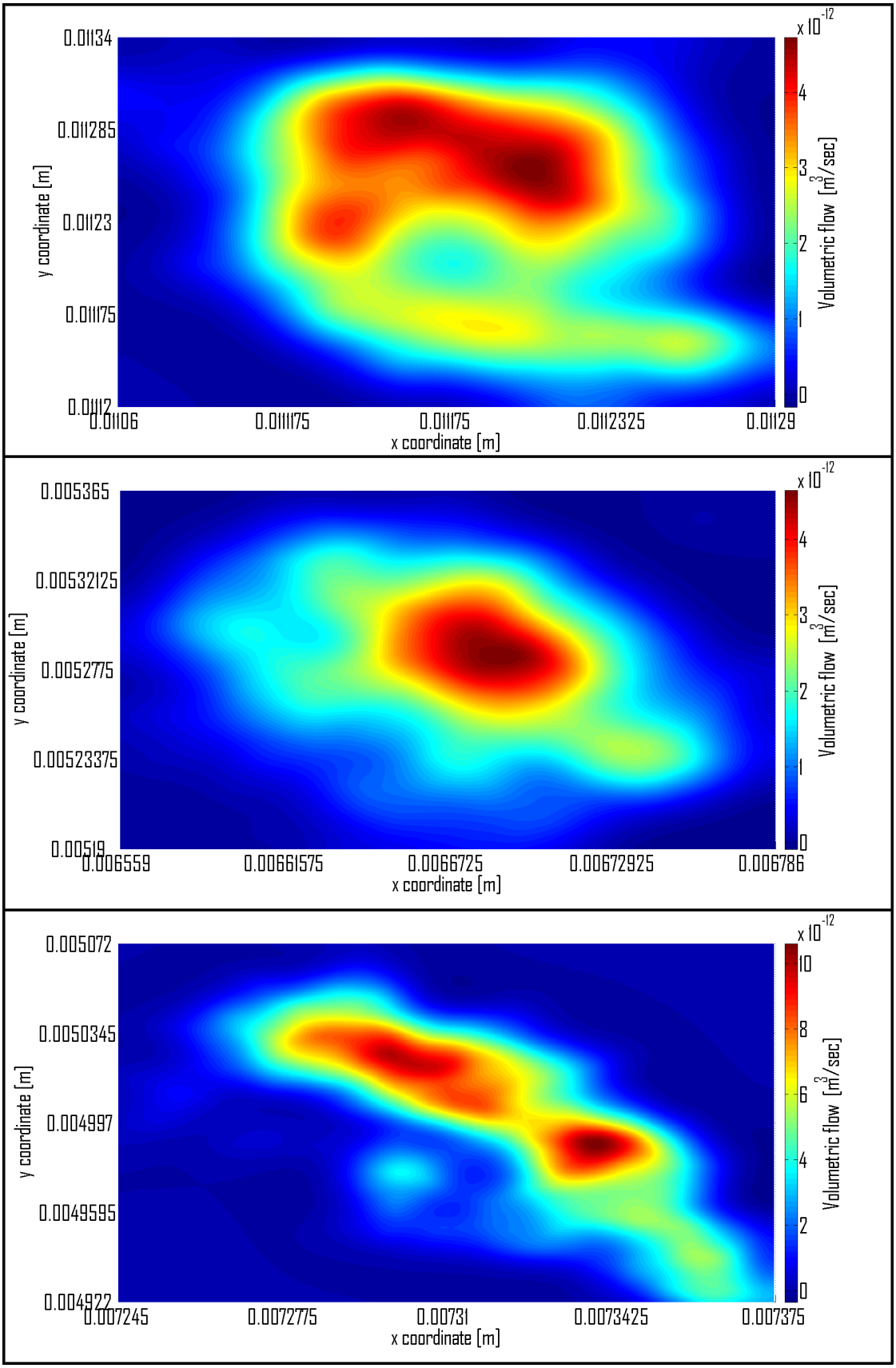


Figure 7.46: **Pore-6 SPH cuts velocity flow** - Volumetric velocity flow plot. The 2D-plot shows discrete position on the surface area and the volumetric velocity flow related to it.

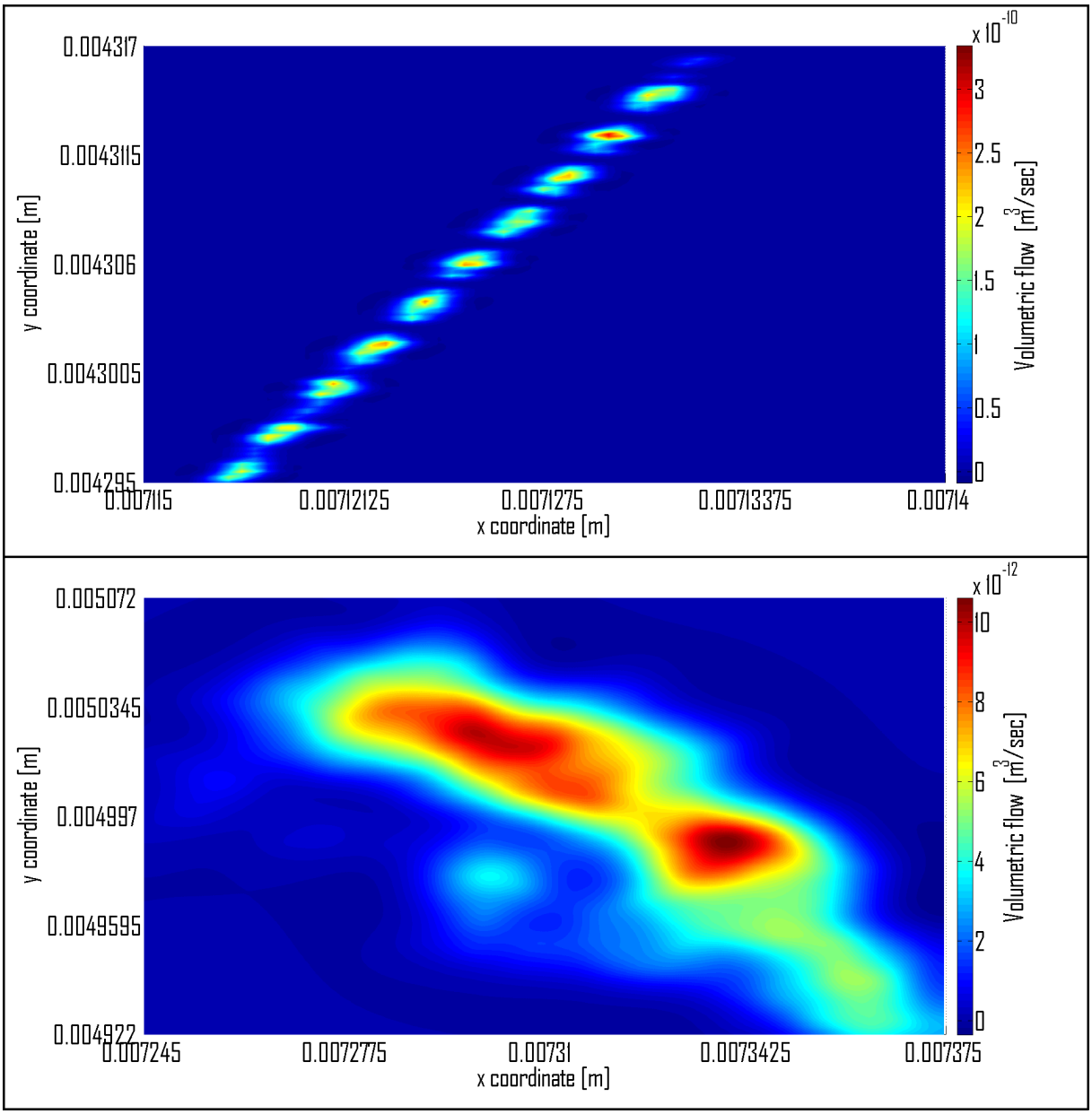


Figure 7.47: **Pore-6 SPH inlet flow - Top:** Pore-6 inlet flow result for the SPH simulation. **Bottom:** The inlet is formed by random seeds and it will not behave as the boundary shape, the simulation will make the fluid behave as the boundary after the inlet.

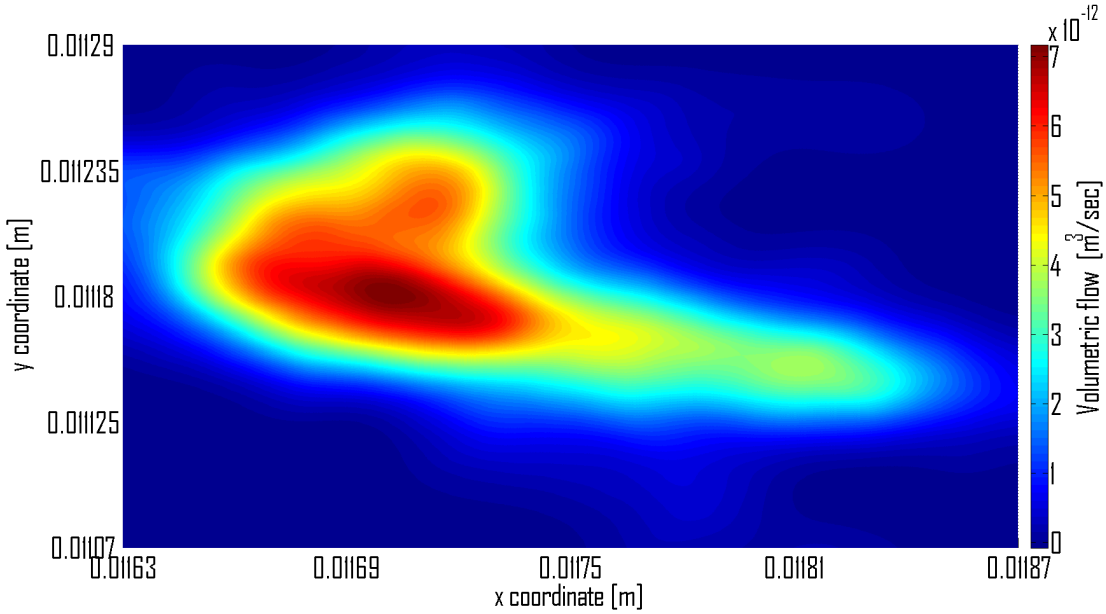


Figure 7.48: **Pore-6 SPH outlet flow** - The cut provides the solution for how much fluid is leaving the pore. Due to the shape of the pore the fluid has a non centric maximum component (affected by gravity).

.

7.6.3 Smoothing Particle Hydrodynamics Pore-9 solution

Pore-9 has a greater angle from the horizontal. This means SPH produces a slower flow transversing the pore. Figures 7.49 and 7.50 shows some cuts with the volumetric flow at different depth levels of the pore.

Smoothing Particle Hydrodynamics Pore-9 inlet

The fluid inlet measured between the collector exit and the entrance to Pore-9 is shown in Figure 7.51.

The resulting integration of volumetric flow and flow over the inlet area is:

$$Q = 2.96 \times 10^{-13} m^3/s \tag{7.38}$$

$$v = 3.37 \times 10^{-7} m/s \tag{7.39}$$

Smoothing Particle Hydrodynamics Pore-9 outlet

The flow of fluid leaving the pore at the outlet is shown in Figure 7.52.

The volumetric flow velocity and the velocity of the flow at the outlet after the complete SPH simulation are:

$$Q = 3.50 \times 10^{-13} m^3/s \tag{7.40}$$

$$v = 2.10 \times 10^{-8} m/s \tag{7.41}$$

Smoothing Particle Hydrodynamics Pore-9 results

After the SPH solution is obtained, Pore-9 permeability can be calculated using the outlet flow output. Using Equation 6.2:

$$\kappa = 1.00 \times 10^{-8} m^2 \tag{7.42}$$

7.7 Result analysis

Table 7.6 contains a summary of the principal results from the analysis of the pores using COMSOL and SPH

It can be seen from Table 7.6 that there is a difference in the amount of fluid volume at the outlet in the COMSOL and SPH simulations. There are several reasons for this. In the COMSOL simulation the differences depend on how many elements the inlet mesh has and it will produced different inlet flow. Different qualities of mesh will give different approximations to the inlet. COMSOL is not open source software so the factors that determine the fluid volume, besides inlet area and the number of elements in the mesh, are not available to us.

In the SPH simulation the fluid volumes depend directly on the number of particles and the volume of each particle. For this simulation the volume of each particle was the same as the resolution of the CT-scan voxel, $8.7 \times 10^{-5} * 8.7 \times 10^{-5} * 8.7 \times 10^{-5} = 6.59 \times 10^{-13}\ m^3$ per particle, and the number of particles was related to the number of internal pixels in each z-cut.

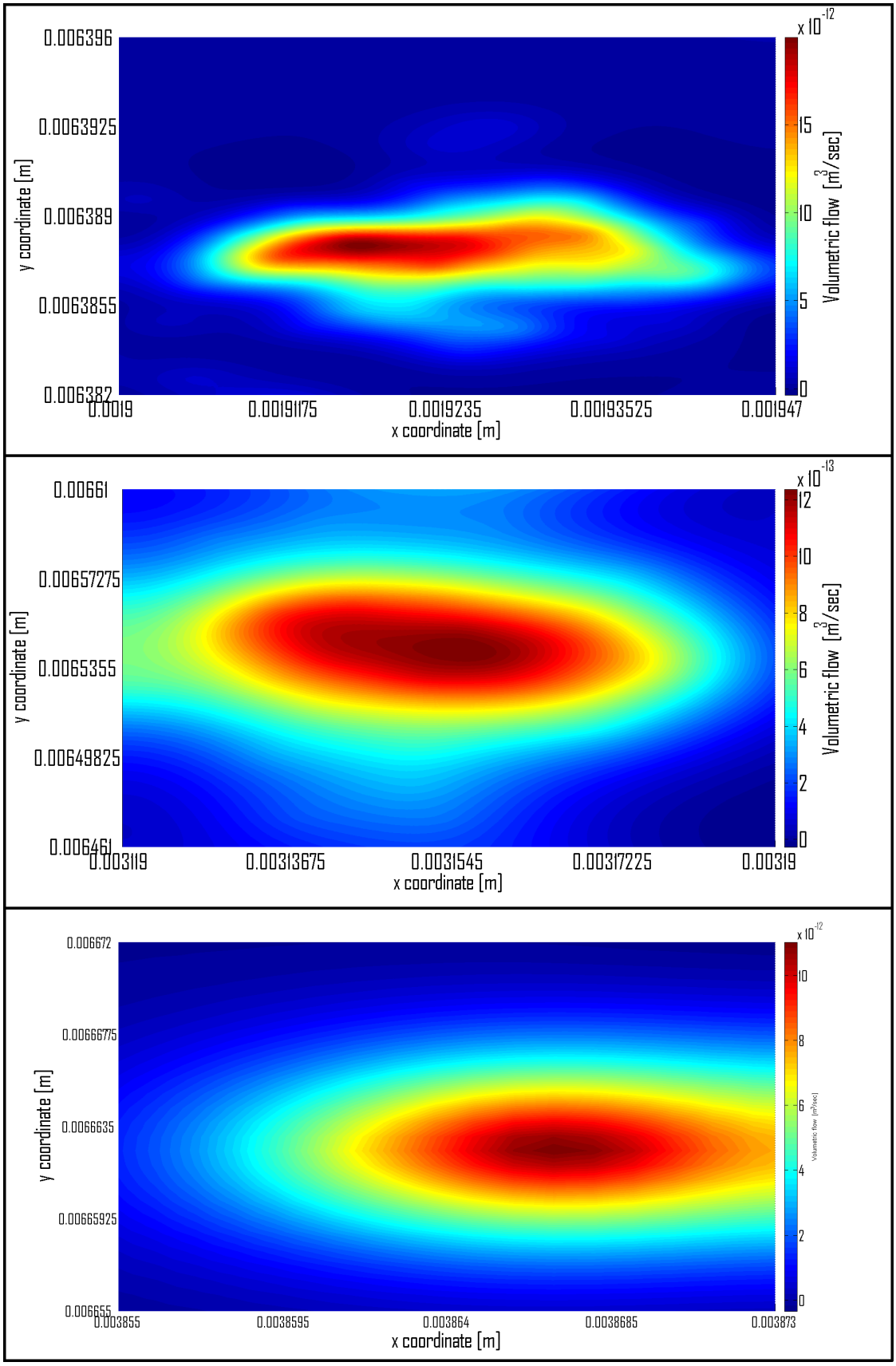


Figure 7.49: **Pore-9 SPH cuts velocity flow** - Pore-9 SPH cuts volumetric velocity flow plot, the flow changes as the boundary conditions shape changes.

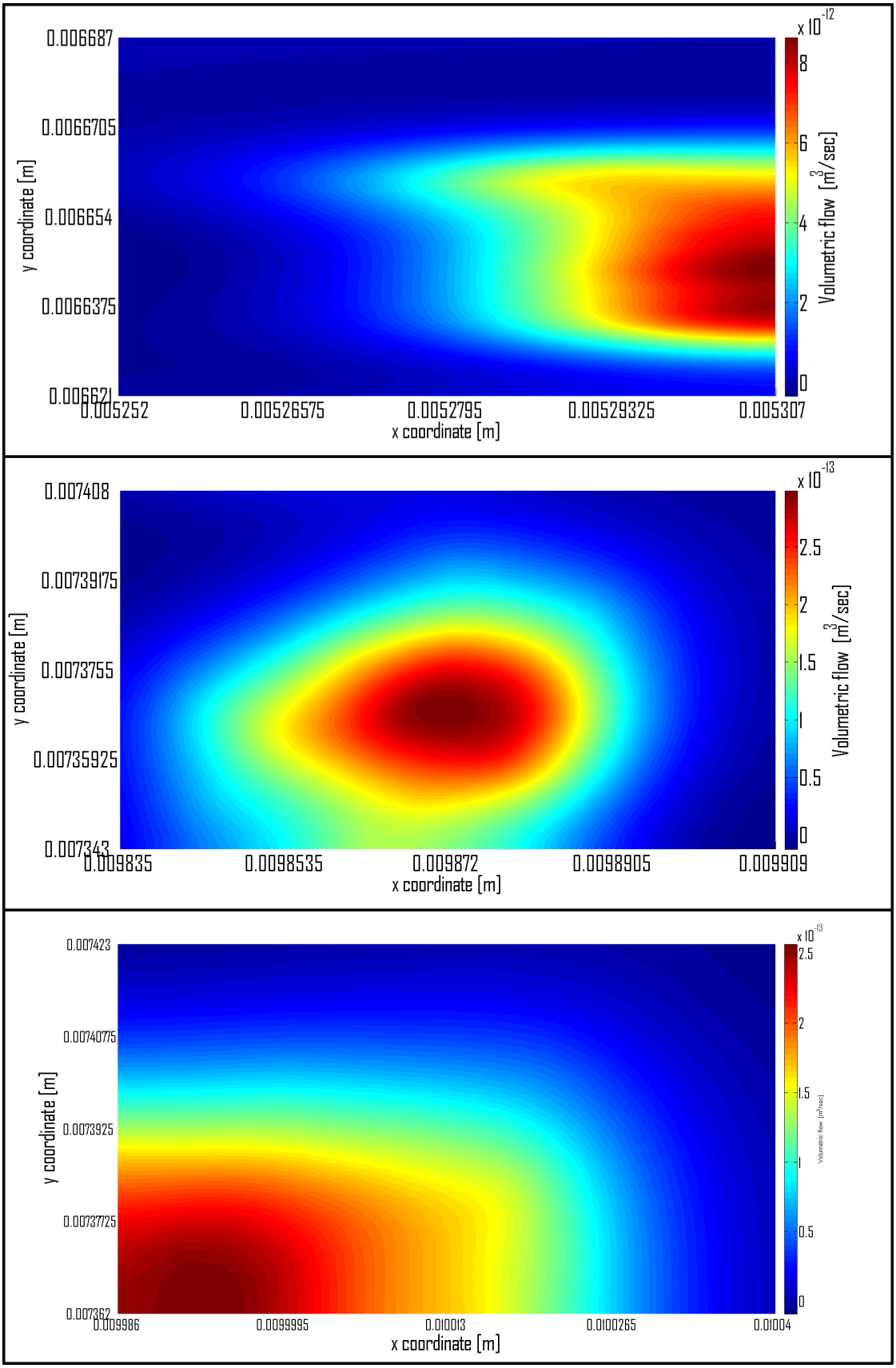


Figure 7.50: **Pore-9 SPH cuts velocity flow** - Pore-9 SPH cuts volumetric velocity flow plot, the flow changes as the boundary conditions shape changes.

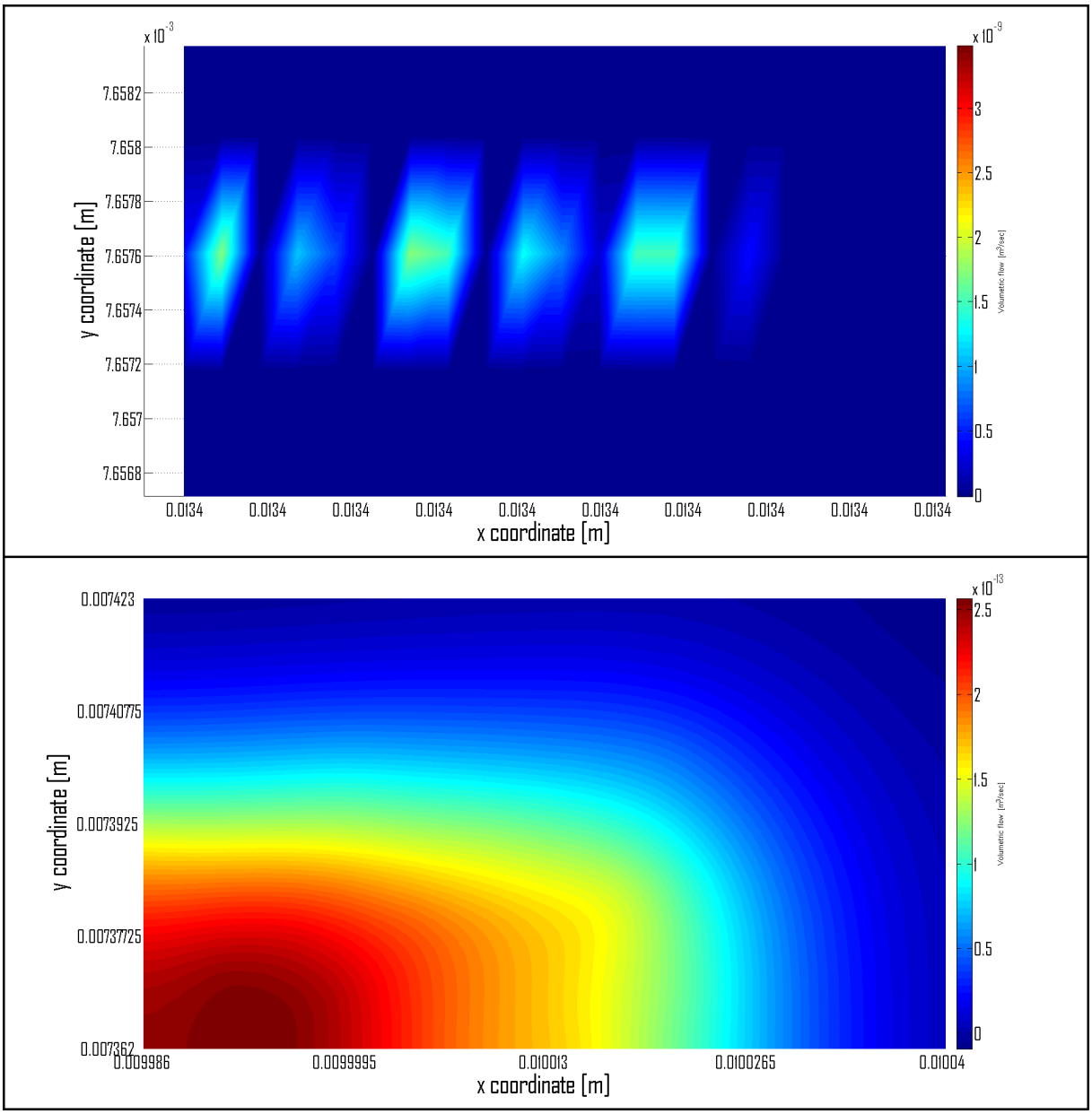


Figure 7.51: **Pore-9 SPH inlet flow - Top:** Pore-9 inlet flow result for the SPH simulation. **Bottom:** The inlet is formed by random seeds and it will not behave as the boundary shape, the simulation will make the fluid behave as the boundary after the inlet.

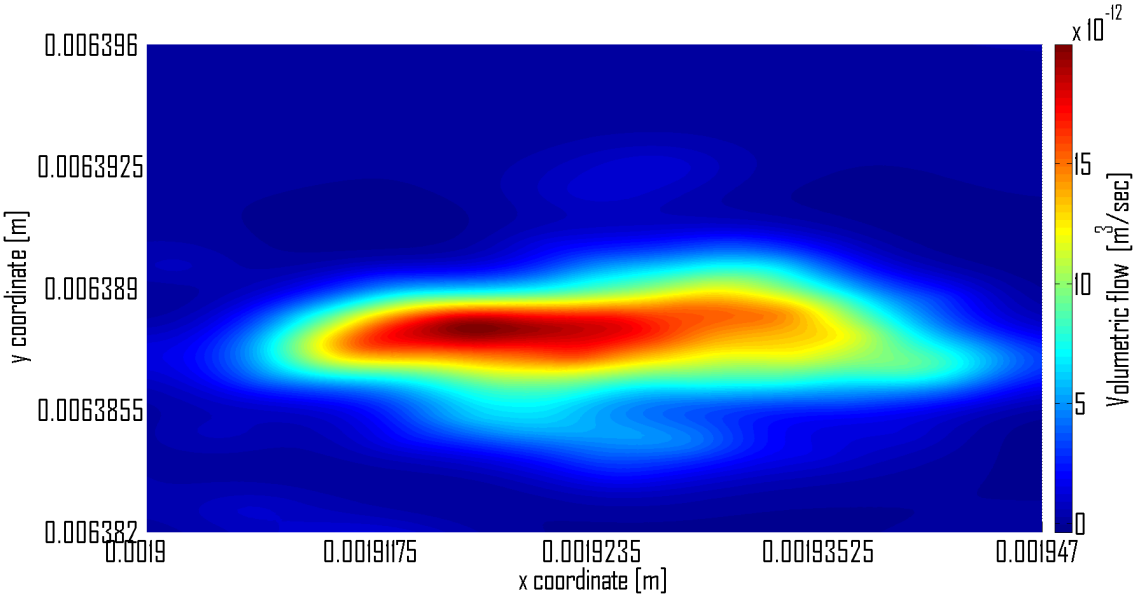


Figure 7.52: **Pore-9 SPH inlet flow** - Pore-9 outlet flow. The cut provides the solution for how much fluid is leaving the pore. Due to the angle of the pore with respect to the z-direction the fluid flow is slower than in pore-3.

The differences in inlet volume resulted in differences between all SPH and COMSOL calculations, but both systems maintain a similar flow at the inlet and the outlet. The two methods gave different solutions for the final porosity.

As shown in the previous sections a strength of SPH is its ability to approximate the boundaries and react as a real fluid when they change the fluid shape. The tortuosity produced a change in the flow and modified the principal flow centre.

Volumetric flows were compared to the flow calculated with Poiseuille law Equation 7.43 :

$$Q_p = \frac{\Delta P \pi d^4}{128 \mu L} \tag{7.43}$$

Where d is the mean diameter of the pore. using the equivalent mean diameter of the pore and the skeleton length. Although values were of the same order of magnitude, simulated flow rates were either lower than the Poiseuille flow (pore 9 with Comsol) or higher (Pore 3 and Pore 6 only for SPH) showing the role played by the variation of area along the pore length and the pore tortuosity, the results are shown on Table 7.7.

7.8 Multi-pores simulation

One advantage of the SPH algorithm is the capability to manage multi-pores with size bigger than the memory available, this provides a tool to simulate structures formed by more than one pore. Figure 7.53 shows the original pore configuration.

Each pore will have a flow with different characteristics dependent on the pore properties. The pores are named from pore 1b to 6b.

Pore-1b is a simple pore component similar to the ones studied in the previous sections, there is one main component where the fluid flows, Figure 7.54.

COMSOL and SPH comparison						
Pore #	Tortuosity	COMSOL tetrahedral	SPH total particles	Outlet volumetric flow	Darcy's velocity	permeability
3	1.16	296998	22978	COMSOL: 4.20×10^{-14} m^3/s SPH: 4.70×10^{-14} m^3/s	COMSOL: 8.60×10^{-9} m/s SPH: 9.40×10^{-9} m/s	COMSOL: 1.90×10^{-8} m^2 SPH: 2.10×10^{-8} m^2
6	1.26	296998	22978	COMSOL: 5.60×10^{-15} m^3/s SPH: 1.10×10^{-13} m^3/s	COMSOL: 1.50×10^{-10} m^3/s SPH: 2.90×10^{-9} m/s	COMSOL: 2.20×10^{-10} m^2 SPH: 4.20×10^{-9} m^2
9	1.21	61384	31065	COMSOL: 8.80×10^{-14} m^3/s SPH: 3.50×10^{-13} m^3/s	COMSOL: 1.50×10^{-9} m/s SPH: 2.10×10^{-8} m/s	COMSOL: 2.60×10^{-9} m^2 SPH: 1.00×10^{-8} m^2

Table 7.6: The table summarises the important results of the computational modelling of the pores and shows the differences in results between COMSOL and SPH.

Poiseuille, COMSOL and SPH volumetric flow comparison			
Pore #	Poiseuille Outlet m^3/s	SPH Outlet m^3/s	COMSOL Outlet m^3/s
3	4.10×10^{-15}	4.70×10^{-14}	4.20×10^{-14}
6	6.30×10^{-15}	1.10×10^{-13}	5.60×10^{-15}
9	1.70×10^{-13}	3.50×10^{-13}	8.80×10^{-14}

Table 7.7: The table summarises the volumetric flow comparisson between the SPH and COMSOL computational method and the analytical result of the Poiseuille equation.

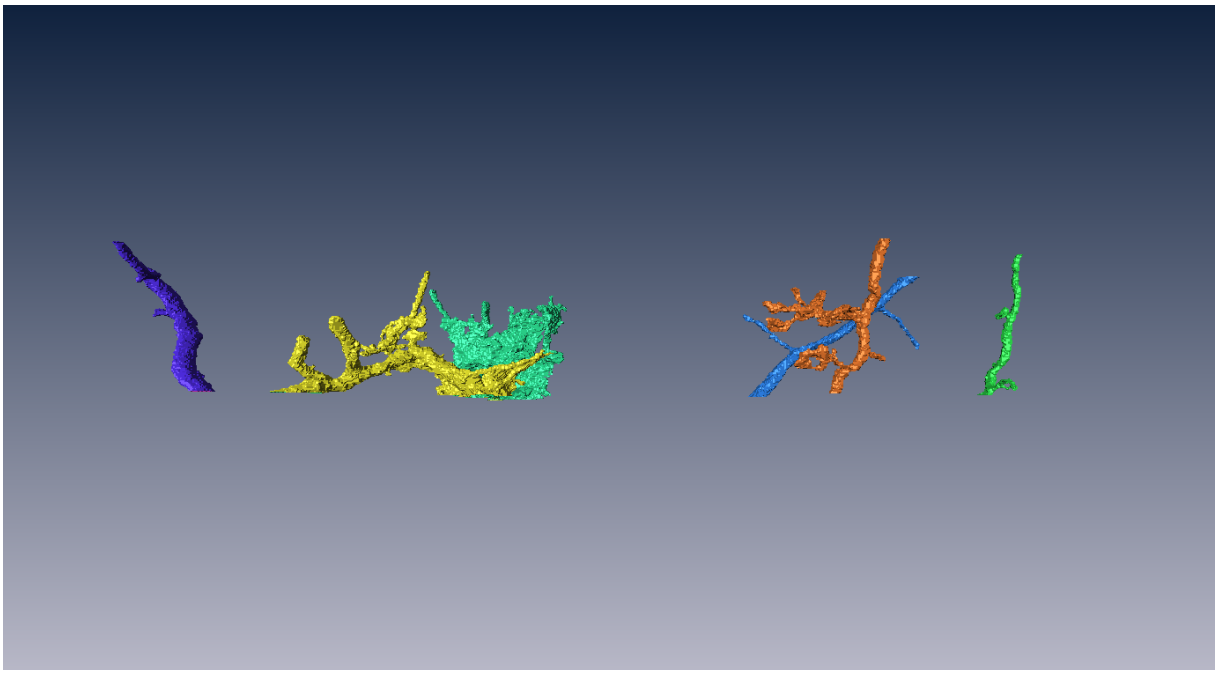


Figure 7.53: **Multi-pores experiment** - Using the SPH capabilities of dividing a volume problem using the tree of trees approach multiple pores are can be solve.

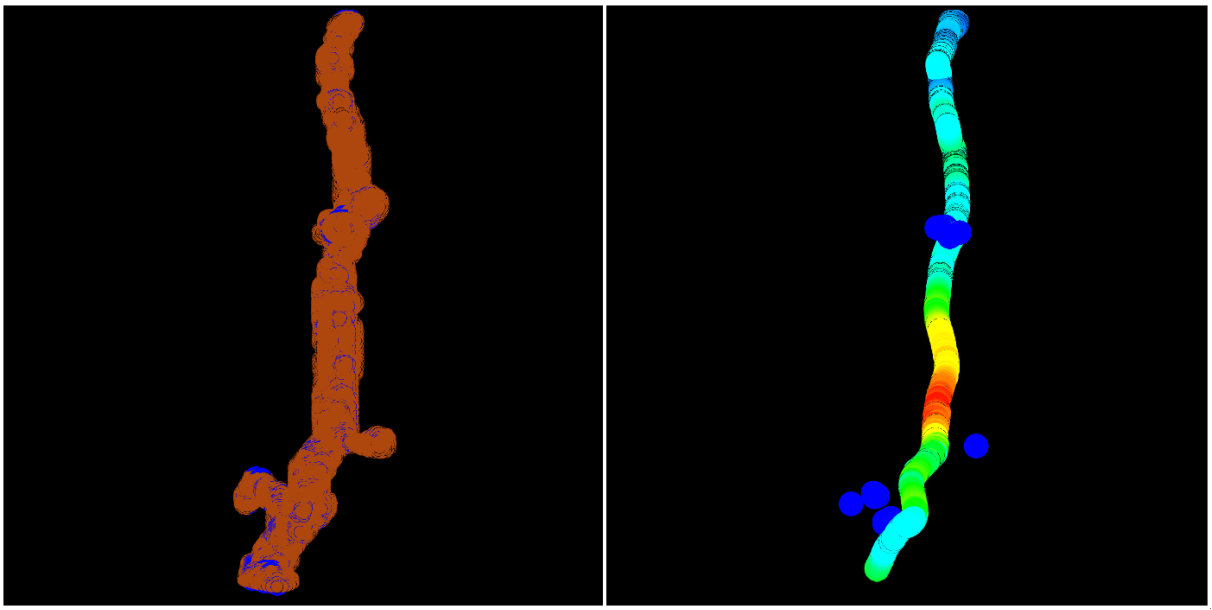


Figure 7.54: **Multi-pores experiment, pore-1b** - The figure shows the fluid flow of the first component of the SPH simulation, the flow traverses the pore from the inlet to the outlet without any problem.

Pore-2b is similar to Pore-1b and has a simple structure but after running SPH it can be seen that there is no passage for the fluid since the CT-scan result gives a collapse area and the pore structure is completely closed. The pore starts full of fluid but after some time step it become empty since no more fluid is entering the bottom part. Figure 7.55 shows the result of the simulation, this is an important step in multiple pore studies since more pores could be connected to this one and they will also not receive fluid input from this component.

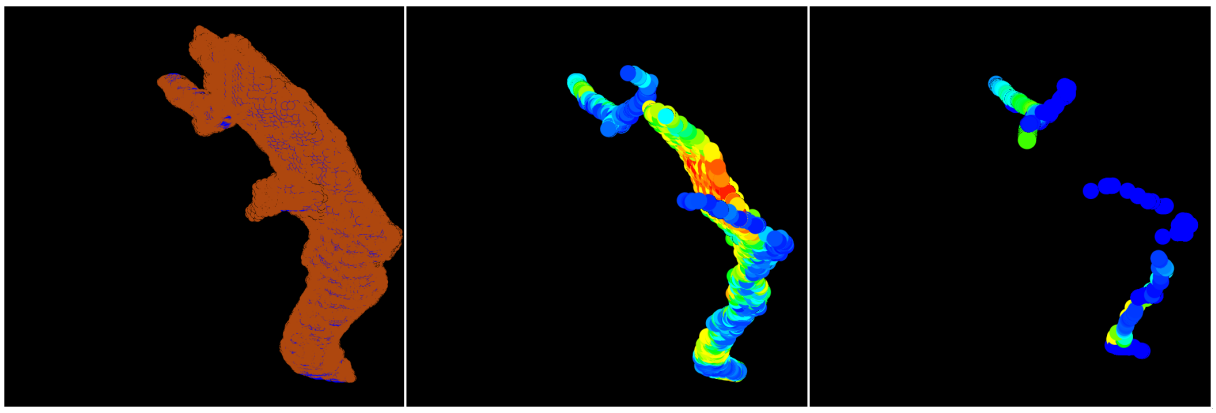


Figure 7.55: **Multi-pores experiment, pore-2b** - In the second component a problem with the flow is presented, the pore is not open internally.

Pore-3b is an example of a pore that is subdivided into more components to enable its simulation. Figure 7.56 shows how the tree of trees subdivided the pore into different octants (white nodes), then the program can calculate how many nodes can be solved at the same time, the complete flow is shown in Figure 7.56.

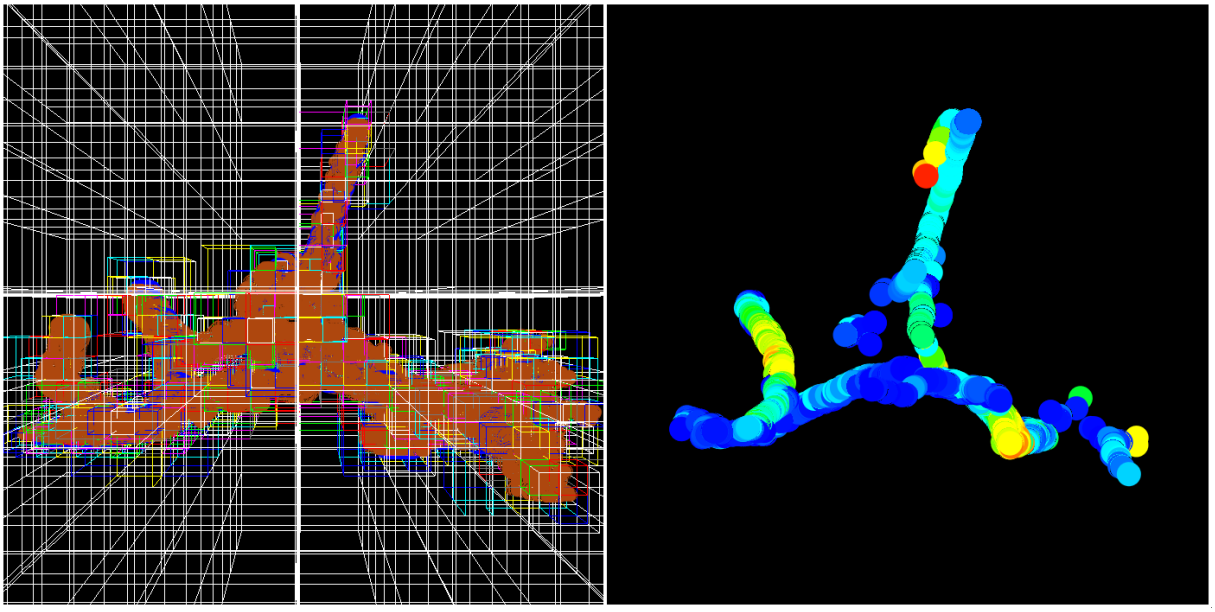


Figure 7.56: **Multi-pores experiment, pore-3b** - Tree of trees application to solved bigger pore components, SPH can adapt the tree structure depending on memory availability.

Pore-4b (Figure 7.57) shows a similar result to Pore-3b but the difference is in the principal flow component. In the case of Pore-4b there is only one and the rest of the pore will only have stationary

fluid, to save memory SPH will consider these parts as full and continuing solving the flow path.

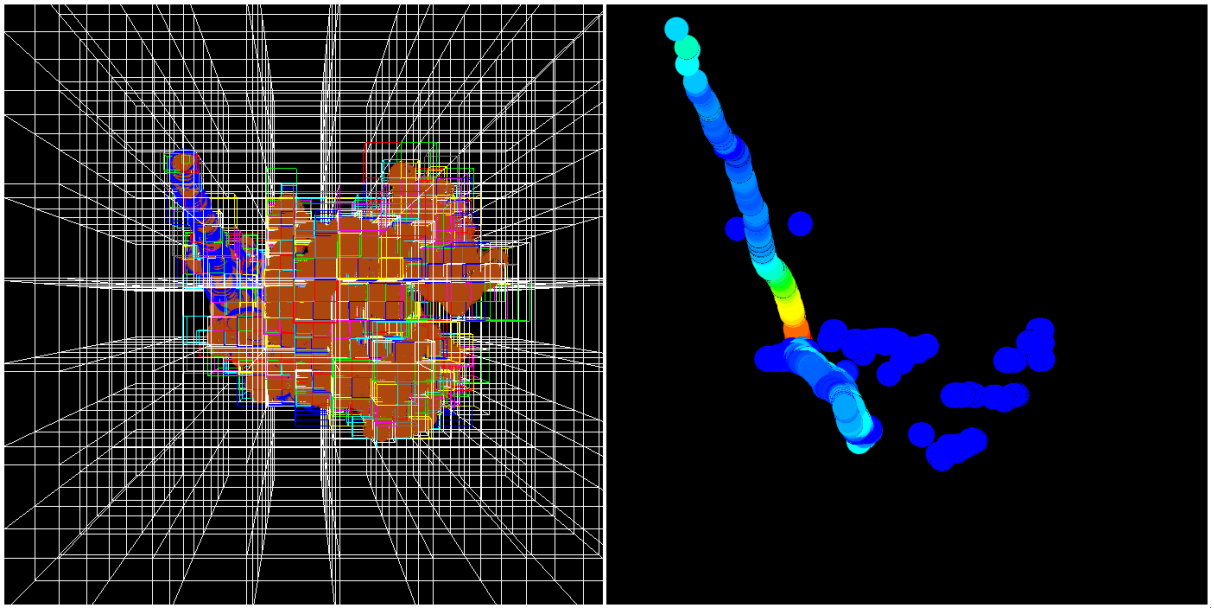


Figure 7.57: **Multi-pores experiment, pore-4b** - Pore component with a flow path surrounded by stationary fluid, SPH will use an approximation for the volume that will not flow.

Pore-5b is another example of a simple pore were the fluid flows without any problem, the components with static flow are solved in the same fashion as the cases presented for Pore-4b, Figure 7.58.

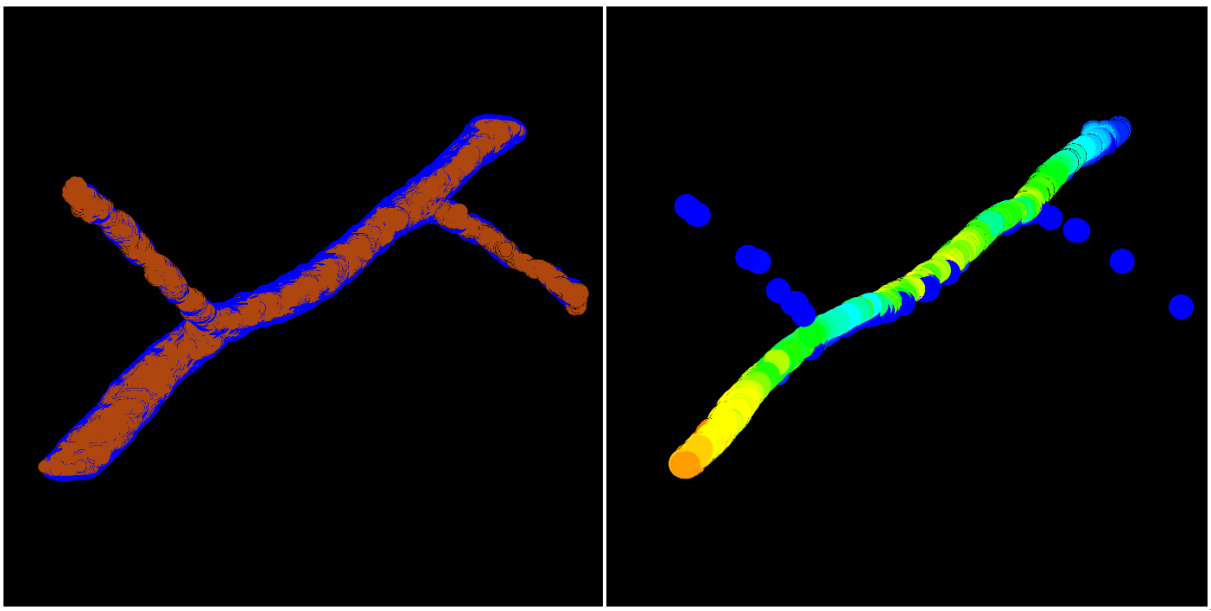


Figure 7.58: **Multi-pores experiment, pore-5b** - Flow of fluid on component number 5 of the pore structures.

The final component, Pore-6b, is a mix between a simple pore and stationary flow components, the SPH algorithm can use the tree of trees and the kinematic flags to approximate the areas where the fluid needs to be solved (since it is changing) and stop the simulation where the fluid is stationary, Figure 7.59.

Using SPH to solve multiple components allows us to have a more complete representation of the

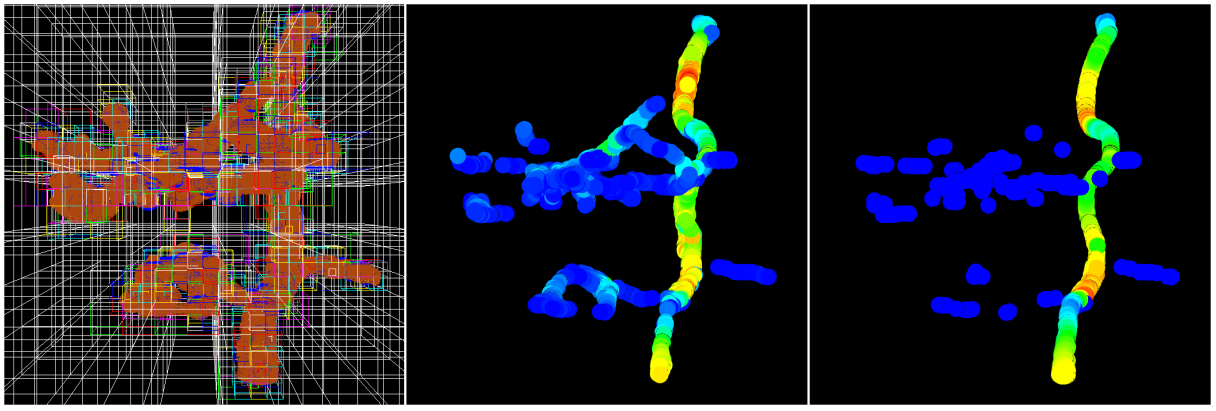


Figure 7.59: **Multi-pores experiment, pore-6b** - In the image from the right the principal component flow and the stationary flow (blue particles) can be seen, the stationary flow components are no longer part of the simulation and their velocity is 0.

problem. The tree of trees structure is being upgraded so it can have statistical measurements of the flow and the connection between components can be done after the top pores are solved.

There is no comparison to COMSOL since the tool does not provided a solution for problems with larger memory requirements than is available on the system.

7.9 Conclusions

In this chapter a study of a pore structure obtained from a CT-Scan system was presented using an implementation of SPH and a commercial software (COMSOL). The computational methods used to solve the flow problems were different, SPH used a mesh-free approach whereas COMSOL used a mesh based method.

After obtaining the results for all the pores, the two methods exhibited consistent differences between their volumetric flow results. The reasons were presented in Section 7.7. In order to determine which method better approximated a real flow situation further studies are necessary.

In a future work, a SPH system will be used to obtain the results from known geometric shapes in order to obtain a better calibration of the particle system and determine the number of particles necessary to solve a specific volume.

The ability of SPH to approximate a specific shape can be seen in Figure 7.60

To solve for the reduced number of particles on the limit of the fluid the particle ghost method presented in [43] is used, with this method a set of fluid particles is created at the boundary to allow the system to retain a sufficient number of particles to obtain a proper approximation with the computational solution.

The used of a CT-scan to obtain the morphological data allows for a better representation of problem. Instead of having a model that describes the shape of the problem the CT-scan data provides a database containing all the morphological data.

The next step for this part of the research is to obtain calibrated data for the soil and use the soil components during the simulation of the fluid.

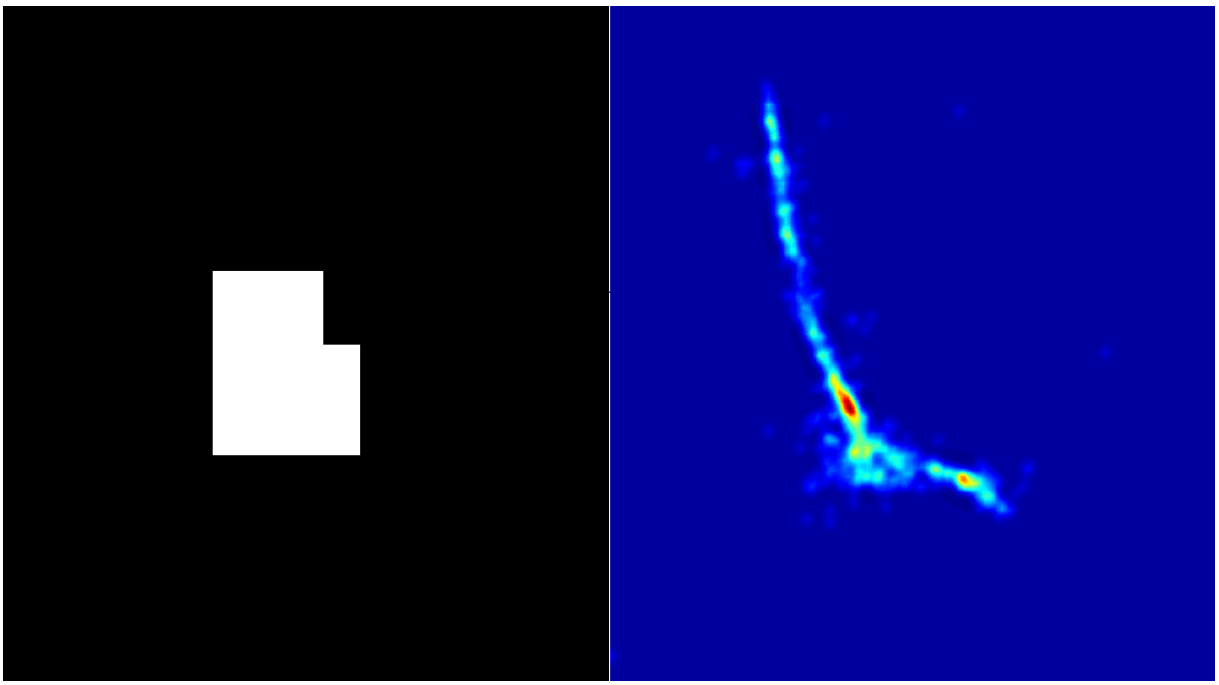


Figure 7.60: **SPH inlet flow and boundary shape** - The mesh-free method of SPH allows for a good approximation of the boundary, the small blue dots outside the area are caused by the integration (it takes some points of the cut above the interest one).

Chapter 8

CONCLUSIONS

8.1 Conclusions

The use of the SPH mesh-free method as a possible solution was chosen as the phenomena being studied in the Intelligent Vision Systems New Zealand laboratory deal with fluid dynamics and solid surface deformation. After a review of the literature it was considered that a mesh-free approach could provide the most accurate computational solution.

During the study of the SPH algorithm different considerations and versions were obtained with a focus on solving the problems presented during the research. The history of the SPH implementation at the Intelligent Vision Systems New Zealand laboratory can be summarized as:

- **ANSI C implementation:** The first approach was to adapt the code presented in the book [5], this first version consisted only of the fluid component with a simple linear search solution for the neighbourhood and the collision lists. The boundaries were formed by virtual particles.

  The lists search algorithm used the same loop to solved the neighbourhood and the collision lists, the discrimination for the algorithm to decide in which one of the list the interaction will belongs too is the particle type property, if two particles have the same value the neighbourhood test will be execute and if they are different the collision test will be run.

  To solve fluid-boundary interaction only the anti-penetration forces are used.

- **Particle structure:** One step that was implemented is the use of structures to define the particle objects in C++. The particle structure contains all the properties associated with the particle type. This implementation was removed from the SPH implementation when the parallel solution was built because of the problem of transferring fragmented memory from the host to the device. Using structures means that all the particle information needs to be transfered to the device, increasing the computational time of the operation and reducing the amount of particles the GPU can handle.

- **Elastic solid implementation:** After the first SPH implementation was obtained the method was expanded to solve elastic solids following the implementation presented in [7]. The implementation was based on the solid solution with SPH but at this point the interaction collision responses between solid particles and fluid particles or different solids was only based on the anti-penetration force and the solid was exclusively elastic.

- **Octree implementation:** To reduce the computational time the octree implementation was introduced to the solution, the first implementation of the tree solution had a linear tree traversal and the Morton key was not yet introduced.

- **Object subdivision:** After adding the octree solution the computational time improved to a point where more objects could be represented in the simulation, it also allowed for more complex fluid-solid interaction. The next problem was to reduce the loop iteration count when visiting all the particles in all calculations.

  An object type is defined to distinguish between objects of different types, this allows us to reduce the loop iteration count to solve each object and to focus on the specific solutions for each type. For example in the case of the internal forces it allows us to implement the specific solution for each particle type without the need to solve an if stament for each particle. This procedure is also used in the parallel solution to organise the objects' memory that is going to be transfered to the device for each kernel.

- **Morton key:** The most costly operation in the octree implementation is the tree traversal operation. Without any space references, all the tree transversal operations for all the particles need to start at the top level node, this increases the number of nodes that need to be tested and thus the computational cost per particle increases. The Morton key implementation to relates the particle position with a node location in the tree.

- **Parallel implementation:** A parallel solution based on the CUDA architecture was also developed at the Intelligent Vision Systems New Zealand laboratory. The parallel solution allows us to solve several particles at a time (Figure 8.1), reducing the computational cost of the SPH solution for problems with a high number of particles (e.g. see Section 6.6). A different solution to the parallel problem than the one presented in [8] is introduced. The new solution is necessary because of the difference in the parallel systems, the work presented in [8] is based on a solution for multi-core CPUs whereas the one developed in this thesis is based on the CUDA architecture.

  The biggest difference between the methods is the number of parallel threads and how they are handled. In the CPU implementation the number of parallel threads is low enough (eight core CPUs) that each thread can solve an octant of the problem, and since the implementation is based on a CPU there is enough memory to solve complete octants. In the CUDA implementation the solution is on the other side of the spectrum, a greater number of cores with less memory available. The ghost node implementation was introduced as a solution to the division of the tree into multiple threads.

- **Viscoelastic solids:** Since the solid behaviour being studied at the Intelligent Vision Systems New Zealand laboratory was not perfectly elastic (e.g. see Section 6.5) a new viscoelastic implementation was added to the SPH solution to simulate this type of solids.

- **Stereo-vision input:** A module to connect to the results obtained from stereo-vision techniques and equipment developed at the Intelligent Vision Systems New Zealand laboratory was added. The change in depth of a real object undergoing external forces was measured to obtain the material properties with an inverse implementation of the SPH application (e.g. see Section 6.5).

- **Kinematic flags:** The next code implementation is the introduction of the flags system based on the movement of the particles and their relation with the tree structure. The flags track when a particle needs to update its neighbourhood, collision list, or whether it going to be solved in the current time step.

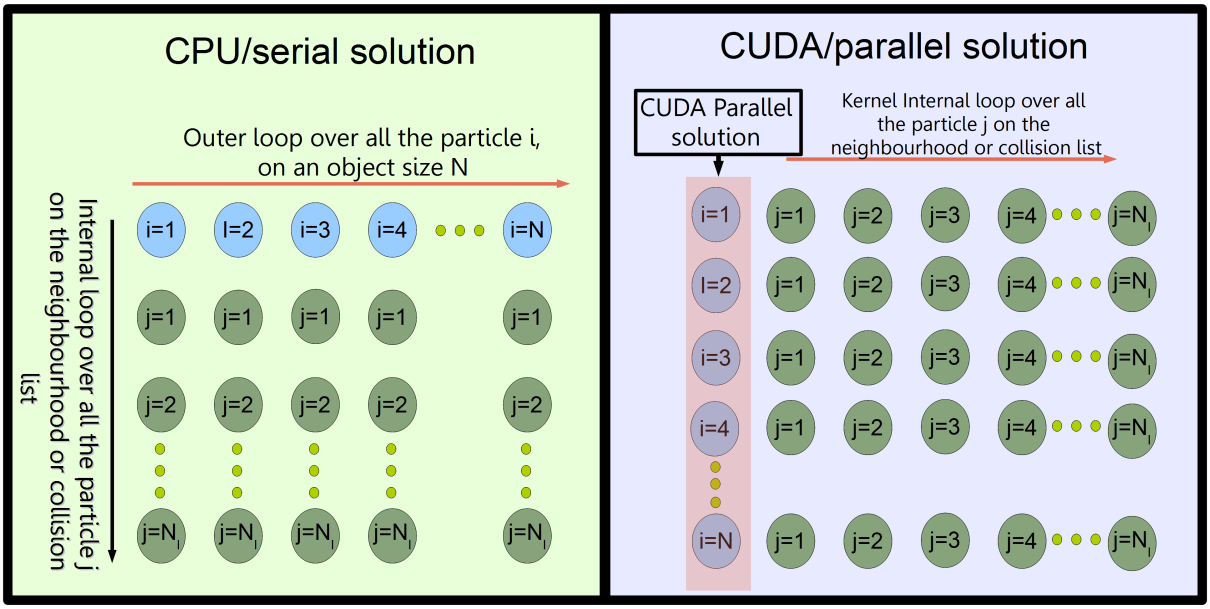


Figure 8.1: **Parallel and serial solutions differences - Left:** The serial solution has two loops that need to be solved, the outer loop solves all particles i of an object and the internal loop solves the collision and neighbourhood list per particle i. **Right:** The CUDA implementation parallelised the outer loop, leaving each thread to solve the inner loop per particle.

- **Tree of trees:** The final implementation added to the SPH solution is the tree of trees structure. This structure subdivides the problem volume when the relation between the object resolution and total volume is big or when the system memory is not enough to solve the complete problem.

Figure 8.2 shows a schematic view of the final implementation of the SPH algorithm presented in this thesis.

Chapter 6 presents the different results of each implementation and the improvements that each module bring to the SPH solution. To compare the accuracy of the solutions two different experiments were implemented, the first one tests the module to relate SPH to the stereo-vision technique, where the objective was first to obtain the material values of the real object and then obtain the SPH solution of the strain field. The second test was the use of the SPH implementation in the solution of the flow of fluid in macro-pores structures. To test the solution of SPH the commercial implementation COMSOL was used to test the same objects. Chapter 7 presents the advantages of using SPH to simulate flow inside the pores. The more important advantages are:

- **1:** SPH closely approximated the boundary conditions since the particles move and adapt closely to it, in the COMSOL mesh methods the boundary is defined by the original tetrahedra and the resolution to the boundary depends on the original mesh resolution but the computational cost and memory requirement for solving more dense meshes make it impossible to use COMSOL with current consumer hardware.

- **2:** Since the implementation of the SPH algorithm contains several tools to solve the problem of memory limitations, the resolution and problem size can be bigger than the limits presented by COMSOL, these limit s are only associated with the application and not with the technique it used to solve the problem.

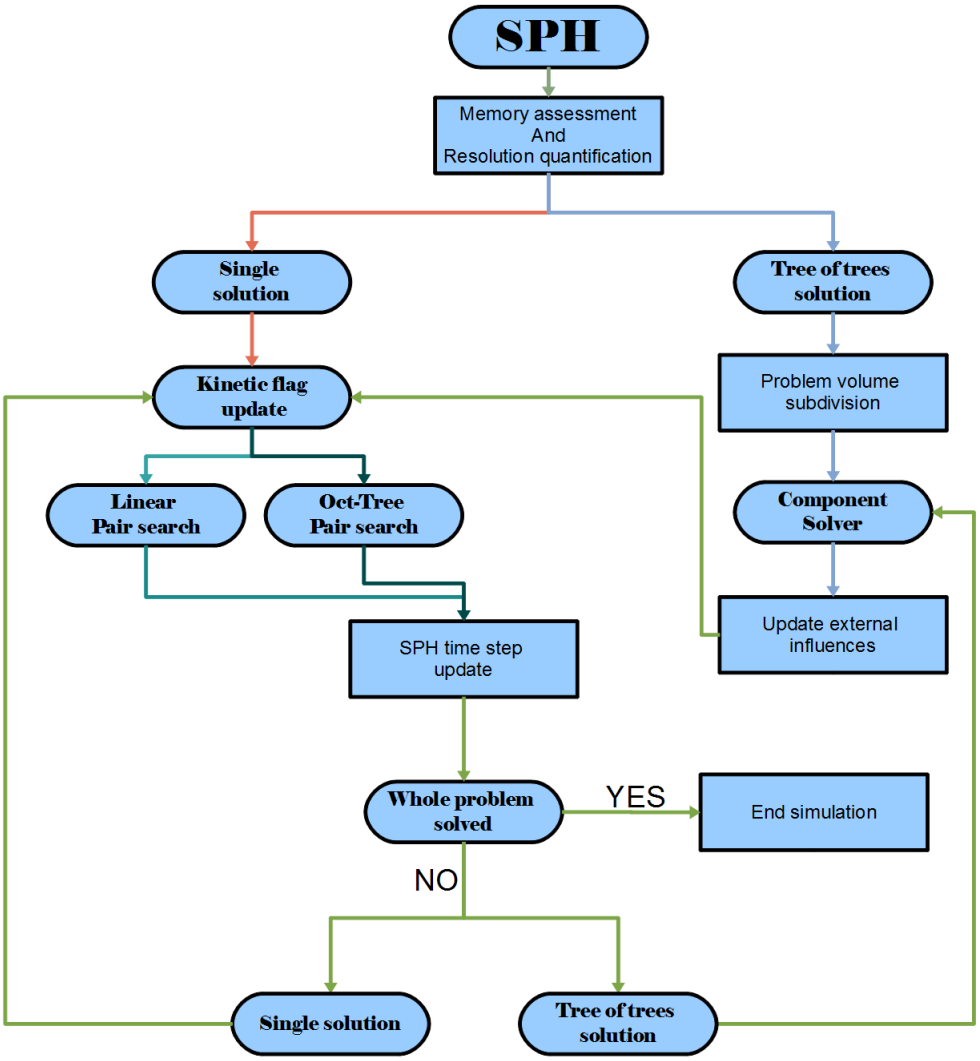


Figure 8.2: **Schematic view of the SPH solution** - The final SPH implementation will decided which components are the ones that will solve the problem being simulated with the lest computational cost.

- **3:** The use of the GPU to obtain solutions to problems with a number of particles above 20000 efficiently, the GPU implementation has a faster computational time for complex problems.

- **4:** In some problems, such as pore-9, the solution presented by COMSOL shows a different behaviour on the isosurface, how COMOSL approximated the solution when the isosurface is not found in part of the problem is not known to us, in SPH the solution for the angle that Pore-9 has is solved by increasing the number of boundary particles and thus increasing the resolution on the boundary interaction. In COMSOL a more refined mesh was defined but the results were similar, the mesh was set to the maximum possible without any meaningful change in the behaviour.

- **5:** The inlet-outlet equal value relation to maintain the volume inside the pore was only obtained by SPH. Using COMSOL, the inlet and outlet are not equal. This is one of the advantages of having our own source code to solve specific problems since it is possible to modify the solution to approximate the conditions that are specific to the phenomenon being studied.

The current SPH implementation has some numerical errors and limitations. The most important current limitations are:

- **1:** The dynamic solution of SPH is found at the cost of longer computational time since each new step needs to be solved.

- **2:** The post-simulation analysis on COMSOL is more user friendly and complete than the current solution for SPH. At this time the post-simulation analysis is done using MATLAB and is not integrate as a module on the SPH implementation. This causes problems in the portability of the system since MATLAB code is needed to use the analysis tool provided with SPH. To solve this problem we are considering implementing the code analysis in the SPH implementation using C and an open source library such as OCTAL.

- **3:** Another disadvantage is the number of modules available to the user. At this point the user interface module for defining different problems has been developed on a necessities basis, so a new type of problem can require the modification of the input interface. It is planned to extend the code to a more user-friendly interface.

- **4:** Solution correctness is also a point of study. New research following the approach of using known shapes such as the ones presented in Sections 6.4.2 and 6.4.1 is being implemented, in the planned research a 3D-printer is going to be used to build the shapes and test them so it becomes possible to compare directly against real results.

- **5:** The stereo solution module is also being researched at the Intelligent Vision Systems New Zealand laboratory, as the research progresses and more data is available the solution will be extended to more applications.

- **6:** Volume size resolution. In the SPH implementation of the pore studies the volumes were obtained from CT-scan data, these volumes are approximated from square voxels to spherical representations. In Section 6.4.3 the management of this problem in the current SPH implementations by increasing the boundary resolutionis presented but this solution is still an approximation of the volumes. This will always be a problem in mesh-free methods when approximating a continuum volume, for problems where the specific data is not known the total geometric volume (voxel counts) is used to obtain the volume per particle.

- **7:** In the parallel implementation a current limitation is the cost of using double precision, since the GPU will take twice the time to solve the same operation and the number of available cores are limited (depending on the GPU model). The SPH implementation lets the user decide whether to use float or double precision since the CUDA architecture is evolving in a direction that will solve this problem in the future.

At this time the work presented in the Thesis provides a mesh-free parallel solution that can be adapted to different problems and provides a modular structure that allows the use of the basic SPH computational solution for new problems integrating only the new physical laws of the problem. Some of the limitations and new elements are currently being researched, the following points expected to be integrated into the SPH implementation in the future:

- **1:** The user interface is based on ASCII ".dat" files containing the initial information about the simulation, at some point the work will be extended to create a graphical user interface. Another area of current research is the integration of the SPH interface with a web application with the ability to run the solution on a remote server, the importance of this future implementation is the cost efficiency in having a central server with multiple GPUs and top of the line CPU and RAM memory size that can support the requests of several users around the world.
- **2:** New research on more accurate stereo-vision systems could potentially provided the SPH implementation with more spatial information about the objects under deformation. To be able to make use of this a new constitutive equation for complex solids needs to be implemented.
- **3:** Application of the SPH algorithm to solve Biomechanical problems, at this moment the SPH viscoelastic module is being adapted to solve deformations of the oesophagus.
- **4:** SPH adapted to a multiple GPU implementation, where different kernels can be solved at the same time in different GPUs.
- **5:** New graphic output for problems related to animation and visualisation. The SPH implementation can also be used in the area of visualisation and some work oriented to having a graphical output is being researched by students involved in the project.
- **6:** As mentioned, the limitations of the measurement of the accuracy of SPH for problems in where experimental data is not available will be addressed by the use of general geometric forms. In Sections 6.4.2 and 6.4.1 the mathematical definitions of different shapes were used to measure the behaviour of the SPH under different conditions. This test will be extended with the use of a 3D-printer and experimental measurements to obtain a benchmark of the behaviour under experimental conditions of the fluid on different shapes.

Appendix A

CUDA ARCHITECTURE

The information presented on this appendix is base on the information presented by NVIDIA on the documents [118, 119, 120], the book [121] was use to obtain the description of the different memory types currently available on the CUDA architecture.

At the end of the appendix an example of how to use CUDA to obtain a parallel implementation of the mean filter is introduce, the code is base on a solution implemented by me and use in the IVS lab, note that the solution was modified to serve as an example and some of the components whe implemented

CUDA (Compute Unified Device Architecture) is a parallel computing architecture developed by NVIDIA as a computing engine for the NVIDIA graphics processing units (GPUs), CUDA is used to obtain a parallel solution to computational problems, some of the advantages that CUDA provide are:

- **Heterogeneous serial parallel programming (CPU + GPU):** The CPU (host) and the GPU (device) communicate by memory transfer. This operation can be done at any time during a solution, allowing the CUDA implementations to intermix serial and parallel components.

- **Scalable solution:** The number of cores and threads can be adapted to each specific problem making it possible to scale (within the hardware limits) the number of elements (blocks and threads) to fit the specific problem. When the number of elements have been defined for a specific hardware the same implementation can be used on other models of GPU and CUDA will scale the problem to fit the limits of the new hardware.

- **Code language:** The coding commands and structure are similar to those used in C/C++.

- **Cost efficiency:** The relation between price and computing power decreases with each iteration of the hardware released by NVIDIA, and with the new low level entries such as the GTX 560 Ti the use of parallel systems using GPU solutions has become a more affordable solution.

- **Notebook implementation:** In the new generation of portable computers Nvidia introduced the GeForce GTX 560M, reducing the gap between portable computer and desktop and making it possible to run the SPH implementation without any modifications.

The CUDA solutions run in the host CPU and it is from the host that the calls to the GPU (device) are made, the GPU solutions are called with the use of kernels, the kernels are written in C/C++. One device executes one kernel at a time and the kernel is solved in N threads in parallel. Figure A.1 shows

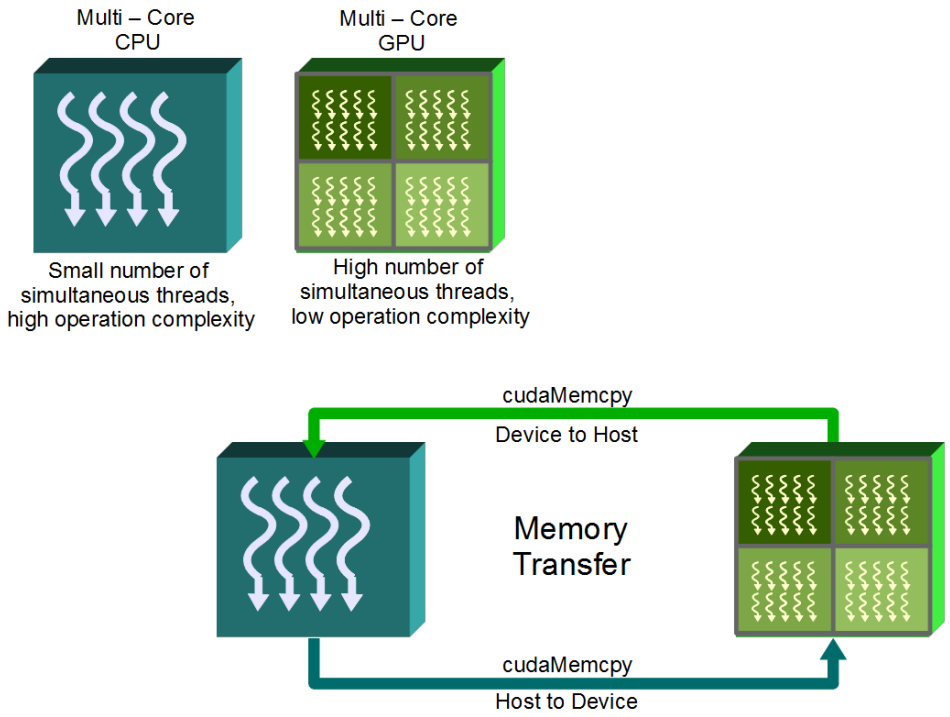


Figure A.1: **Heterogeneous progression of a CUDA implementation** - The Heterogeneous progression of a CUDA implementation allows for mixed solutions between serial and parallel components. The computational cost to the system increases with the number and size of the memory transfer. CPU multithreading allows for solutions that runs the solution concurrently on the CPU and the GPU.

a schematic representation of the code progression with calls between the host and the device and the necessary memory transfers.

Each kernel defines the code that will be solved by all the threads in parallel, each thread solves the same code with different data. Conditional branching is possible but if this occur each branch will be solve in serial. CUDA organise the parallel solution using a grid representing the GPU device, each grid is divided into blocks and each block contains the CUDA threads. In order to make use of the CUDA thread organization the API provides with three three-dimensional vectors variables: threadIdx, blockIdx and blockDim.

Each block contains a number N of threads, this size of the block is define by the user on each dimension. The user also provide with the number of blocks per grid. Each block can contain up to 1024 threads, and in order to executed more threads the kernel can be executed by a number of equally shaped blocks. An NVIDIA GPUs multiprocessor is designed to execute hundreds of threads concurrently using the architecture called SIMT (Single-Instruction, Multiple-Thread). In Figure A.2 is shown the relation between the grid, block and thread organizatin, the grid represents the device been use and the Figure shows the 3D structure of the threads and blocks.

If each one of the individual thread are used to directly read a part of a linear array, the linear index for each one of the three dimensions using the thread and block index are define on Equation A.1 for 2D and Equation A.2 for the 3D:

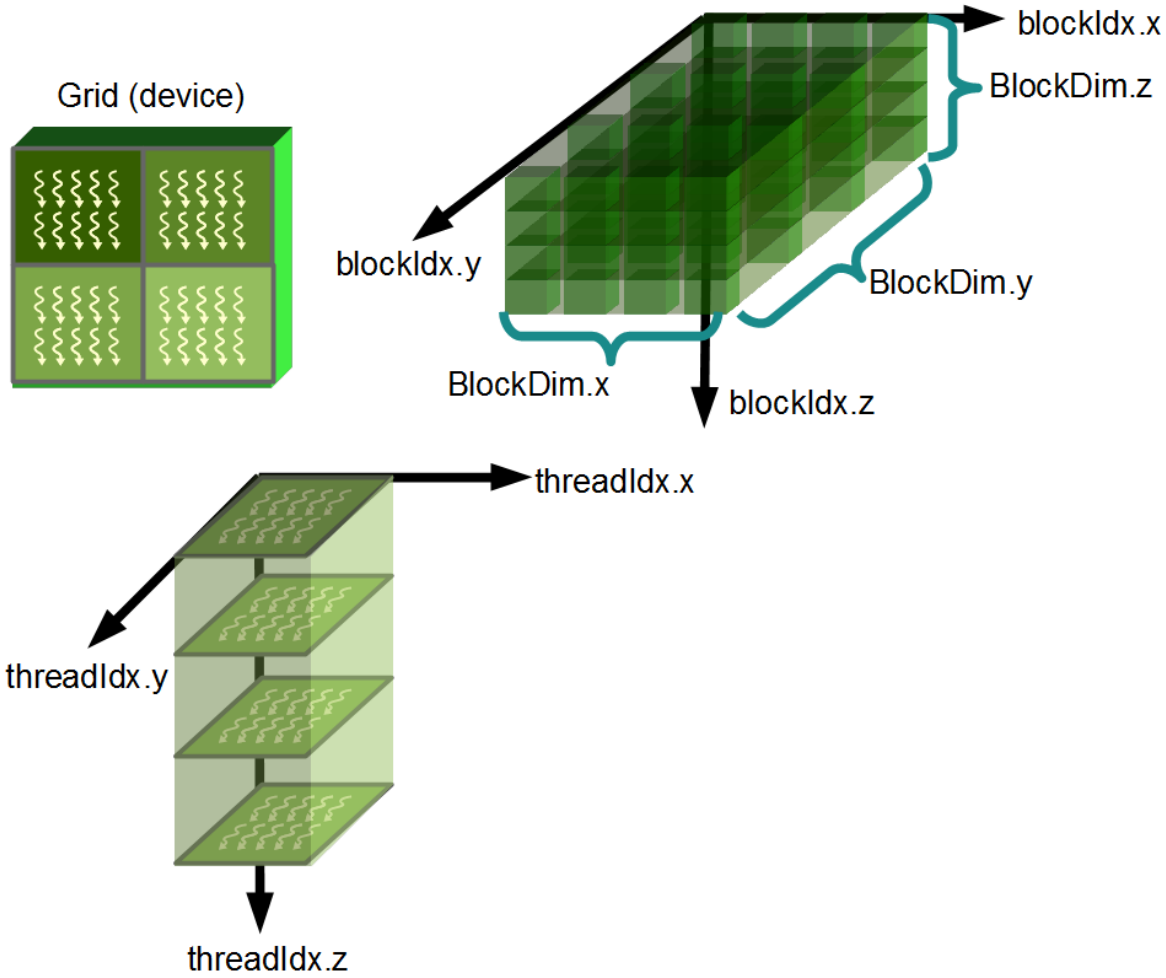


Figure A.2: **CUDA logical structure** - CUDA make uses of the grid, block and threads elements to organized the parallel solution of the kernel.

$$
\begin{aligned}
&unsignedintx = threadIdx.x + (blockIdx.x * blockDim.x)\,;\\
&unsignedinty = threadIdx.y + (blockIdx.y * blockDim.y)\,;\\
&unsignedintw = array\_width;\\
&unsignedintindex = x + (y * w)\,;
\end{aligned}
\tag{A.1}
$$

$$
\begin{aligned}
&\text{unsigned int x} = threadIdx.x + (blockIdx.x * blockDim.x)\,;\\
&\text{unsigned int y} = threadIdx.y + (blockIdx.y * blockDim.y)\,;\\
&\text{unsigned int z} = threadIdx.z + (blockIdx.z * blockDim.z)\,;\\
&\text{unsigned int v} = array\_width;\\
&\text{unsigned int w} = array\_depth;\\
&\text{unsigned int index} = x + (y * v) + (z * v * w)\,;
\end{aligned}
\tag{A.2}
$$

Using a combination of threadIdx, blockIdx and blockDim each individual thread on the kernel can be identify. On Figure is shown an schematic representation of the relations between blocks and threads, note that inside each block the thread index start at 0 A.3.

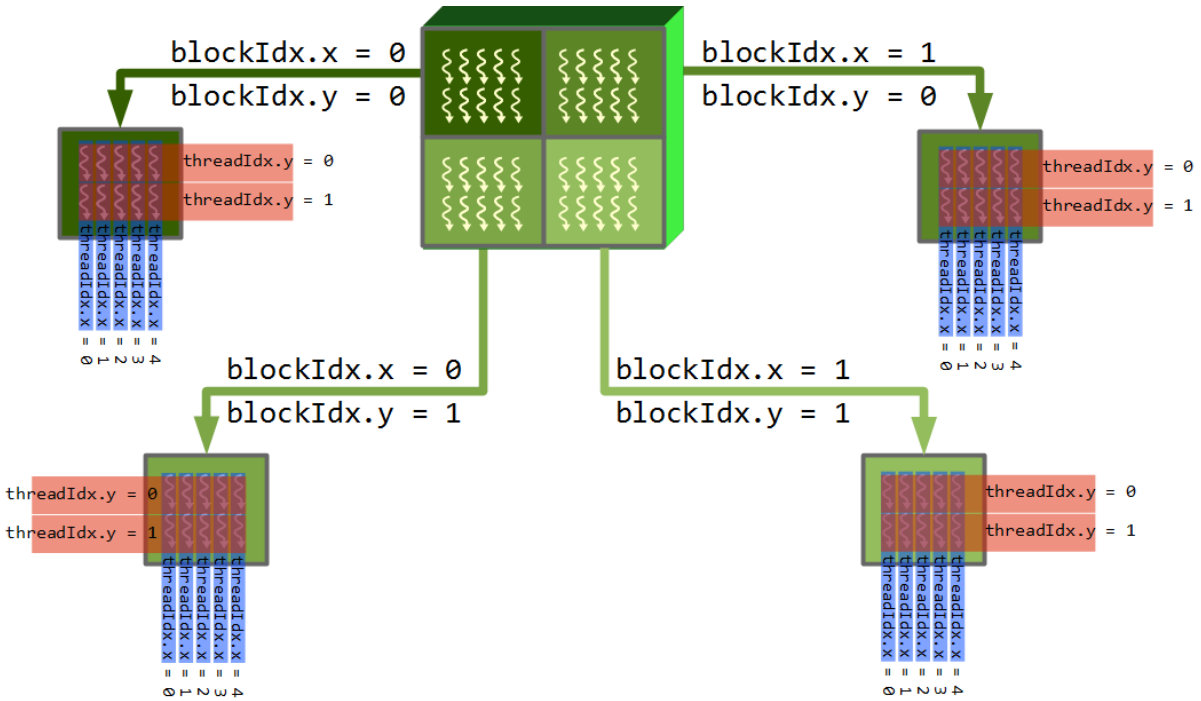


Figure A.3: **CUDA blocks and threads index** - Each block contains a set of threads that will be execute in parallel, the thread index per block starts a 0 and the full individual index of each thread is a combination of the values threadIdx, blockIdx and blockDim .

The kernel functions are solved one at the time in the grid. The operations inside the device can have different computational times for different threads. To force the algorithm to wait for all the treads inside a kernel to be completed before continuing executing the kernel the command __syncthreads() is use.

The blocks cannot synchronise, the reason for this is that blocks can be solved in any order, and more importantly the blocks can be solved either in serial or in parallel. CUDA manages the blocks in this manner so the kernel functions can be scaled across any number of parallel cores(depending on the application and the GPU capabilities). Scalability of the code allows its use in different GPUs with different specifications without needing to modify the code. The serialisation of the blocks will depend on the GPU. Figure A.4 shows an example of two different devices, one with 2 cores and another with 4 cores.

CUDA has a set of memory types that the device has access to. In the book [121] is summarize the different memory access each thread can made in the different memory types:

- Read/write per-thread registers, the register memory is local to each thread and provides a very fast access memory, the registers are use to store the variables declared in the thread.
- Read/write per-thread local memory, local memory resides in the global memory, so is equal in performance to the global memory, CUDA uses local memory when the registers are full.
- Read/write per-block shared memory, if a memory space is available for all the threads that belong to the same block.
- Read/write per-grid global memory, the Global memory component is the one that all the threads have access to and is the slowest of all the available memory.

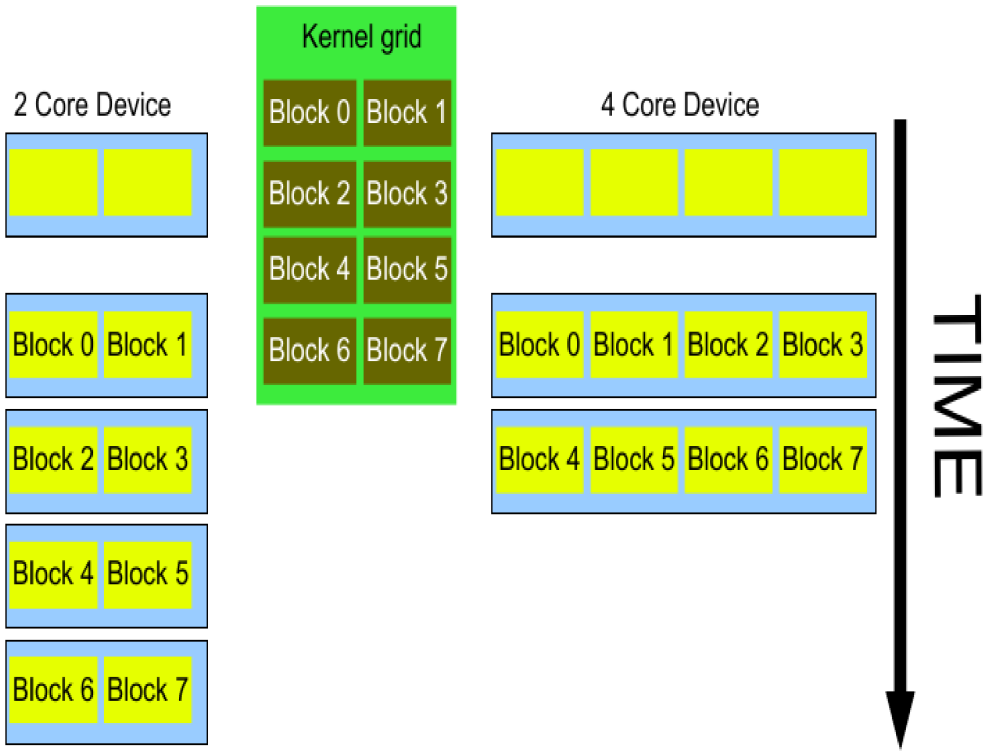


Figure A.4: The implementations on the CUDA architecture will scale to the capabilities of the GPU they are running on. The more parallel cores a system can solve at a time the less computational time will be required for the system to solve the algorithm.

- Read only per-grid constant memory, constant memory is read only from kernels and is hardware optimised for the case when all threads read the same location.

- Read only per-grid texture memory, GPUs use the texture memory to improve the computational time when performing operations such as mapping 2D to 3D or deforming 3D objects.

Using the definitions presented, the memory model, and the id array, a simple numerical example of the different index values per block is shown in Figure A.5.

In the following code examples a basic image filtering algorithm, median filtering, is present to exemplified some of the stages involve in a CUDA solution. The algorithm explanation has been divided into different components in order to show the coding process involve in parallel programming.

After loading the images into the computer memory (Host, RAM memory) the first stage consist in allocating the solution require memory on the GPU and provide with all the constant definitions that will be used on the code Algorithm A.1.

Algorithm A.1: Memory allocation and constant definitions on the GPU.

```
1 //CUDA allows for the declaration of global constant values on the devise, this
     variables will be available to all kernels, and all the thread of the kernels
2 __device__ __constant__ unsigned int w;
3 __device__ __constant__ unsigned int h;
4 __device__ __constant__ unsigned int size;
```

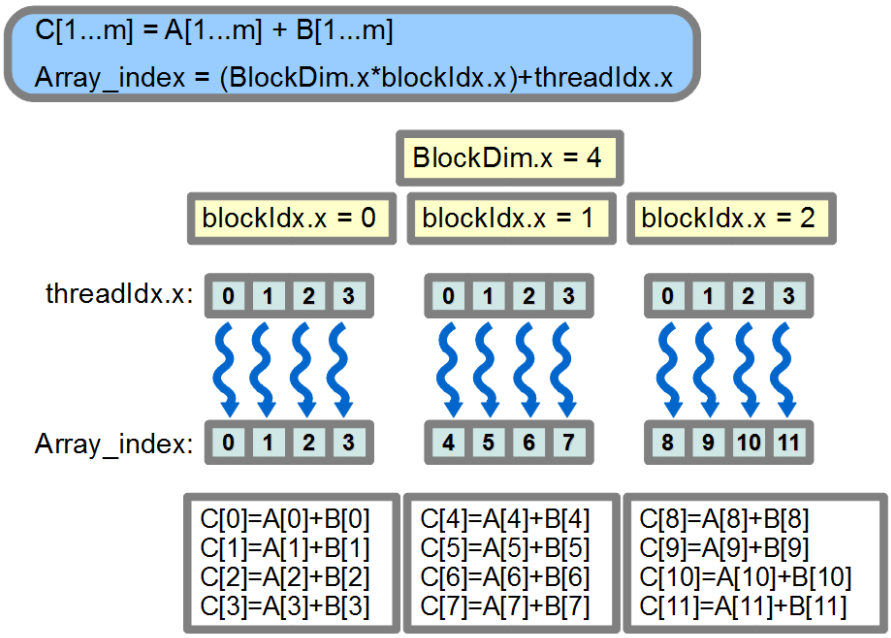


Figure A.5: Numerical example of the index adquisition inside a kernel, the index is use as the method to visit the different components of the array being solved.

```
5  __device__ __constant__ unsigned int size2;
6  __device__ __constant__ unsigned float fsize;
7
8  void Cuda_init(unsigned int img_width, unsigned int img_height, unsigned char*&
       cu_image, unsigned char*& cu_grayImage, unsigned char*& cu_FgrayImage, unsigned
        char*& cu_FRImage, unsigned char*& cu_FGImage, unsigned char*& cu_FBImage,
       unsigned int mean, unsigned int median)
9  {
10   //First the image width and height are define as constants on the GPU, the values
         are obtain from the information provided to the CPU when opening the images
11   cudaMemcpyToSymbol("w", &img_width, sizeof(unsigned int), 0,
         cudaMemcpyHostToDevice);
12   cudaMemcpyToSymbol("h", &img_height, sizeof(unsigned int), 0,
         cudaMemcpyHostToDevice);
13
14   //The next pair of constants are use to define the kernel size of the filters, in
         order to reduce the number of operations, the square of the kernel size is
         also a constant. booth values are defined on the CPU and copy to the GPU.
15   cudaMemcpyToSymbol("size", &kernel_size, sizeof(unsigned int), 0,
         cudaMemcpyHostToDevice);
16   cudaMemcpyToSymbol("size2", &kernel_size2, sizeof(unsigned int), 0,
         cudaMemcpyHostToDevice);
17
18   //since the kernel is solve directly in each pixel position the pixel value is
         weighted before the addition is perform. the variable fsize will contain on
```

```
    the device the weight value.
19  kernel_fsize = 1/kernel_size2;
20  cudaMemcpyToSymbol("fsize", &kernel_fsize, sizeof(float), 0,
        cudaMemcpyHostToDevice);
21
22  //finally the user can modified how many pixels are going to be process per
        thread in the RGB to Gray function, user_weight will defined how many pixels
        on the line the thread will solve.
23  const int num_pix = ceil((float)img_width/(user_weight));
24  cudaMemcpyToSymbol("numpixel", &num_pix, sizeof(unsigned int), 0,
        cudaMemcpyHostToDevice);
25
26
27  //After the constants of the program have been defined, the memory allocation on
        GPU is perform.
28  //First the memory size are define
29  //size_img value contains the total number of pixels in the image
30  int size_Img= img_width * img_height;
31
32  //The images are always converted to RGBA before passing it to the GPU. Reading
        four pixels at the time improve the coalesce memory access on the GPU. The
        memory size for the images is define as:
33  int mem_size_ImgT= sizeof(unsigned char) * size_Img * 4;
34
35  //The image size for the Gray scale image and the filter image is define as:
36  int mem_size_Img= sizeof(unsigned char) * size_Img;
37
38  //After the memory size has been defined the arrays are allocated
39
40  //First the array that will contain the image
41  cudaMalloc((void**) &cu_image, mem_size_ImgT);
42
43  //After that the memory for the gray scale image and the filter image are
        allocated.
44  cudaMalloc((void**) &cu_grayImage, mem_size_Img);
45  cudaMalloc((void**) &cu_FgrayImage, mem_size_Img);
46
47  //Finally allocated the memory for the ouput of each channel
48  cudaMalloc((void**) &cu_FRImage, mem_size_ImgT);
49  cudaMalloc((void**) &cu_FGImage, mem_size_ImgT);
50  cudaMalloc((void**) &cu_FBImage, mem_size_ImgT);
51 }
```

If all the images that are going to be filter have the same size the allocation stage needs to be perform only once. After the memory allocation the image information is transfer from the CPU to the GPU Algorithm A.2.

Algorithm A.2: Memory transfer form the CPU into the GPU is done using the CUDA function cudaMemcpy using the flag cudaMemcpyHostToDevice.

```
1 void Cuda_CPUtoGPU(unsigned char *image, unsigned int img_width, unsigned int
      img_height, unsigned char*& cu_image)
```

```
{
  // copy image from the CPU to the GPU
  int mem_size_ImgT= sizeof(unsigned char) * size_Img * 4;
  cudaMemcpy(cu_image, image, mem_size_ImgT, cudaMemcpyHostToDevice);
}
```

Until this point all the code have been executed on the Host and the CUDA API provides with the tools to allow the Host to open memory, defined constant values and transfer memory to the GPU. After the set-up is finish the first parallel code is define, the CUDA kernel will be use to apply mean filter in the image, Algorithm A.3.

Algorithm A.3: Parallel CUDA implementation of the mean filter.

```
__global__ void RGBAtoGray(unsigned char* cu_image, unsigned char* cu_grayImage,
    unsigned char* cu_FgrayImage, unsigned char* cu_FRImage, unsigned char*
    cu_FGImage, unsigned char* cu_FBImage)
{
  //the variables y and thread are obtained using the CUDA internal thread and
      block index, y is used to keep track of the line number and thread is used to
       know with section of the line is going to be solve by the thred.

  int y = blockIdx.x; //line number
  int thread = threadIdx.x; //initial part of the line that are solve.

  //The filter solution will be implemented on share memory, on CUDA the share
      memory arrays are define using the call __shared__
  __shared__ unsigned char Red[w]; \\w is the width of a line of the image.
  __shared__ unsigned char Green[w];
  __shared__ unsigned char Blue[w];

  //In order to take advantage of the share memory, the image is read in a
      collaborative manner. Each thread is responsible on reading part of the image
      .

  //Fist the variables use to read the data are define.
    //low limit is use to defined the starting position of the array (on the
        current image line)
    int limlow = thread * numpixel;
    //The top limit is use to defined the last pixel that is going to be read by
        this thread.
    int limtop = (thread+1) * numpixel;
    //To help on the array reading the variable ywf defined the starting position
        of the array for this thread (line number times width of the image times
        size of the pixel)
    int ywf = y * w * 4;
    //ifour is use to move in the array four spaces at the time (RGBA)
    int ifour;

  //The loop read the RGBA image into share memory
  for(int i = limlow; i < limtop; i++){
    if(limtop<w){
      ifour = i*4;
```

```
29        Red[i]   = cu_image[ywf + ifour ];
30        Green[i] = cu_image[ywf + ifour + 1];
31        Blue[i]  = cu_image[ywf + ifour + 2];
32
33        //The same loop initialize the arrays that will contain the output result
              from the filter process. These arrays are store in global memory and any
              thread can access any part of them.
34        cu_FRImage[i] = 0.0;
35        cu_GRImage[i] = 0.0;
36        cu_BRImage[i] = 0.0;
37      }
38    }
39    // A __syncthreads is use to ensure that all the threads have read their
          corresponding pixels to memory before continuing
40    __syncthreads();
41
42    //At this point \the three RGB channels are in GPU shared memory (per block
          memory) and operations can be perform on them. It needs to be notice that the
           Gray scale image could also have been created directly at this point in
          share memory but the example will show how to copy from share memory to
          global memory as the final step when obtaining the final Gray scale image.
43    //2d mean filter
44    {
45      //Per thread variable to keep track of the image index.
46      size_t ind_mean;
47
48      //position of the pixel on 3D coordinate system
49      unsigned int psixel_x;
50      unsigned int pixel_y;
51
52
53      //The loop to read the share memory RGBA image is:
54      for(int i = limlow; i < limtop; i++){
55        if(limtop<w){
56          //i contains the pixel linear value position
57          //The x coordinate is:
58            pixel_x = i - (w*y);
59          //the y coordinate is:
60            pixel_y = y;
61
62          //The pixel values of all the channels are read into a thread variable
63          float val_R  =  Red[i];
64          float val_G  =  Green[i];
65          float val_B  =  Blue[i];
66
67          //The following is a branching conditional, two branch will be create and
                they need to solved independently, the first branch will contain the
                pixels inside the limits and the second branch the pixels that are
                outside the limit.
68          if( (pixel_y>(size+1) && pixel_y<(h-(size+1))) && (pixel_x>(size+1) &&
                pixel_x<(w-(size+1)))){
69          //each thread visit the pixels inside the kernel centre on the focus pixel.
```

```
            The value is weighted added
70        for(int i=0; i<size; i++){
71          for(int j=0; j<size; j++){
72            //first the index of the pixel on the image been updated is obtained
73            ind_mean = ((pixel_y+1-j) * w) + (pixel_x+1-i);
74            //the pixels around the kernel are updated.
75            cu_FRImage[ind_mean] = cu_FRImage[ind_mean] + (fsize * val_R);
76            cu_GRImage[ind_mean] = cu_GRImage[ind_mean] + (fsize * val_G);
77            cu_BRImage[ind_mean] = cu_BRImage[ind_mean] + (fsize * val_B);
78          }
79        }
80      }
81    }
82   }
83  }//end of the mean filter
84  //The output arrays from the filter are in global memory, a final operation will
       obtain the Gray scale image resulting of the channels using global memory for
       the operations.
85  //Since the arrays are on global memory the access to each component could be
       done a pixel per thread but this kernel was defined to solved several pixels
       per thread so the same for loop structure is keep.
86  for(int i = limlow; i < limtop; i++){
87    if(limtop<w){
88      cu_grayImage[i] = 0.30 * cu_FRImage[i] + 0.59 * cu_FGImage[i] + 0.11 *
          cu_FBImage[i];
89    }
90  }
91 }
```

After the kernel finish, the solution of the filter is store in the GPU global memory arrays (cu_grayImage, cu_FRImage, cu_FGImage and cu_FBImage). to transfer the results back to the CPU the cudaMemcpy function is use A.4.

Algorithm A.4: Memory transfer form the GPU into the CPU is done using the CUDA function cudaMemcpy and the flag cudaMemcpyDeviceToHost.

```
1  void Cuda_CPUtoGPU(unsigned char& *FRImage, unsigned char& *FGImage, unsigned char&
       *FBImage, unsigned char& *grayImage, unsigned int img_width, unsigned int
     img_height, unsigned char* cu_FRImage, unsigned char* cu_FGImage, unsigned char
     * cu_FBImage, unsigned char* cu_grayImage)
2  {
3    int size_Img= img_width * img_height;
4    int mem_size_Img= sizeof(unsigned char) * size_Img;
5
6    // copy image from the CPU to the GPU
7    cudaMemcpy(FRImage, cu_FRImage, mem_size_Img, cudaMemcpyDeviceToHost);
8    cudaMemcpy(FGImage, cu_FGImage, mem_size_Img, cudaMemcpyDeviceToHost);
9    cudaMemcpy(FBImage, cu_FBImage, mem_size_Img, cudaMemcpyDeviceToHost);
10   cudaMemcpy(grayImage, cu_grayImage, mem_size_Img, cudaMemcpyDeviceToHost);
11 }
```

This is the final stage for the GPU, now the memory of the GPU can be set free.

Algorithm A.5: The GPU memory is free with the use of cudaFree.

```
1 void Cuda_free(unsigned char* cu_image, unsigned char* cu_FRImage, unsigned char*
    cu_FGImage, unsigned char* cu_FBImage, unsigned char* cu_grayImage)
2 {
3   cudaFree(cu_FRImage);
4   cudaFree(cu_FGImage);
5   cudaFree(cu_FBImage);
6   cudaFree(cu_grayImage);
7   cudaFree(cu_image);
8 }
```

After this the result is on the CPU and can be use as an output or further operations.

Bibliography

[1] Nvidia"Nvidia gpu computing documentation", *NVIDIA*. Access date 1/06/2009, Last update 5/11/2001. 2011. [cited at p. 2]

[2] S.Baker, D.Scharstein, J. P.Lewis, S.Roth, M. J.Black and R.Szeliski, "A database and evaluation methodology for optical flow", *11th international conference on computer vision (iccv)*, pp.1-8, 2007. [cited at p. 2]

[3] S. S.Beauchemin and J. L.Barron, "The computation of optical flow", *ACM Comput. Surv.*, vol.27, no.3, pp.433-466, 1995. [cited at p. 2]

[4] J. J.Monaghan and J. C.Lattanzio, "A refined particle method for astrophysical problems", *Astronomy and Astrophysics*, vol.149, pp.135-143, 1985. [cited at p. 5, 22]

[5] G.Liu and M.Liu, "Smoothed particle hydrodynamics : a meshfree particle method", *Singapore : World Scientific*, 2003. [cited at p. 5, 15, 31, 32, 35, 50, 84, 113, 181]

[6] J. J.Monaghan, "Smoothed particle hydrodynamics", *Reports on Progress in Physics*, vol.68, no.8, pp.1703, 2005. [cited at p. 5, 18, 20, 32, 113]

[7] M.Muller, S.Schirm, M.Teschner, B.Heidelberger and M.Gross, "Interaction of fluids with deformable solids: Research articles", *Comput. Animat. Virtual Worlds*, vol.15, no.3-4, pp.159-171, 2004. [cited at p. 5, 15, 35, 181]

[8] M. S.Warren and J. K.Salmon, "A parallel hashed oct-tree n-body algorithm", *supercomputing '93. proceedings*, pp.12-21, 1993. [cited at p. 5, 54, 59, 114, 182]

[9] R. H.Landau, "Computational physics : problem solving with computers", *New York : Wiley, c1997.*, 1997. [cited at p. 6]

[10] J. M.Thijssen, "Computational physics", *New York : Cambridge University Press*, 1999. [cited at p. 6]

[11] D.Harmon, E.Vouga, B.Smith, R.Tamstorf and E.Grinspun, "Asynchronous contact mechanics", *ACM Trans. Graph.*, vol.28, no.3, pp.1-12, 2009. [cited at p. 6]

[12] Y.Lee, D.Terzopoulos and K.Waters"Realistic modeling for facial animation", *ACM.* 1995. [cited at p. 6]

[13] A.Macchietto, V.Zordan and C. R.Shelton, "Momentum control for balance", *ACM Trans. Graph.*, vol.28, no.3, pp.1-8, 2009. [cited at p. 6]

[14] T.Yao-Yang, L.Wen-Chieh, K. B.Cheng, L.Jehee and L.Tong-Yee, "Real-time physics-based 3d biped character animation using an inverted pendulum model", *Visualization and Computer Graphics, IEEE Transactions on*, vol.16, no.2, pp.325-337, 2010. [cited at p. 6]

[15] M.Aschwanden, "Image processing techniques and feature recognition insolar physics", *Solar Physics*, vol.262, no.2, pp.235-275, 2010. [cited at p. 6]

[16] T. R.Shaikh, H.Gao, W. T.Baxter, F. J.Asturias, N.Boisset, A.Leith and J.Frank, "Spider image processing for single-particle reconstruction of biological macromolecules from electron micrographs", *Nat. Protocols*, vol.3, no.12, pp.1941-1974, 2008. [cited at p. 6]

[17] K.Borne, "Scientific data mining in astronomy", *Next Generation of Data Mining*, pp.91-114, 2009. [cited at p. 6]

[18] Z.Jinglun, L.Qiang, J.Guang, S.Quan and X.Min, "Reliability modeling for momentum wheel based on data mining of failure-physics", *knowledge discovery and data mining, 2010. wkdd '10. third international conference on*, pp.115-118, 2010. [cited at p. 6]

[19] X.Ma, C.-H. F.Fung, J.-C.Boileau and H. F.Chau, "Universally composable and customizable post-processing for practical quantum key distribution", *Computers Security*, vol.30, no.4, pp.172-177, 2011. [cited at p. 6]

[20] J.Hua, A. P.Kharam, M. D.Riedel and K. K.Parhi, "A synthesis flow for digital signal processing with biomolecular reactions", *computer-aided design (iccad), 2010 ieee/acm international conference on*, pp.417-424, 2010. [cited at p. 6]

[21] M.Asim Mubeen and K. H.Knuth, "Evidence-based filters for signal detection: Application to evoked brain responses", *ArXiv e-prints*, 2011. [cited at p. 6]

[22] P.David, "Computational physics", *London, New York, J. Wiley*, 1973. [cited at p. 6]

[23] J. B.Gary, "Fortran for scientists and engineers", *El Granada, CA : Scott/Jones*, 1995. [cited at p. 6]

[24] J. D.Anderson, "Basic philosophy of cfd", In J. F.Wendt, editor, *Computational Fluid Dynamics*, pp.3-14, Springer Berlin Heidelberg, 2009. [cited at p. 7]

[25] M.Abbett and G.Moretti, "A time-dependent computational method for blunt body flows", *AIAA*, vol.4, no.12, pp.2136-2141, 1966. [cited at p. 7]

[26] J. D.Anderson, "Governing equations of fluid dynamics", In J. F.Wendt, editor, *Computational Fluid Dynamics*, pp.15-51, Springer Berlin Heidelberg, 2009. [cited at p. 7, 28]

[27] L. D.Landau and E. M.Lifshits, "Fluid mechanics, by l.d. landau and e.m. lifshitz. translated from the russian by j.b. sykes and w.h. reid", *London, Pergamon Press; Reading, Mass., Addison-Wesley Pub. Co.*, 1959. [cited at p. 7]

[28] H.Lomax, T. H.Pullian and D. W.Zingg, "Fundamentals of computational fluid dynamics", *University of Toronto Institute for Aerospace Studies*, 1999. [cited at p. 7]

[29] M. C.Hsu, Y.Bazilevs, V. M.Calo, T. E.Tezduyar and T. J. R.Hughes, "Improving stability of stabilized and multiscale formulations in flow simulations at small time steps", *Computer Methods in Applied Mechanics and Engineering*, vol.199, no.13-16, pp.828-840, 2010. [cited at p. 7]

[30] C.Boutsioukis, C.Gogos, B.Verhaagen, M.Versluis, E.Kastrinakis and L. W. M.Van Der Sluis, "The effect of apical preparation size on irrigant flow in root canals evaluated using an unsteady computational fluid dynamics model", *International Endodontic Journal*, vol.43, no.10, pp.874-881, 2010. [cited at p. 7]

[31] T. E.Tezduyar, K.Takizawa, C.Moorman, S.Wright and J.Christopher, "Spacetime finite element computation of complex fluidstructure interactions", *International Journal for Numerical Methods in Fluids*, vol.64, no.10-12, pp.1201-1218, 2010. [cited at p. 7]

[32] J. T.Oden, T.Belytschko, I.Babuska and T.Hughes., "Research directions in computational mechanics", *A Report of the United States National Committee on Theoretical and Applied Mechanics*, 2000. [cited at p. 7]

[33] W. M.Lai, "Introduction to continuum mechanics", *Oxford : Butterworth-Heinemann*, 1996. [cited at p. 7]

[34] E. H.Dill, "Continuum mechanics : elasticity, plasticity, viscoelasticity", *Boca Raton, FL : CRC/-Taylor and Francis, c2007.*, 2007. [cited at p. 7]

[35] S. H.Crandall, "An introduction to the mechanics of solids", *McGraw-Hill*, 1959. [cited at p. 7]

[36] A.Gastelum, D.P., M.J., W.A., J.J., L.M. and G.G., "3d lip shape sph based evolution using prior 2d dynamic lip features extraction and static 3d lip measurements", In A.Wee, C.Liew and S.Wang, editor, *Visual Speech Recognition: Lip Segmentation and Mapping*, pp.213-238, IGI Global, 2007. [cited at p. 7]

[37] A. B.Shiflet, "Introduction to computational science : modeling and simulation for the sciences", *Princeton, N.J. : Princeton University Press*, 2006. [cited at p. 8]

[38] G. R.Liu, "The finite element method", *Oxford ; Boston : Butterworth-Heinemann*, 2003. [cited at p. 8]

[39] O. C.Zienkiewicz, "The finite element method its basis and fundamentals", *Elsevier Butterworth-Heinemann*, 2005. [cited at p. 10]

[40] S. S.Rao, "The finite element method in engineering", *Amsterdam ; Boston, MA : Elsevier/Butterworth Heinemann*, 2005. [cited at p. 10]

[41] H. S. G.Rao, "Finite element method vs. classical methods", *New Delhi : New Age International P Ltd., Publishers.*, 2007. [cited at p. 10]

[42] O.Oluwole, "Finite element modeling for materials engineers using matlab", *London ; New York : Springer.*, 2011. [cited at p. 10]

[43] G. R.Liu and Y. T.Gu, "Overview of meshfree methods", *An Introduction to Meshfree Methods and Their Programming*, pp.37-53, Springer Netherlands, 2005. [cited at p. 10, 34, 179]

[44] G. R.Liu and D.Karamanlidis, "Mesh free methods: Moving beyond the finite element method", *Applied Mechanics Reviews*, vol.56, no.2, pp.B17-B18, 2003. [cited at p. 11]

[45] G. R.Liu, "Mesh free methods: moving beyond the finite element method", *Boca Raton, Fla. : CRC Press*, 2003. [cited at p. 11, 13]

[46] R. A.Gingold and J. J.Monaghan, "Smoothed particle hydrodynamics - theory and application to non-spherical stars", *Royal Astronomical Society, Monthly Notices*, vol.181, pp.375-389, 1977. [cited at p. 15]

[47] H.Takeda, S. M.Miyama and M.Sekiya, "Numerical simulation of viscous flow by smoothed particle hydrodynamics", *Progress of Theoretical Physics*, vol.92, pp.939, 1994. [cited at p. 15]

[48] M. R.Bate and A.Burkert, "Resolution requirements for smoothed particle hydrodynamics calculations with self-gravity", vol.288, pp.1060-1072, 1997. [cited at p. 15]

[49] P. W.Randles and L. D.Libersky, "Smoothed particle hydrodynamics: Some recent improvements and applications", *Computer Methods in Applied Mechanics and Engineering*, vol.139, no.1-4, pp.375-408, 1996. [cited at p. 15]

[50] M.Becker and M.Teschner, "Weakly compressible sph for free surface flows", *proceedings of the 2007 acm siggraph/eurographics symposium on computer animation*, pp.209-217, 2007. [cited at p. 15, 35]

[51] F.Losasso, G.Irving, E.Guendelman and R.Fedkiw, "Melting and burning solids into liquids and gases", *Visualization and Computer Graphics, IEEE Transactions on*, vol.12, no.3, pp.343-352, 2006. [cited at p. 15]

[52] B.Solenthaler, J.Schlafli and R.Pajarola, "A unified particle model for fluid–solid interactions: Research articles", *Comput. Animat. Virtual Worlds*, vol.18, no.1, pp.69-82, 2007. [cited at p. 15, 35]

[53] R.Keiser, B.Adams, D.Gasser, P.Bazzi, P.Dutre and M.Gross, "A unified lagrangian approach to solid-fluid animation", *point-based graphics, 2005. eurographics/ieee vgtc symposium proceedings*, pp.125-148, 2005. [cited at p. 16]

[54] S. E.Hieber, J. H.Walther and P.Koumoutsakos, "Remeshed smoothed particle hydrodynamics simulation of the mechanical behavior of human organs", *Technol. Health Care*, vol.12, no.4, pp.305-314, 2004. [cited at p. 16, 35]

[55] M. B.Teschner, M.Ihmsen and Matthias, "Corotated sph for deformable solids", *eurographics workshop on natural phenomena*, 2009. [cited at p. 16, 35]

[56] R.Sowerby, J. L.Duncan and E.Chu, "The modelling of sheet metal stampings", *International Journal of Mechanical Sciences*, vol.28, no.7, pp.415-430, 1986. [cited at p. 16, 35]

[57] Y. H.Chan, A.Gastelum Strozzi, A.Lau, R.Gong, P.Delmas and G.Gimel'farb, "Modelling of elastic deformation using stereo vision and smooth particles hydrodynamics", *24th int. image and vision computing new zealand*, pp.1-8, 2009. [cited at p. 16]

[58] K. S.Bhimsen, "Theoretical fluid dynamics", *Wiley*, 1998. [cited at p. 16]

[59] Z. U. A.Warsi, "Fluid dynamics: theoretical and computational approaches", *Taylor and Francis.*, 2006. [cited at p. 16]

[60] K.Tsutomu, "Elementary fluid mechanics", *World Scientific*, 2007. [cited at p. 16]

[61] R. M.James, "Topology", *Prentice Hall*, 2000. [cited at p. 16]

[62] S.Gerald Jay, "Structure and interpretation of classical mechanics", *MIT Press*, 2001. [cited at p. 18]

[63] J. J.Monaghan, "Smoothed particle hydrodynamics", *Annual review of astronomy and astrophysics*, vol.30, pp.543-574, 1992. [cited at p. 19, 22, 35]

[64] M.J.J, "An introduction to sph", *Computer Physics Communications*, vol.48, no.1, pp.89-96, 1988. [cited at p. 20]

[65] P. E.Warnatz, D.Etling, U.Mller, U.Riedel, K.Sreenivasan and J., "Prandtl's essentials of fluid mechanics", *New York : Springer, c2004.*, 2004. [cited at p. 25]

[66] P. K.Kundu, "Fluid mechanics", *Academic Press*, 2002. [cited at p. 25]

[67] T. E.Faber, "Fluid dynamics for physicists", *Cambridge Univ. Press*, 1995. [cited at p. 25, 28]

[68] Y.Nakayama, R. F.Boucher and K.F., "Introduction to fluid mechanics", *Oxford England ; Woburn, Mass. : Butterworth-Heinemann*, 2000. [cited at p. 25, 26]

[69] J.Blazek, "Computational fluid dynamics principles and applications", *Amsterdam ; San Diego : Elsevier, 2005.*, 2005. [cited at p. 26, 31]

[70] E.George, "Analytical fluid dynamics", *Boca Raton : CRC Press*, 2001. [cited at p. 31]

[71] P. W.Randles and L. D.Libersky, "Smoothed particle hydrodynamics: Some recent improvements and applications", *Computer Methods in Applied Mechanics and Engineering*, vol.139, no.1-4, pp.375-408, 1996. [cited at p. 32]

[72] D.Baraff and A.Witkin, "Large steps in cloth simulation", *proceedings of the 25th annual conference on computer graphics and interactive techniques*, pp.43-54, 1998. [cited at p. 35]

[73] F.Losasso, G.Irving, E.Guendelman and R.Fedkiw, "Melting and burning solids into liquids and gases", *IEEE Transactions on Visualization and Computer Graphics*, vol.12, no.3, pp.342352, 2006. [cited at p. 35]

[74] W. M.Lai, "Introduction to continuum mechanics", *Butterworth-Heinemann*, 1996. [cited at p. 36, 41, 42]

[75] F. B.Allan, "Applied mechanics of solids", *CRC Press*, 2010. [cited at p. 37]

[76] I. Z.Tarek, "Introduction to computational micromechanics", *Springer*, 2005. [cited at p. 38]

[77] L.Robert William, "Elasticity", *Prentice-Hall*, 1973. [cited at p. 41]

[78] T. J.Chung, "Applied continuum mechanics", *Cambridge ; New York, NY, USA : Cambridge University Press, 1996.*, 1996. [cited at p. 41]

[79] B. D.Nautiyal, "Introduction to structural analysis / b.d. nautiyal", *New Delhi : New Age International.*, 2001. [cited at p. 41]

[80] G. T. M.Mase and G.E., "Continuum mechanics for engineers", *Boca Raton : CRC Press*, 2010. [cited at p. 41]

[81] A. M.Marc and K. K.Chawla, "Mechanical behavior of materials", *Cambridge University Press*, 2009. [cited at p. 42]

[82] C.Ericson, "Real-time collision detection (the morgan kaufmann series in interactive 3-d technology", *Morgan Kaufmann Publishers Inc.*, 2004. [cited at p. 46]

[83] A.Witkin and D.Baraff, "Physically based modeling: Principles and practice", *Siggraph '97 Course notes*, Siggraph, 1997. [cited at p. 47]

[84] P. A.Tipler, "Physics for scientists and engineers", *W.H. Freeman.*, 2004. [cited at p. 47]

[85] D. E.Knuth, "The art of computer programming", *Addison-Wiley*, 1968. [cited at p. 50]

[86] J.Cheng, P.Finnigan, A.Hathaway, A.Kela and W.Schroeder, "Quadtree/octree meshing with adaptive analysis", *Numerical Grid Generation in Computational Fluid Mechanics*, pp.633-642, 1988. [cited at p. 52]

[87] N.Dale and S. D.Lilly, "Pascal plus data structures", *D. C. Heath and Company*, 1995. [cited at p. 53]

[88] C. A. R.Hoare, "Quicksort", *The Computer Journal*, vol.5, no.1, pp.10-16, 1962. [cited at p. 58]

[89] D.Benson and J.Davis, "Octree textures", *ACM Trans. Graph.*, vol.21, no.3, pp.785-790, 2002. [cited at p. 59]

[90] K.Hegeman, N.Carr and G.Miller, "Particle-based fluid simulation on the gpu computational science iccs 2006", In V.Alexandrov, G.Albadavan et al., editor, pp.228-235, Springer Berlin / Heidelberg, 2006. [cited at p. 60]

[91] D.Madeira, A.Montenegro and T.Lewine"Gpu octrees and optimized search", *Sociedade Brasileira da Computasao*. 2009. [cited at p. 60]

[92] N.Nakasat, "Oct-tree method on gpu: 42/gflops cosmological simulation", *Instrumentation and Methods for Astrophysics*, vol.1, no.1, 2009. [cited at p. 60]

[93] T.Hamada, T.Narumi, R.Yokota, K.Yasuoka, K.Nitadori and M.Taiji"42 tflops hierarchical n body simulations on gpus with applications in both astrophysics and turbulence", *ACM*. 2009. [cited at p. 60]

[94] T. H.Cormen, C. E.Leiserson, R. L.Rivest and C.Stein, "Introduction to algorithms, third edition", *The MIT Press*, 2009. [cited at p. 62]

[95] J.Sanders and K.Edward, "Cuda by example: An introduction to general-purpose gpu programming", *Addison-Wesley Professional*, 2010. [cited at p. 64]

[96] "Nvidia cuda c getting started guide for linux", *NVIDIA Corporation*, 2011. [cited at p. 71]

[97] C.Pozrikidis, "Fluid dynamics theory, computation, and numerical simulation : accompanied by the software library fdlib", *Boston : Kluwer Academic Publishers, c2001.*, 2001. [cited at p. 86]

[98] S.Boy, G.Guennebaud and C.Schlick, "Least squares subdivision surfaces", *Computer Graphics Forum*, vol.29, no.7, pp.2021-2028, 2010. [cited at p. 96]

[99] F.Devernay and O.Faugeras, "Straight lines have to be straight", *Machine Vision and Applications*, vol.13, no.1, pp.14-24, 2001. [cited at p. 102]

[100] B.Prescott and G. F.McLean, "Line-based correction of radial lens distortion", *Graphical Models and Image Processing*, vol.59, no.1, pp.39-47, 1997. [cited at p. 102]

[101] Y.Chan, A.Gastlum, A.Lau, R.Gong, P.Delmas, G.Gimelfarb and J.Marquez, "Modelling of elastic deformation using stereo vision and smoothed particle hydrodynamics", *proceedings of the 24th international conference of image and vision computing new zealand*, pp.43-54, 2009. [cited at p. 102]

[102] J. J.Orteu, T.Cutard, D.Garcia, E.Cailleux and L.Robert, "Application of stereovision to the mechanical characterisation of ceramic refractories reinforced with metallic fibres", *Strain*, vol.43, no.2, pp.96-108, 2007. [cited at p. 103]

[103] M.Mler, S.Schirm, M.Teschner, B.Heidelberger and M.Gross, "Interaction of fluids with deformable solids", *Journal of Computer Animation and Virtual Worlds*, vol.15, no.3-4, pp.159-171, 2004. [cited at p. 114]

[104] N. J.Jarvis, "A review of non-equilibrium water flow and solute transpor in soil macropores: principles, controlling factors and consequences for water quality", *European Journal of Soil Science*, vol.58, pp.523-546, 2007. [cited at p. 117]

[105] H.Flhler, W.Durner and M.Flury, "Lateral solute mixing processes – a key for understanding field-scale transport of water and solutes", *Geoderma*, vol.70, no.2-4, pp.165-183, 1996. [cited at p. 117]

[106] O.Monga, N.Ndeye and J. F.Delerue, "Representing geometric structures in 3d tomography soil images: Application to pore-space modelling", *Computers & Geosciences*, vol.33, no.9, pp.1140-1161, 2007. [cited at p. 117]

[107] G. T.Herman and A.Kuba, "Discrete tomography : foundations, algorithms, and applications", *Birkha*, 1999. [cited at p. 117]

[108] H.Jiang, "Computed tomography principles, design, artifacts, and recent advances", *SPIE*, 2009. [cited at p. 117]

[109] C.Duwig, P.Delmas, K.Mller, B.Prado, J.Etchevers, H.Morin and K.Ren, "Quantifying fluorescent tracer distribution in allophanic soils to image solute transport", *Eur. J. Soil Sci.*, vol.59, pp.94-102, 2008. [cited at p. 118]

[110] B.Prado, C.Duwig, C.Hidalgo, D.Gomez, H.Yee, C.Prat, M.Esteves and J.Etchevers, "Characterization, functioning and classification of two volcanic soil profiles under different land uses in central mxico", *Geoderma*, vol.139, pp.300-313, 2007. [cited at p. 118]

[111] R.Wooding, "Steady infiltration from a shallow circular pond", *Water Resour. Res.*, vol.4, pp.1259-1273, 1968. [cited at p. 119]

[112] W. R.Gardner, "Some steady-state solutions of unsaturated moisture flow equations with application to evaporation from a water table", *Soil Sci.*, vol.85, pp.228-232, 1958. [cited at p. 119]

[113] R.Deriche, "Using canny's criteria to derive a recursively implemented optimal edge detector", *International Journal of Computer Vision*, vol.1, no.2, pp.167-187, 1987. [cited at p. 123]

[114] S.RoscoatRolland du, M.Decain, X.Thibault, C.Geindreau and J. F.Bloch, "Estimation of microstructural properties from synchrotron x-ray microtomography and determination of the rev in paper materials", *Acta Materialia*, vol.55, no.8, pp.2841-2850, 2007. [cited at p. 123]

[115] W. E.Lorensen and H. E.Cline, "Marching cubes: A high resolution 3d surface construction algorithm", *SIGGRAPH Comput. Graph.*, vol.21, no.4, pp.163-169, 1987. [cited at p. 126]

[116] F.Leymarie and L.M.D., "Simulating the grassfire transform using an active contour model", *IEEE Pattern Anal*, vol.14, pp.56-75, 1992. [cited at p. 128]

[117] Y. S.Schultz and M.H., "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM Journal on Scientific and Statistical Computing*, vol.7, no.3, pp.856-869, 1986. [cited at p. 145]

[118] Nvidia"Cuda c best practices guide", *NVIDIA*. Last update May. 2011. [cited at p. 187]

[119] Nvidia"Nvidia cuda c programming guide version 4.0", *NVIDIA*. Last update June. 2011. [cited at p. 187]

[120] Nvidia"Cuda api reference manual version 4.0", *NVIDIA*. Last update February. 2011. [cited at p. 187]

[121] D.Kirk and W.Hwu, "Programming massively parallel processors: A hands-on approach", *Morgan Kaufmann Publishers*, 2010. [cited at p. 187, 190]