Parallel Lagrangian Particle Transport. Application to Respiratory System Airways

PhD Thesis

Edgar Olivares Mañas

Barcelona Supercomputer Center (BSC-CNS)

Facultat de Matemàtiques i Estadísitca (FME)
Universitat Politècnica de Catalunya (UPC)

Directed by Dr. Guillaume Houzeaux. September 2018

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

Parallel Lagrangian Particle Transport. Application to Respiratory System Airways

PhD Thesis

Computer Applications in Science & Engineering (CASE) department

2018

Versión 1.0+

Barcelona Supercomputer Center (BSC-CNS)

Facultat de Matemàtiques i Estadísitca (FME)

Universitat Politècnica de Catalunya (UPC)

Directed by Dr. Guillaume Houzeaux. September 2018

A la meva iaia,

Agraïments

You know what my philosophy of life is?
That it's important to have some laughs,
but you gotta suffer a little too, because
otherwise you miss the whole point to
life.

W. Allen

Durant els 4 anys (més propina) que ha durat aquesta tesi hi ha hagut moments per a tot: per plorar i per riure, per gaudir i per trobar forces per no abandonar. Però durant aquest temps, sobretot, ha passat molta gent per la meva vida. Sense ells i elles, aquesta tesis haguès sigut impossible.

Alguns ja hi éren, altres han arribat per quedar-se. D'altres, han passat de visita, més curta o més llarga, sempre deixant records perdurables i, malauradament, alguns han marxat per sempre: a la meva àvia, que no entenia ben bé el que feia però quan intentava explicar-li de manera senzilla que estudiava el sistema respiratori amb l'ajuda d'un superordinador per simular-lo, sempre em responia: "cuando seas doctor, me curarás de mis problemas respiratorios". No sé què entenia ben bé, i tot que jo sé que com a físic no l'hagués pogut curar mai, m'hagués agradat arribar a temps perquè m'hagués pogut veure ser doctor.

Per a la família, per ma germana i en especial la meva estimada mare i el meu pare (amb qui tants partits del Barça hem compartit) i que des de petit m'han acompanyat a cada pas.

Pels amics, els de tota la vida d'Horta, Héctor i Bernat sobretot, però també la resta, que malauradament ara veig tan poc, d'ençà que vaig deixar el pis de Gràcia on vivia en començar aquesta tesi.

Als de la uni, Laia, Marc i Marina (que han tingut un fill preciós, en Martí) a qui en general segueixo veient, tot i que alguns, la Lídia o el Bartrés, viuen a massa quilòmetres per fer-ho tan sovint com abans.

Parlant de quilòmetres, a tots els que n'han compartit algun corrent junts. I aquí són massa gent per citar-los a tots. Als del club d'atletisme Serra Marina d'Alella. Al grup de Begues, per posar-los un sobrenom: Kike, Joan, Miguel, Josep i Andrea (uns altres que acaben d'estrenar paternitat),

Rubén o Sílvia. A l'Oriol, que tan bona amistat vam fer durant la meva estada a Anglaterra. I a la Clàudia, que mantenim una afició comuna pels esports i la muntanya i una gran amistat que es va forjar ja fa anys a El Salvador. Em sap greu pels que no em vau poder seguir el ritme i pels pocs que vau córrer més que jo, no us relaxeu que encara sóc a temps d'atrapar-vos.

Per a tots els companys i companyes de pis que he tingut, com el Nacho, l'Aberto, l'Estel i la Georgina, que durant un temps va ser alguna cosa més que una companya de pis.

A l'Anna Maria, que viu massa enfeinada i no tan lluny en realitat (a Arbeca) per veure'ns tant com voldríem.

Òbviament, als companys de feina del BSC. Al Cristo, que es va fer amic meu quan li vaig dir que era d'esquerres i ja només ens ha pogut separar un postdoc a l'estranger. Quins bon moments compartint cafès, "fruit time" i congressos a l'estranger que sempre aprofitàvem per agafar-nos més dies i viatjar. Als argentins Matías (el cuerdo) i l'Alfonso (no tan cuerdo), a la gallega Mariña i l'andalusa Maria, al Georgios que no és massa xerraire però sempre ha estat al costat, a l'escalador d'en Jordi i a l'Àlex. També als companys de dinar del divendres, Dani, Xavi, Raúl i Hadrien (un respi més). I a la Paula, que ja la coneixia de Física, però que ha sigut al BSC on de veritat hem creat una indestructible amistat.

Al Guillaume, el meu director que m'ha intentant guiar com ha pogut durant aquests 4 anys d'anades i tornades.

Finalment a la Laura, que era la meva parella en començar la tesi, just quan ella va marxar a Londres i en pocs mesos la distància va fer l'inevitable. I a la Mireia, amb qui tinc la sort de compartir la meva vida en el moment d'acabar-la i espero que durant molt de temps més.

Abstract

We are not to tell nature what she's gotta be. She's always got better imagination than we have.

R. Feynmaan

This thesis is focused on particle transport in the context of high computing performance (HPC) in its widest range, from the numerical modeling to the physics involved, including its parallelization and post-process. The main goal is to obtain a general framework that enables understanding all the requirements and characteristics of particle transport using the Lagrangian frame of reference.

Although the idea is to provide a suitable model for any engineering application that involves particle transport simulation, this thesis uses the respiratory system framework. This means that all the simulations are focused on this topic, including the benchmarks for testing, verifying and optimizing the results. Other applications, such as combustion, ocean residuals, automotive or aeronautics have also been simulated by other researchers using the same numerical model proposed here. However, they have not been included here in the interest of allowing the project to advance in a specific direction, and facilitate the structure and comprehension of this work.

Human airways and respiratory system simulations are of special interest for medical purposes. Indeed, human airways can be significantly different in every individual. This complicates the study of drug delivery efficiency, deposition of polluted particles, etc., using classic in-vivo or in-vitro techniques. In other words, flow and deposition results may vary depending on the geometry of the patient and simulations allow customized studies using specific geometries. With the help of the new computational techniques, in the near future it may be possible to optimize nasal drugs delivery, surgery or other medical studies for each individual patient though a more personalized medicine.

In summary, this thesis prioritizes numerical modeling, wide usability, performance, parallelization, and the study of the physics that affects particle

transport. In addition, the simulation of the respiratory system should carry out interesting biological and medical results. However, the interpretation of these results will be only done from a pure numerical point of view.

Achievments

Action is the foundational key to all success.

P. Picasso

Gambaruto, A., Olivares, E., Calmet, H., Houzeaux, G., Bates, A. and Doorly, D. Transport and deposition in the upper human airways during a sniff. COMPSAFE2014. Pages 13–16. Sendai, Japan. 2014.

Olivares, E. Cajas, J.C. and G. Houzeaux. A Lagrangian Particle Transportation Solver Using a Coupled Software. CMN2015, Lisboa, Portugal. 29 June–2 July 2015.

Houzeaux, G., Garcia-Gasulla, M., Cajas, J., Artiguesa, A., Olivares, E., Labarta, J. and Vázquez, M. Dynamic load balance applied to particle transport in fluids. International Journal of Computational Dynamics, vol. 30(6), pages 408–418, 2016.

Olivares, E. and Houzeaux, G. Time Integration Schemes Comparative for Particles Transport. ECCOMAS. Creta, Greece. 2016

Olivares, E., Calmet, H. and Houzeaux, G. Nanoparticle Deposition In Nasal Cavity Airways and the importance of a good implementation of Brownian diffusivity. CMBE. Pittsburg, USA. 2017.

Calmet, H., Kleinstreuer, C., Houzeaux, G., Kolanjiyil, A., Lehmkuhl, O., Olivares, E. and Vázquez, M. Subject-variability effects on micron particle deposition in human nasal cavities. Journal of Aerosol Science, vol. 115, pages 12–28, 2018.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The average introduction to almost any book is somewhat of a bore.

B. Karloff

Summary: The main motivations of this thesis may be split in two topics. First, its medical application of particle transport simulations in respiratory system airways. Second, the numerical challenges to build a robust code capable of simulating particle transport (section 1.1).

Most of the particle transport applications are considered computationally intensive tasks. This thesis has been carried out in a supercomputation center, guaranteeing access to high performance computing (section 1.3). However, the capacity to access this computational muscle does not exempt this work from the need to adopt some compromises as far as respiratory system simulation is a complex multi-physics problem (section 1.4). Therefore, the goals of this thesis are limited to simulating the part of the respiratory system that corresponds to particle transport (section 1.5).

1.1 Motivation

Particle transport simulations require the use of computers to convert numerical models into a code capable of describing particle transport behavior. Particle transport modelling can involve a large amount of different applications, such as icing on aircraft (Myers, 2001), combustion (Dziugys y Peters, 2001), ash propagation (Folch et al., 2009), car engines, respiratory system (Bates et al., 2014; Inthavong et al., 2011) or particle fusion in a Tokamak (Sovinec et al., 2004), among others.

The external field in charge of transportation can vary in every application. Some of the typical fields are:

- Electromagnetic (EM) field.
- Gravitational field.
- Fluid flow. Here it can be the air flow, water...

For example, in fusion application, the transport is due to the EM field through the Lorentz force. However, in other applications (e.g., icing, ash propagation or respiratory system simulations) the transport is due to the fluid flow, mainly because of the drag force. But even with the same external field such as a fluid flow and similar agents in charge of transport (e.g., involved forces), a different physical model may be required, depending on the application, for taking into account possible events (e.g., collisions, phase transitions, etc.). By way of example, on the one hand, combustion may require interaction between particles and particle–fluid two–way coupling, as a particle momentum may affect the flow. On the other hand, an icing application may require a model of the transition phase from liquid to solid. Finally, in the respiratory system any of these processes are neglectable, although while studying the physics in charge of deposition they may also become fundamental.

In many of the aforesaid examples, this thesis focuses on the respiratory system. There is a large amount of medical applications in this area. Particle transport simulation can be used to study drug delivery, tumor fighting or surgery (after modeling the optimal geometry for the patient), among others. These multiple applications make this topic something particularly cheerful to study.

Airflow is the main external field in charge of transport in the respiratory system airways. Therefore, being able to properly model the air behavior is fundamental before taking on the particle transport problem. Moreover, a deep analysis of the physics involving particle deposition is required after studying particle transport in the respiratory system.

For drug delivery, inhaled dust or sniffed drugs, the diameter of the particles may vary from micrometers to nanometers. In this range of values, the most dominating physics may also vary. Therefore, this thesis has also performed the study on the most important physics concerning microparticles and nanoparticles transport. In addition, reproducing its complicated geometries also becomes a challenge. In this case, human respiratory system airways may vary in every person and a good replication of each-individual geometry is essential for properly studying any issue of a specific patient.

Modelling the respiratory system is a computationally intensive task, and for this reason, high computing capacity is required.

Respiratory system simulations require High Performance Computing (HPC). HPC simulations may open a new window to this field, where a better comprehension of respiratory system behavior, or even a path to personalized medicine, can become a reality.

This thesis has been carried out in the Barcelona Supercomputing Center (BSC-CNS) which hosts MareNostrum (MN), one of the most powerful supercomputer in Europe at the time of writing this thesis. This has guaranteed accessing to HPC.

The algorithm presented in this thesis has been implemented in Alya. Alya software is a multi-physics code developed and maintained at BSC-CNS that is in charge of simulating particle transport, because it is an optimized code for supercomputers, such as MN but also for any other supercomputers. In fact, it has also been feasible to use Alya in other supercomputers, such as Blue Waters (Illinois) or Occigen (CINES, Montpellier), where particle transport was computed.

Finally, this thesis can be split in two main fields. On the one hand, there is the physics involving particle transport and how to process it in a HPC context. On the other hand, there are the applications focusing on current medical research on the respiratory system.

1.2 Computational, numerical and medical challenges of the thesis

Of great potential is improving the actual medicine techniques by including computational simulations. In the particular case of the respiratory system, the drug-aerosol delivery is considered as a real option for vaccines, insulin and medication. Herein, using Computational Fluid-Particle Dynamics (CF-PD), has shown many advantages including repeatability and regional deposition resolution (Kolanjiyil y Kleinstreuer, 2016). In order to simulate drug-aerosol delivery, particle transport equations must be solved. For this purpose, modelling particle transport requires taking the following steps. First, a deep study of the physics affecting particles must be carried out. Next, determining how to solve the physics equations over time is needed, which implies the requirement of an integration scheme. Finally, this integration scheme suitable to HPC must be implemented.

1.2.1 Medical applications of respiratory system simulations

Nowadays, *in-vivo* or *in-vitro* experimentation of the respiratory system is still a challenge because of the complexity and variability of the specific geometries in each patient. Recent techniques such as Magnetic Resonance Imaging (MRI) and Computer Tomography (CT) have helped in reconstructing physiologically realistic models.

Simultaneously, the advancements in computer hardware and simulation software technology to obtain detailed, accurate and realistic visualization of the flow field and particle transport and deposition have opened a new way for studying such subject-specific geometries (Kleinstreuer et al., 2014).

Using nasal drug delivery (introducing medicine via nasal spray) has become one of the preferred medical options due to its advantages. As (Djupesland, 2013) says, *nasal delivery is the logical choice for topical treatment of local diseases in the nose and paranasal sinuses such as allergic and non-allergic rhinitis and sinusitis.* The pulmonary route for direct drug-aerosol delivery is also an engaging approach to combat brain or lung diseases or to reach systemic regions (Kleinstreuer et al., 2014). Intranasal direct drug delivery is being considered as a preference to deliver vaccines, insulin, and medication for treating various diseases and disorders affecting the central nervous system (Illum, 2003).

A more detailed explanation of the medical applications of particle transport is given in chapter 2.

1.2.2 Physics affecting particles

In the respiratory system, particles are transported by the air, mainly because of the drag force, but many other different forces may affect particles. These forces may or may not depend on the fluid. But only the dominant ones should be taken into account among the physics and the large number of forces existing in nature. For example, apart from drag force, another typical fluid-dependent force is buoyancy (it depends on the fluid density). However, buoyancy also depends on a fluid-independent force like gravity. If the particle is light enough to neglect the gravity effects and the difference of density between air and particle are small enough, the buoyancy will be negligible too. Other fluid-dependent forces may also become important in function of the size of the particle. Brownian diffusion can become absolutely necessary if the particles are small enough. But lift force will only be necessary for larger particles, instead (Li y Ahmadi, 1992). Deciding which forces are necessary to calculate, or which ones to neglect, is an interesting topic that requires a high comprehension of the literature.

1.2.3 Numerical method

To solve the fluid in a certain geometry, this must be discretized. In the case of this thesis, the Finite Elements Method FEM was used. Likewise, particles are tracked in the space individually, by using a Lagrangian frame of reference and a time integration scheme. General frameworks are available for the construction of time-step integration algorithms applied to particle transport in the particular case of time discretization methods. Some examples are Runge–Kutta methods, weighted residual methods, Taylor series collocation

methods, Hamilton's principle, Hamilton's law, or least–squares methods (Fung, 2003). Every method has its pros and cons, frequently not being feasible for a large range of applications. Hence, the difficulty of the choice can be found in balancing its versatility and convergence behavior, without losing accuracy. In this thesis, the particle transport in respiratory system application is solved using a semi-implicit Newmark-β with an inner Newton-Rapshon (NR), as proposed in chapter 6.

1.3 High performance computing environment

This thesis was developed in a supercomputer center, Barcelona Supercomputer Center BSC-CNS. Here, the department of Computer Applications in Science & Engineering (CASE) aims at applying computational models to science and engineering by means of HPC.

BSC-CNS hosts a supercomputer called MareNostrum MN, shown in figure 1.1. Supercomputers are an extremely useful tool for complex simulations, such as biomechanics or engineering devices. In this context, CASE department is in charge of simulating the physics affecting biological or engineering processes.

Figure 1.1: Picture of the MareNostrum (MN) supercomputer, hosted inside a chapel.

For this purpose, a multi-physics and multi-scale software optimized for HPC, called Alya, was initially created in 2005. Nowadays, Alya is still being constantly developed and growing. Among the physics solved by Alya it can be mentioned: Incompressible/compressible flow, non-linear solid mechanics, chemistry, particle transport, heat transfer, turbulence modelling, electrical propagation, radiation, etc.

1.3.1 Hardware: MareNostrum and other supercomputers

A supercomputer is a computer with a high level of computing performance compared to a general-purpose computer (e.g., a laptop, a desktop computer or even compared to a regular cluster). The performance of a

supercomputer is measured in floating-point operations per second FLOPS. This measurement makes it easy to compare the power of a computer. As will be shown next, a supercomputer like MN has a performance of about petaFLOPS, whereas a regular laptop has about gigaFLOPS, and some of the most powerful desktop computers, teraFLOPS.

Supercomputers play an important role in the field of computational science (a multidisciplinary field that uses advanced computing capabilities for understanding and solving complex problems), and are used to solve computationally intensive tasks in a wide range of fields in physics and engineering. The first supercomputers are considered to have appeared in the 1960s with the Atlas at the University of Manchester, the IBM 7030 Stretch, and a series of computers at Control Data Corporation (CDC) designed by Seymour Cray.

In the case of MN, the first was built in 2004. This first supercomputer was called MareNostrum I (MN I) and its computing capacity was 42.35 Teraflops. In 2006 its capacity was increased to 94.21 Teraflops (from 4.812 processors to 10.240) and it was called MN II. Next, with the MN III upgrade during 2012-2013, it achieved a peak performance of 1.1 Petaflops. Finally, at the end of June of 2017 began operating MN IV (shown on figure 1.2) reaching a peak performance of 13.7 Petaflops. At the time of writing of this thesis, it was the most powerful supercomputer in Spain and third one in Europe.

Figure 1.2: Lastest upgrade of MareNostrum is called MareNostrum IV.

In total, MN has 384.75 Terabytes (TB) of main memory and 3456 nodes divided in:

- 2x Intel Xeon Platinum 8160 24C at 2.1 GHz,
- 216 nodes with 12x32 GB DDR4-2667 DIMMS (8GB/core),
- 3240 nodes with 12x8 GB DDR4-2667 DIMMS (2GB/core).

Each node has a certain number of cores. The communication between cores in the same node is faster than between different nodes, but the capacity of

a node is delimited by its Random-Access Memory (RAM). Communication within nodes is done within next interconnection networks:

- 100Gb Intel Omni-Path Full-Fat Tree,
- 10Gb Ethernet.

Finally, the MN operating system is SUSE Linux Enterprise Server 12 SP2.

1.3.2 Software: Alya, a multi-physics code optimized for high performance computing

Capable of working in MN and also in any other distributed memory supercomputer, Alya emerges as a multi-physics code capable of distributing the computation in a hybrid way.

Hybrid parallelism means that Alya can distribute the work using distributed or shared memory at the same time:

- *Distributed memory:* The message passing interface (MPI) library distributes computation between different nodes or cores. The computation division by MPI is called a MPI process.
- *Shared memory:* At the node level, threads using OpenMP or tasks using OmpSs, can also be generated to achieve shared memory parallelism.

Even so, at the node level Dynamic load balance (DLB) techniques have also been introduced to take advantage of computational resources. Accelerators like GPU can also be included to further enhance the performance of the code.

The Alya structure is split into different modules (which are shown in figure 1.3) and a kernel. The kernel contains the common functions for any module, whereas each module contains the specific equations to solve its particular physics. The PARTIS module has been specifically developed for particle transport simulations, and is aimed to be suitable for any application that may need particle transport, as mentioned above. Depending on the type of simulation, other modules will be required.

During this thesis, the PARTIS module became the principal actor. Since the time of writing this thesis, although many applications may require this module, Partis has mainly been focused on simulating human respiratory system airways. Currently, simulating it typically requires two different modules from Alya: The PARTIS and NASTIN. PARTIS module, for solving particle transport equations. The NASTIN module solves an incompressible flow, in this case the air.

Figure 1.3: Alya modules. This thesis is focused on the PARTIS module.

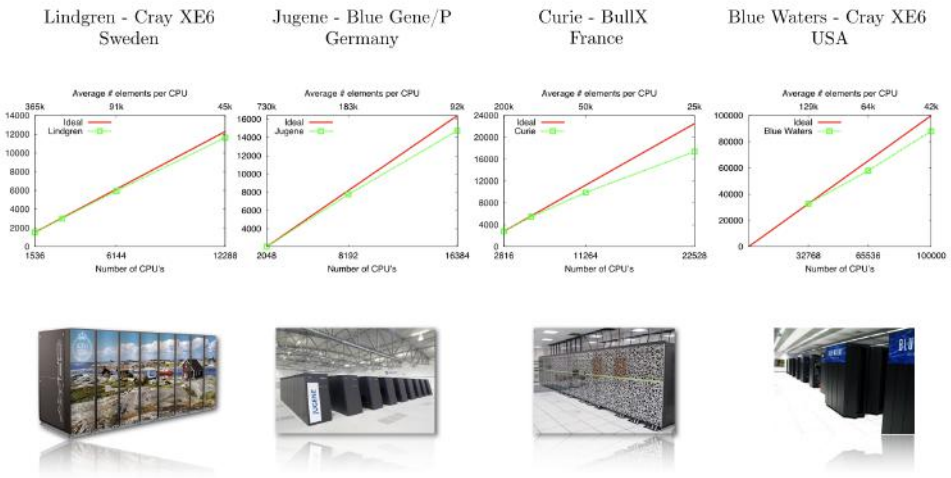


Figure 1.4: Alya strong scalability in different tested supercomputers.

1.3.3 High Performance Computing context in biological simulations

Biological systems, and in particular, physiological systems are very complex with a very hierarchical structure that presents complexity at every level, from organs to cells. Numerical models are key for a better description and comprehension of these systems. Even though numerical models can simulate specific physiological systems (e.g, the cardiovascular system, respiratory system, etc.), we are far from being able to describe it as a whole and with precision. Physics and biological behaviors are hierarchical and only the dominant processes should be taken into account. Thus, smaller or bigger compromises are assumed, depending on the computational cost and accessibility to computational power.

The respiratory system is not an exception in complexity. Many of the simulations require an elevated computational cost, where an HPC environment becomes necessary to minimize the number of compromises. In the particular case of this thesis, the main sources of an elevated computational cost are:

- The level of resolution of the geometry to compute turbulent scales.
- The mathematical model in charge of solving the fluid.
- The number of particles involved.
- The number of time steps required to solve the simulation.

Different ways of coupling. Another potential problem is that the air velocity and pressure must be solved before starting to solve the physics of particle transport, which may depend on these variables. Hence, air and particle transport models need to be coupled, by first solving the air and next the transport. There are three typical solutions for coupling fluid and particles:

- Writing the solution of the fluid on the disk and then using this solution to solve particles. This method is extremely slow and unaffordable for large meshes and lots of time steps.
- Single code: Solves the fluid equation for time $n + 1$; and then transports the particles from n to $n + 1$ using the fluid velocities obtained at n and $n + 1$
- Multi-code: The fluid and particle equations are solved using two different codes or instances of the same code, one dedicated to the fluid and the other to the particles. The coupling is then realized once the velocities of the fluid are obtained at time $n + 1$, using MPI as a message passing interface between the fluid and particle processes. Once the MPI messages have been sent, the fluid process can proceed to the next time step, thus enabling overlap of the computation of the fluid at $n + 2$ and particle transport between n and $n + 1$.

This thesis is mainly focused on the remaining two items approach.

Another issue that this thesis addresses in terms of computational efficiency is that fluid and particle ideal discretization, or even parallelization, are extremely different. Thus, the coupling of the air (NASTIN module in Alya) and particles (PARTIS module in Alya), can run into some of the difficulties when:

- *Transferring values from NASTIN to PARTIS:* The fluid variables (e.g., fluid velocity) are solved in the computational domain, which has to

be discretized. At the same time, particles require some of these fluid variables, but they are located at a certain point of the computational domain where a numerical method must be adopted. In the case of this thesis, the Finite Element Method (FEM) enables working from the discrete space to the approximate results in the continuum, in order to obtain the variable value in the exact location of a given particle.

- *Communicating between each module:* At a higher level than the previous item, the three above solutions were presented for sending the fluid variables of particle calculation (i.e., writing a solution in the disk, using a synchronous mono-code approach, or an asynchronous/synchronous multi–code one). The most suitable option may depend on the computational cost and the amount of data stored of each simulation.

- *Balancing:* However, as fluid occupies the entire computational domain, its calculation can be split into subdomains, while particles can be anisotropically distributed across the domain (Houzeaux et al., 2016).

1.4 Hypothesis and limitations of the model

Computing hierarchical and complex systems like biological models require a specific scope, which delimits the area or physics that must be modelled. This implies adopting compromises, limitations and hypothesis that enable removing the most negligible physics affecting our system. Otherwise, the computational cost would increase dramatically until becoming an unaffordable simulation.

The main hypothesis adopted in this thesis, together with the limitations that each hypothesis implies, are:

Biological hypothesis. Biological systems are very complex and taking into account all the processes involved may become unaffordable. For this reason, some approximations of its behaviour are considered.

- *Hypothesis:* The geometry is rigid. Thus, the mesh forming the domain (or the geometry) is static and fixed.

  Limitation: During a respiration cycle the system contracts and expands, this is not taken into account because of the rigid geometry hypothesis (Kleinstreuer y Zhang, 2010).

- *Hypothesis:* No mucous layer is considered near walls.

  Limitation: Adding a mucous layer changes the properties of the wall or its shape, which could affect deposition statistics in the nasal vestibule.

- *Hypothesis:* The temperature of the flow remains constant.

  Limitation: The temperature of the flow may have an effect on its behavior or on brownian diffusion (higher temperature means higher diffusion). However, inside the respiratory system air temperature typically remains very stable in a healthy patient even under extreme conditions (McCutchan y Taylor, 1951).

Numerical hypothesis. Numerical approaches can find its own limits as an approximation of the reality. In simulations, equations are discretized in time and space which is an approximation by itself. But apart from it, extra numerical approximations may be done because of performance reasons.

- *Hypothesis:* Only nodal fluid velocities in the elements are taken into account and linearly interpolated to particle position without using a subgrid scale.

- *Limitation:* To improve the precision limited by FEM and its interpolation in a point inside an element, in the literature are found some techniques like using subgrid scales, but in this thesis it is considered that the mesh is fine enough to be able to interpolate linearly.

- *Hypothesis:* Particles are small enough compared to the mesh to consider them as a point.

  Limitation: Particles do not occupy a real volume in the domain, and all the forces must be considered to be active in the center of the particle, so these forces cannot cause a rotation in the particle due to off-center forces, for example. This hypothesis is commonly used for micro and nanoparticles.

- *Hypothesis:* The fluid is an incompressible flow.

  Limitation: While all flows are compressible, flows are usually treated as being incompressible when the Mach number (the ratio of the speed of the flow to the speed of sound) is less than 0.3.

- *Hypothesis:* When a particle hits a wall, no bouncing is considered and it is directly deposited.

  Limitation: Depending on the type of wall, particle, or type of collision, particles could bounce, but this factor could not be taken into account during this thesis.

- *Hypothesis:* The fluid interacts with particles, but particles do not with the fluid (which is called one-way coupling).

  Limitation: Large particles or a given high concentration of particles with large velocity (e.g., combustion) could affect the fluid.

- *Hypothesis:* The density of particles is small enough to neglect any type of mutual interaction like collisions.

  Limitation: As in the case of the one-way coupling limitations, the validity of this hypothesis depends on the concentration of particles and its size (Loth, 2000).

Some more specific hypothesis can be adopted during this thesis, as they are exposed in their specific sections.

1.5 Objectives

The aim of this thesis is to develop a versatile method for particle transport in a fluid flow that is functional in a distributed memory computational environment. This main goal can be split into more specific topics such as:

- Studying the forces affecting particles in every simulation case and developing their implementation.
- Studying the integration scheme with its convergence and accuracy. It is desirable to be able to find a suitable one for the most likely scenarios and applications.
- Discretizing the calculus within a FEM environment and in an efficient and accurate way.
- Coupling the fluid with the particles. This requires testing different methodologies and comparing them with different computational configurations (e.g., number of particles, level of resolution of the fluid, etc.).
- Developing the code for a distributed memory environment in an HPC context. This is mandatory to be able to work in parallel and for balancing fluid and particles resources.
- Applying the code to real cases and experiments. In order to validate the results obtained, they must be compared to other ones from the literature, experimental data, and benchmarks.
- Studying the results in the respiratory system airways and applying them to medical necessities.

1.6 Structure of this thesis

After the introduction this thesis leads to chapter 2, a biological and engineering context chapter with the goal of providing a general idea of

the organs that form the respiratory system and their respective biological functions. In this same chapter 2, the medical and engineering applications, and the main challenges of simulating the respiratory system airways, are exposed.

Once the reader goes in depth into the application context, most of the mathematics of this thesis can be developed. First, what is a computational domain and the whole life cycle of particles are explained in chapter 3. To solve Lagrangian particle transport, it is first necessary to obtain the fluid unknowns by solving the Navier-Stokes (NS) equations (chapter 4) and then the physics affecting the particles (chapter 5). However, external forces act over time and depend on the position of the particle, which also varies in time because of the same forces. For this reason, in chapter 6 an integration scheme is proposed for obtaining the position, velocity and acceleration of each particle in the function of time. The time is discretized and goes forward in chunks of time steps. That chapter also proposes an adaptive time step strategy for particles which are independent of fluid time steps.

As explained during this introduction, Alya code is adapted to HPC. For this reason, in chapter 7 the parallelization of the code and particle transport requirements are exposed and solutions proposed. Being able to computationally balance the fluid and particle resources turns out to be particularly challenging. Also in this chapter, a dedicated section to Brownian diffusion parallelization comes afterwards due to the fact that Brownian is considered to behave as a stochastic process. Because of this, specific parallelization peculiarities show up when parallel pseudorandom number generators are addressed.

Finally, benchmarks and results of simulating respiratory system depositions are shown in chapter 8, followed by final conclusions in chapter 9.

Chapter 2

Biological and engineering context

"Yes, they are elves" Legolas said. "and they say that you breathe so loud they could shoot you in the dark" Sam hastily covered his mouth.

J.R.R. Tolkien

Summary: Modelling particle transport in an HPC environment and applying it to respiratory system airways simulations involves knowledge and information in different fields. The goal of this section is to contextualize the respiratory system and its medical applications by using computational and mathematical tools. For this reason, a brief biological approach to the respiratory system is firstly exposed within the medical advantages of simulating it (section 2.1). The most relevant and significant numbers of respiratory system modelling are addressed next in section 2.2.

2.1 Respiratory system context

The main goal of this section is to understand the function of the organs forming the whole respiratory system, to provide a better comprehension of its behavior and the medical applications for simulating it. Also, this section aims to show how simulations can be of a great help for the study of pathologies that can occur in the system.

The function of the respiratory system is clearly known. It is obvious that respiration is in charge of breathing (inhaling and expiring through the

nose or the mouth). The whole system is formed by different organs, and each one has a specific function to ensure proper breathing.

As regards the medical context of the respiratory system, drug delivery via nasal sprays is one of the most studied applications. The respiratory system can also be studied to treat allergies, contaminated breathing or airways surgery. This is a very new concept in a computational environment, and at present there is not much research on the topic, for example, when compared to the cardiovascular system, which has been more deeply studied in a computational simulation environment as shown on figure 2.1.

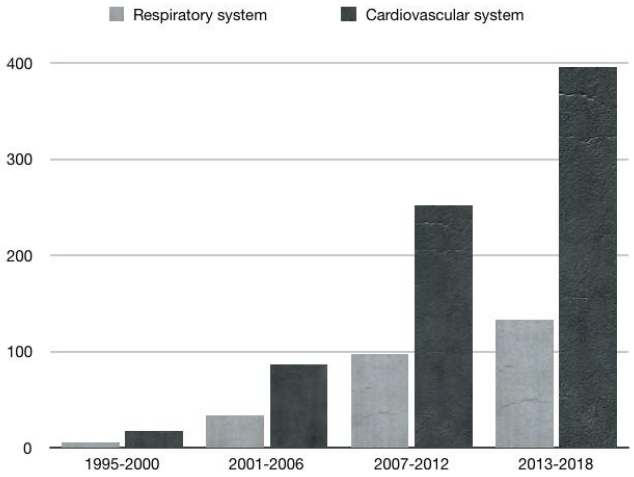


Figure 2.1: Number of entrances in google scholar looking up "respiratory system" + "computational simulations" and "cardiovascular system" + "computational simulations"

2.1.1 Biological overview of the respiratory system

The respiratory system consists of a group of organs and structures (e.g., nasal valve, larynx, trachea, bronchi branch, etc., as shown in figure 2.2) involved in the process of respiration. The global purpose is to draw oxygen into the body and expel carbon dioxide. Cells demand oxygen to obtain energy, whereas the carbon dioxide is the waste product of the metabolism.

The respiratory tract is divided into the upper airways and lower airways.

Upper airways division The upper airways or upper respiratory tract includes the nose and nasal passages, paranasal sinuses, the pharynx, and the portion of the larynx above the vocal folds (cords).

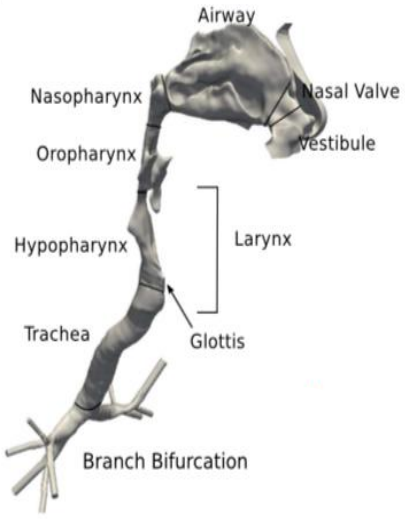


Figure 2.2: Different parts forming the respiratory system

- *Nose and mouth to breath.* Humans are the only mammal that not only breathe with the nose, but also with the mouth. The nose and the mouth are separated by a bone called the palate. In normal conditions, the nose is the preferential option for breathing, making it possible to chew food and breathe at the same time. Moreover, the nose is more efficient at filtering undesirable rubbish. For this purpose, each nostril (one of the two channels of the nose) contains hairs with the function of trapping the biggest dust particles. Sneezing is the mechanism for removing these trapped particles from the nostrils.

- *Nasal cavity.* There are two nasal cavities for each nostril, which are formed by the nasal vestibule at the entrance, followed by the nasal valve, and surrounded by paranasal sinuses.

  The nasal vestibule is the most anterior part of the nasal cavity. It is enclosed by the cartilage of the nose and lined by the same epithelium (one of the four basic types of animal tissue) of the skin. This type of skin is keratinized, meaning that it contains a resistant protein called keratin that waterproofs the epithelium. There are small hairs called vibrissae inside the vestibule to filter dust. The epithelium loses its keratinized nature inside the vestibule and undergoes a transition into a typical respiratory epithelium before entering the nasal valve.

  The function of the nasal cavity is to condition the air that will be

received by other areas of the respiratory track, cooling or warming it to within 1 degree of body temperature, and also humidifying and cleaning it by removing dust particles.

- *Pharynx.* The pharynx is the part of the throat behind the mouth and the nasal cavity (where both are joined). In humans it is part of the respiratory and digestive system, and is important for vocalization.
- *Larynx above vocal folds.* The larynx is involved in breathing, producing sound with vocal cords and protecting the trachea against food aspiration. The larynx is considered a division of the upper airways above the vocal folds or above the cricoid cartilage.

Lower airways division The lower airways or lower respiratory tract includes the portion of the larynx below the vocal folds, trachea, bronchi and bronchioles. The lungs can be included in the lower respiratory tract or as separate entity and include the respiratory bronchioles, alveolar ducts, alveolar sacs, and alveoli.

- *Larynx below vocal folds.* The part of the larynx below the cricoid cartilage houses the recurrent laryngeal nerve, which innervates the only muscle capable of opening the vocal cords.
- *Trachea.* The trachea is a cartilaginous tube that connects the pharynx and larynx to the lungs, allowing the passage of air. It is present in almost any air-breathing animal with lungs.

  The trachea of an adult has an inner diameter of about 1.5 to 2 centimeters and a length of about 10 to 11 centimeters.

  The trachea is surrounded by uncompleted C–shaped rings, although the cricoid cartilage at the top of the trachea attached to the larynx is the only complete ring.
- *Bronchus.* A bronchus is the passage of airway that conducts air into the lungs. The first bronchi to branch from the trachea are the right and left main bronchi, which are the widest ones. At each hilum (the root of the lung) they branch into a higher bronchi order. When the bronchi are too narrow to be supported by cartilage, they become bronchioles.

  No gas exchange takes place in the bronchi.

2.1.2 Why does modelling particle transport in respiratory system have medical interest?

Simulating the respiratory system using particle transport has evolved in the last few years due to its pharmaceutical and medical interest. Most of the

information exposed in this subsection can be found in a coauthored paper (Calmet et al., 2018),

The anatomy of the respiratory system is complex and variable, and each person has his/her own characteristics. Therefore, the ideal geometry used for simulation should be patient-specific, including the details of the person. Recent developments in Magnetic Resonance Imaging (MRI) and Computer Tomography (CT) techniques have helped in the reconstructing of physiologically realistic models. Such subject-specific geometries can be coupled with the advances in simulation technology to obtain detailed, accurate and realistic visualization of the flow field and particle transport and deposition (Kleinstreuer et al., 2014). Some examples of studies of direct drug delivery numerical analysis concerning nasal airway models that have been able to reveal this include:

- Detailed nasal airflow fields (Kimbell et al., 2007; Garcia et al., 2007).
- Particle dynamics (Schroeter et al., 2006; Garcia et al., 2015; Zhang y Kleinstreuer, 2011).
- Dosimetry of inhaled vapors (Asgharian et al., 2012; Schroeter et al., 2008)
- Odorant delivery (Keyhani et al., 1997).
- Nasal surgery (Rhee et al., 2011).
- Trachea diseases (Bates et al., 2016).
- Intranasal drug delivery (Kimbell et al., 2007; Inthavong et al., 2008; Gambaruto et al., 2014).

This thesis uses these case studies to focus on particle dynamics, which is directly related to another cited application for intranasal drug delivery.

Intranasal drug delivery and sniffs. The pulmonary route for direct drug-aerosol delivery is an engaging approach for fighting brain or lung diseases or for reaching systemic regions. This approach has great potential for optimal targeting solid tumors or severely inflamed areas with multifunctional particles, while provoking fewer side-effects and at lower costs than other treatment options, such as chemotherapy or radiation (Kleinstreuer et al., 2014; Kolanjiyil y Kleinstreuer, 2016; Kolanjiyil et al., 2017). By way of example, intranasal direct drug delivery is being considered as a possible and effective route for delivering vaccines, insulin, and medication to treat various diseases and disorders affecting the central nervous system (Illum, 2003; Mistry et al., 2009). At this point, intranasal drug delivery it is especially challenging for the inhaled drugs to reach the

olfactory region and once there obtain the chance of overpassing the blood–brain barrier, which can be an opportunity to reduce or eliminate brain tumors or to maximize the affect on the central nervous system (Dhuria et al., 2010; Thorne et al., 1995). Even so, intranasal delivery presents greater advantages: The drug-amount actually reaching the brain is quite low, but probably higher than intravenous administration (Thorne et al., 2004; Schroeter et al., 2006; Garcia et al., 2015).

In view of this, intranasal targeted drug delivery to a specific location could boost delivery efficiency (Inthavong et al., 2008; Shi et al., 2008a, 2007a). We can generalize that high particle deposition at any predetermined site will depend on the airway geometry and on the fluid-particle inlet conditions. This last point includes the breathing mode and the type of inhaler employed (Keeler et al., 2016; Segal et al., 2008). Moreover, the complex geometrical structure of the nasal cavity makes it difficult to predict the airflow and aerosol transport (Schroeter et al., 2010), while the geometrical variability among individuals raises important challenges in producing efficient drug delivery devices (Inthavong et al., 2008; Garcia et al., 2009a; Kimbell et al., 2007).

Limitations of experimental studies. Apart from computer simulations, deposition of inhaled aerosols in the human nasal cavity has been extensively studied using in vivo experiments (Kesavanathan y Swift, 1998; Kesavan et al., 2000; Rasmussen et al., 2000; Cheng et al., 1991, 1996), in vitro experiments (Garcia et al., 2009a; Cheng et al., 2001; Kelly et al., 2004a,b) and in silico ones (Schroeter et al., 2010; Garcia et al., 2015; Inthavong et al., 2006, 2008; Zhang y Kleinstreuer, 2011). The deposition results from these studies indicate that there are significant variations in human nasal aerosol deposition (i.e., fraction of the total inhaled particles deposited in the nasal area). The major reason for these variations is due to anatomical variations, but differences in experimental techniques can also affect the reported nasal aerosol deposition outcome (Shi et al., 2007a; Kelly et al., 2004a,b; Schroeter et al., 2011). Even though an in vivo deposition measurement on human subjects is the most physiologically realistic method, there are many limitations. These experiments are restricted due to possible side effects, especially when using radioactive aerosols, and hence are limited in the number of trials. Equally important, in vivo measurements cannot easily yield detailed regional deposition measurements or the existence of subject variability limits by comparative analyses without geometrical correlations (Rasmussen et al., 2000; Cheng et al., 1996). These limitations can be overcome with in vitro experiments, but in such case, and even small differences in the in vitro model geometry can significantly alter aerosol deposition. Recently, investigations have shown that surface irregularities (surface roughness vs. surface smoothness) due to the differences in the

fabrication process and/or due to the low resolution of the scanned images have resulted in significant variations in aerosol deposition (Shi et al., 2007a; Kelly et al., 2004a,b; Schroeter et al., 2011). Therefore, as an alternative to these experimental techniques, numerical analysis, i.e., using Computational Fluid-Particle Dynamics (CF-PD), has shown many advantages including repeatability and regional deposition resolution (Kolanjiyil y Kleinstreuer, 2016).

2.1.3 Particle types and sizes in respiratory system

Particle size is crucial for determining its penetration. Smaller particles will penetrate more deeply than larger ones. This can be good news for drug delivery, but bad for non-desirable particles, such as smog or tobacco, which have a considerably smaller size that makes them dangerous and harmful. In figure 2.3, some of the most typical inhaled particles are presented with their size for a better comprehension of their respective penetration capability. In the particular case of the nasal sprays, the commercial ones generate particles that fall approximately within a range of $d_p = 1\mu m$ to $d_p = 20\mu m$. However, there are studies in process to find out how they can be made to penetrate further (reaching alveoli regions) by using bolus inhalation (Heyder, 2004; Sturm, 2017) or smaller particle size such as nanoparticles (Buckley et al., 2016).

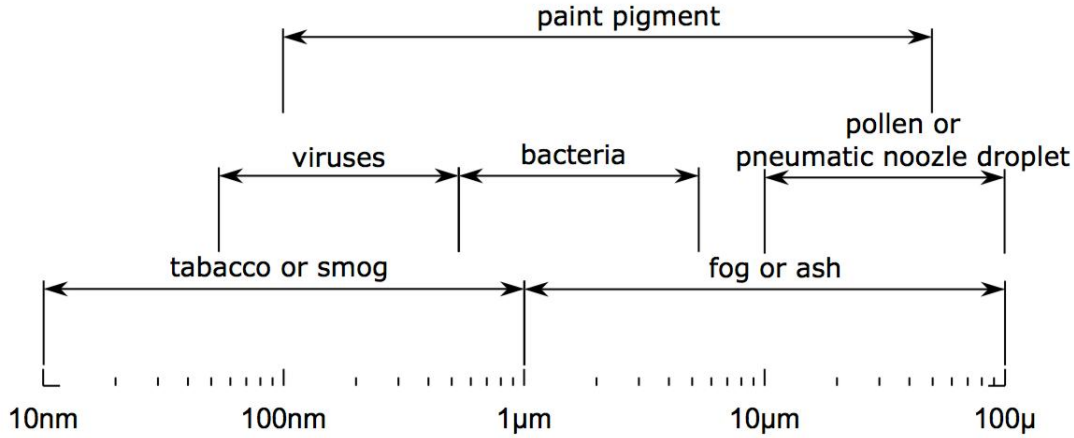


Figure 2.3: Examples of different typical inhaled particles and its size

2.1.4 Nasal aerosol deposition in accordance with particle types

Nasal deposition studies have shown that the nasal aerosol deposition is a function of inhalation conditions and particle properties, including size, shape and density (Shi et al., 2007a; Schroeter et al., 2015). These studies suggest that the nasal passage acts as a filtering mechanism for the incoming particles, which leads to large deposition in the anterior part, thereby reducing the amount of drug aerosols that reach their predetermined areas (Garcia et al., 2009a). In the inertial regime, considering a particle diameter equal to $d_p = 1\mu m$, the nasal aerosol deposition increases with the particle size and air flow rate, following a sigmoidal curve with very low deposition of lower micron particles (Shi et al., 2007a). Conversely, when a particle diameter (d_p) varies from $1nm$ to $100nm$ (which are considered nanoparticles), nasal aerosol deposition decreases in diameter due to the higher diffusivity of smaller particles leading to higher nasal aerosol deposition (Garcia et al., 2015; Shi et al., 2008a). Although this topic has been investigated for different inhalation conditions and particle sizes, only a limited number of studies have focused on estimating the regional distribution of the deposited particles (Schroeter et al., 2006; Garcia et al., 2015; Shi et al., 2007a), leaving the dependence of regional distribution of subject variability unknown.

2.1.5 Particle size and affecting forces in respiratory system airways

As previously explained, the size of the particle is a crucial characteristic in the study of particle deposition. Smaller particles penetrate deeper than larger ones. The first filters in the respiratory system are the mouth and the nose, which do not permit particles with a larger diameter than tens micrometers to penetrate, as represented on figure 2.4.

The size of the particle also implies which are the predominant forces in particle transport and which are negligible. In general, they can be roughly differentiated between microparticles and nanoparticles.

2.1.5.1 Most relevant forces in microparticle transport

Big particles are also heavier (assuming the same density). Thus, the bigger a particle is, the more its trajectory will differ towards the air pathlines. The force exerted by the fluid to try to make particles follow their pathlines is called drag force. Another force due to the flow that can also affect microparticles is the lift force (Li y Ahmadi, 1992) (in this case a force perpendicular to the oncoming fluid and particle relative direction), which is the same force that sustain planes in the air. Finally, gravity and buoyancy

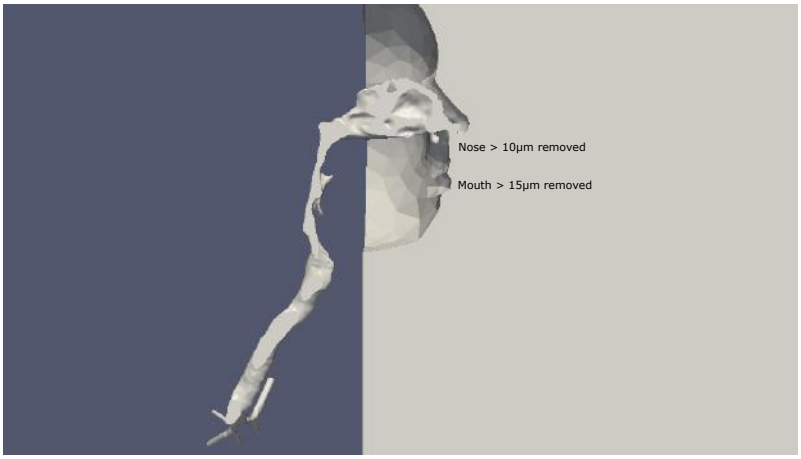


Figure 2.4: Preferential aerosol particle size deposition

can still have some effect on the transport of microparticles.

2.1.5.2 Most relevant forces in nanoparticle transport

The smallest particles resistance to the fluid is close to null. Thus, particles tend to follow the flow pathlines exactly. Lift force (except near wall (Zheng y Silber-Li, 2009)), gravity and buoyancy can be neglected (Li y Ahmadi, 1992). However, nanoparticles are light enough to notice constant air molecule collisions. This external force is assumed to behave as a random pattern, so it is characterized by stochastic techniques (A.Einstein, 1903).

2.2 Significant numbers in respiratory system simulations context

This is a good place to give some significant and general numbers of the air and the particles in the respiratory system that will help make the comprehension of this thesis easier.

2.2.1 Typical hypothesis of air properties in respiratory system simulations

The air in the respiratory system airways, as shown on figure 2.5, has the highest energy flux in the larynge and by generalizing, it can be considered to behave like a fluid with relatively low velocity fields (Calmet et al., 2016). This is an important property for deciding what method is used to solve the flow, as shown next in next section 4.

Air density ρ_f and viscosity μ_f are considered to be constant. The air at room temperature has a density $\rho_f = 1.18415\, kg/m^3$ and and viscosity $\mu_f = 1.85505 \cdot 10^{-5}\, \frac{kg}{m\,s}$.

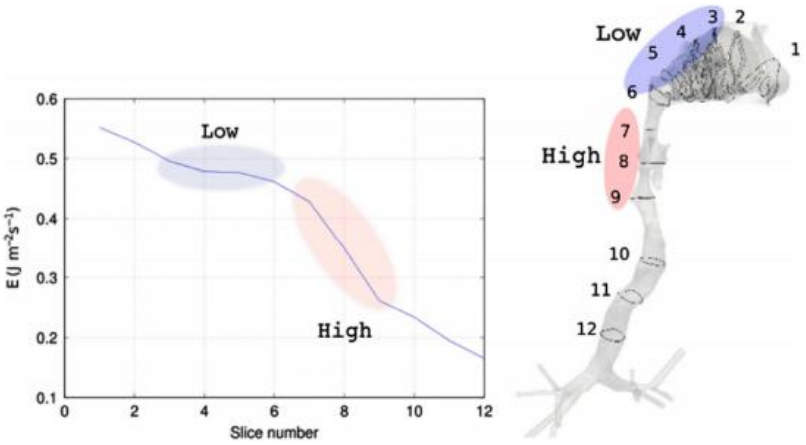


Figure 2.5: Energy flux computed at different sections along the airways.

In the final results, it is important to carefully study the inlet conditions for the flow. The entrance volume per time is called the flow rate. A higher flow rate means a more transient flow (or higher Reynolds number, as will be explained next in section 4). This flow rate varies if a sniff or a spray injection is simulated.

2.2.1.1 Flow rate profile in a sniff

In the case of a sniff the flow rate profile used in this thesis is shown in figure 2.6. The profile shown in the figure is obtained by a 10th order polynomial function describing the temporal evolution of the flow rate. (Rennie et al., 2011).

2.2.1.2 Flow rate profile in a spray particle injection

Flow rate in a spray delivery has a value between 20-40 L/min (Inthavong et al., 2011). The initial conditions of the particles are important too. However, there is still a lack of experimental data on these conditions (Inthavong et al., 2006). Some authors use analytical functions to model the initial particle conditions (Inthavong et al., 2006), and others use independent initial velocities (Basu et al., 2017).

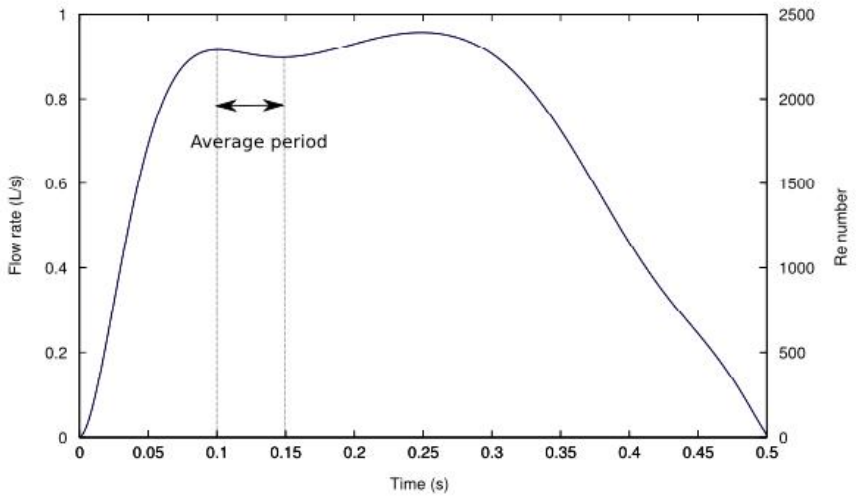


Figure 2.6: Flow rate profile during time in a sniff.

Chapter 3

Particle transport cycle

Life is like riding a bicycle. To keep your balance, you must keep moving.

A. Einstein

Summary: A computational simulation can be split into three main steps: pre-processing, processing and post-processing.

The particle transport models presented in this thesis are based on discretization methods and the use of meshes to define the geometrical domain (section 3.1). Obviously, this mesh must be generated before starting the calculation and when preparing for its parallelization, i.e., dividing it into subdomains. This process is called pre–processing (section 3.2).

The process is the part where the main calculations, according to the physics involved in the problem, are done. This process is the muscle of this thesis and many chapters are dedicated to it. In this section, in particular, we will explain how particles are injected, removed and located in the domain (section 3.3). Therefore, remarkable attention is given to the point (or particle) test inclusion technique by highlighting an innovative proposed method for high order elements.

Finally, post-processing involves any extra-calculus or visualization done once the simulation is over (section 3.4).

3.1 Introduction to discretization methods

A computational mesh is defined by a group of elements which approximates the geometry of a computational domain. Hence, meshing means discretizing the representation of the geometry involved in the problem. The elements forming the mesh make it possible to approximate the equations that define the physics of the problem in the space.

There are different types of meshes and discretization methods to approximate equations involved in the problem. There are also mesh-free methods which can obtain good solutions in cases where the geometry moves around or large deformations occur (e.g., if the expansion/contraction of the trachea is going to be taken into account). However, that approach is not considered in this thesis.

3.1.1 Types of meshes

The geometry defining the problem is discretized with a mesh. In some way, the points of this mesh will store the information needed to solve our equations in the space. These points form elements, and the set of elements consists in the computational domain.

Meshes can be structured or unstructured.

- *Structured meshes:*

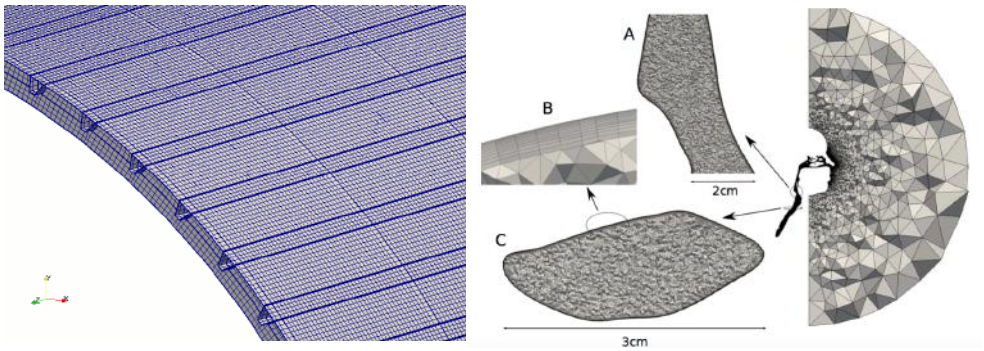
  Structured meshes have regular connectivity, which implies direct data addressing. Due to this fact, constructing high order schemes is simpler than unstructured meshes. They are usually used for the simplest geometries, as shown in figure 3.1(a).

- *Unstructured meshes:*

  Unstructured meshes have irregular connectivity, making them a more complex numerical option than structured meshes (e.g., interpolation inside the elements), but at the same time they enable generating more complicated geometries more accurately; thus avoiding approximate boundary conditions. In this thesis they will be used in respiratory system geometries as shown in figure 3.1(b).

Unstructured meshes can be composed of different types of elements. In the context of this thesis, hybrid meshes with different types of elements (tetrahedra, pyramids, prisms or hexahedra) are considered. Specifically, tetrahedra are used in respiratory system simulations, to define the inner part of the geometry, whereas the boundary layer (which refers to the layer of fluid in the immediate vicinity of a bounding surface where the effects of viscosity are significant) is built by prisms to improve the interpretation of the limits of the domain. In addition, pyramids are often required to connect prisms and tetrahedra.

It is important to emphasize that using unstructured meshes makes it much harder to calculate the exact position inside an element (this process is straightforward for structured meshes with linear elements). Indeed, this is one of the biggest computational processes of this thesis because it must be repeated for each particle and iteration, as will be deeply explained in section 3.3.2.



(a) (b)

Figure 3.1: (a) Simple 3D geometry with a structured mesh. Although elements may have different sizes, its conectivity remains regular. (b) Complicated 3D geometry (respiratory system) with unstructured mesh and different type of elements near walls.

3.1.2 Discretization methods

There are many different discretization techniques. Because of the complex geometries, one the most commonly used in respiratory system simulations is the Finite Elements Method (FEM), which right now is the most popular as shown on figure 3.2. In this figure, we compare the use of FEM against the Finite Volumes Method (FVM) and Lattice Boltzmann Method (LBM). FVM is the less popoular method, although in the 90s this was more common than LBM, that is becoming quite popular in recent years. It is important to emphasize that LBM rquires strucutred meshed and hence more elements for forming the geometry. FEM will be used in the particular case of this thesis.

Finite Element Method. The FEM was firstly developed for structural mechanics in the 1940s; but later adopted in Computer Fluid Dynamics (CFD). This method yields to approximate values of the unknowns at discrete number of points over the domain.

The FEM is based on the Galerking method. We are going to briefly describe the principles of the method. Let us consider the following partial differential equation

$$\mathcal{L}u = f \quad \text{in } \Omega, \tag{3.1}$$

where $\mathcal{L}$ is a second order operator, f is a forcing term, and Ω the spatial domain of interest. This equation is referred to as the *strong formulation* of the problem to be solved. Such an equation is typically an

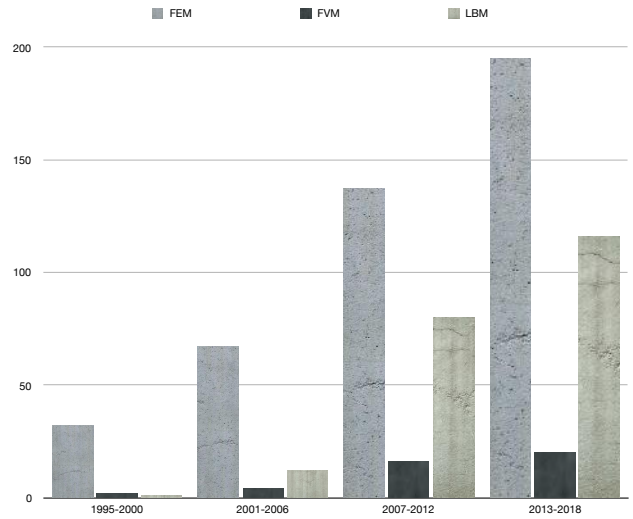


Figure 3.2: Number of entrances in google scholar searching "respiratory system" + one these three options: "finite elements", "finite volumes" or "lattice Boltzman"

advection-diffusion-reaction equation which models conservation principles, like conservation of energy, momentum, mass, etc. of the form

$$\mathcal{L}u := \boldsymbol{a} \cdot \nabla u - k\Delta u + su, \tag{3.2}$$

where $\boldsymbol{a}$ is the advection (velocity) field, k the diffusion coefficient, s the reaction and $\Delta \equiv \nabla^2$ the Laplacian. Formally, equation 3.1 must be supplied by boundary conditions on the domain boundary $\Gamma := \partial\Omega$.

If it exists, a solution to this equation is referred to as a strong solution but this is in general impossible to obtain analytically. One way around is to seek an approximate solution in a finite dimensional space, namely a discrete weak solution. The obtain such a solution, the first step consists in obtaining the weak formulation of the strong form 3.1. Let us define the residual of

the equation as

$$\mathcal{R}u := f - \mathcal{L}u. \tag{3.3}$$

Instead of requiring this strong residual to be zero, we can formulate a *weighted residual* equation by multiplying equation 3.3 by a weight function v, and integrate it over Ω. Then, the problem can be reformulated as: Find u in an appropriate space such that

$$\int_\Omega (f - \mathcal{L}u) v d\Omega = 0 \qquad \forall v \text{ in an appropriate space.} \tag{3.4}$$

The choice of the space depends on the regularity of the sought solution, the boundary conditions and the specific differential operators present in the equation. The operator $\mathcal{L}$ may have high order derivatives, thus imposing strong regularity requirements on the solution. In order to relax these requirements, we derive the weak form of equation 3.4.

To this end, we need the *Gauss's theorem* (also referred to as divergence theorem), which states that

$$\int_\Omega \nabla u d\Omega = \int_\Gamma u \boldsymbol{n} \Gamma, \tag{3.5}$$

where $\boldsymbol{n}$ being the outward normal to Γ. By applying this to the diffusion term of equation 3.4, we obtain

$$\int_\Omega \nabla u v d\Omega = -\int_\Omega \nabla u \cdot \nabla v d\Omega + \int_\Gamma v \nabla u \cdot \boldsymbol{n} \Gamma. \tag{3.6}$$

By doing this, we have reduced the regularity requirement on the sought solution u (by reducing the highest order derivative). The chosen space should therefore be such that this integral is bounded so that the problem makes sense.

Then, the *weak formulation* of the problem reads: find u in an appropriate space such that

$$a(u, v) = f(v), \qquad \forall v \text{ in an appropriate space,} \tag{3.7}$$

where the bilinear and linear forms a and f read:

$$a(u, v) := \int_\Omega (\boldsymbol{a} \cdot \nabla u) v d\Omega + \int_\Omega k \nabla u \cdot \nabla v d\Omega + \int_\Omega s u v d\Omega, \tag{3.8}$$

$$f(v) := \int_\Omega f v d\Omega. \tag{3.9}$$

The next step consists in choosing an appropriate discrete subspace of our original continuous space. The first choice consists in selecting the same spaces for u and v. This is the *Galerkin method*. In the finite element

method, the spaces are constructed from a meshing (e.g., triangulation in 2D) of the computational domain. This mesh should be formed by disjoint and conforming finite elements. By defining some shape functions N_i associated to each node i of the N nodes of the mesh, we approximate the solution u_h as

$$u_h(\boldsymbol{x}) = \sum_{i=1}^{N} N_i(\boldsymbol{x}) u_i. \tag{3.10}$$

In order to recover the nodal solution at each node of the mesh, that is $u_h(\boldsymbol{x}_i) = u_i$, the shape functions are such that

$$N_i(\boldsymbol{x}_j) = \delta_{ij}, \tag{3.11}$$

where δ_{ij} is the Kronecker delta such that

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \tag{3.12}$$

In addition, such functions are constructed so that they have compact supports, that is N_i is non-zero only in the elements node i belongs to.

To set up the discrete weak form, we then require that $u_h(\boldsymbol{x})$ satisfies the continuous weak formulation 3.7, and the weight function is selected such that N equations are generated. The discrete weak form then reads:

$$\sum_{i=1}^{n} \int_\Omega (\boldsymbol{a}\cdot\nabla N_i) N_j d\Omega + \int_\Omega k \nabla N_i \cdot \nabla N_j d\Omega + \int_\Omega s N_i N_j d\Omega = \int_\Omega f N_j d\Omega, \qquad \text{for } j = 1, 2, \ldots, N. \tag{3.13}$$

We thus generate N equations for the N unknowns of the problem, leading to an algebraic system of equations.

Let us finish with a practical implementation comment. As shape functions have a compact support consisting of the neighboring elements of the nodes of the mesh, they are never stored as global functions. Instead, they are defined element-wise. Then, when interpolating a variable at any point $\boldsymbol{x}$ of the computational domain, we just need to identify the element e it belongs to, referred to as the host element, and then interpolate the value as

$$u_h(\boldsymbol{x}) = \sum_{i=1}^{N_e} N_i^e(\boldsymbol{x}) u_i, \tag{3.14}$$

where N_e is the number of nodes of the element e.

This point is important as this is one of the main operations we will carry out when transporting particles through the finite element mesh used to solve the fluid equations.

3.2 Pre-process: generating a computational domain

The first step in any simulation consists in choosing the geometry that is required to be simulated. Using the respiratory system as example, it is obvious that if only deposition in the upper airways needs to be studied (e.g., drug delivery), it will be only necessary to build the upper respiratory system computational domain; otherwise, if deposition in bronchus needs to be studied (e.g., contamination respiration), upper and lower airways will be necessary to be included in the computational domain.

Once a computational geometry is chosen and generated, we are ready to discretize the computational domain into a mesh.

3.3 Process: Life circle of particles in the domain

Particle life during a simulation starts when they are injected. From the first moment particle coordinates and host element must be computed. During the simulation particles will move inside the domain, changing of host element and its coordinates until they finish their life cycle.

3.3.1 Beginning and end of a particle

A particle starts to exist once it is injected and it is removed when gets deposited, leave the domain or simulation is over. Next, how they get injected and removed is explained.

3.3.1.1 Particle injection

The first step to transport particles is injecting them in the domain. The injection can be done at the beginning of the simulation or during the simulation. For example, in the case the fluid must evolve before, the injection will be done after some time steps. In addition, particles can also be injected in a single injection, distributed in different injections or injected periodically (e.g., to guarantee a continuous particle flow, replacing removed particles).

Particles are injected using different 1D, 2D or 3D geometries. Next, are detailed the different patterns of injection implemented in Alya: Particles may be homogeneously distributed or randomly injected around the whole geometry. More than one geometry can be used in a single injection, likewise more than one type of particle can be injected in the same geometry.

1D Geometry:

- **Pointwise:** Coordinates of the point are demanded, x^i. Where i goes from 1 to the dimension of the problem.
- **Segment:** Coordinates of two points (both sides) are demanded, x_0^i and x_f^i.

2D Geometry:

- **Square:** Coordinates of two points, the maximum and the minimum are required, x_{min}^i x_{max}^i
- **Rectangle:** Coordinate of three points are demanded, x_1^i, x_2^i, x_3^i.
- **Circle:** Coordinates of the center, x_c^i, radius r and normal n^i are demanded.

3D Geometry:

- **Sphere:** Coordinates of the center, x_c^i and radius are demanded.
- **Semi-sphere:** Coordinates of the center, x_c^i, radius r and normal n^i are demanded.
- **Cone:** Coordinates of the center of the base, x_c^i, radius r, height h and normal n^i are demanded.

During the development of this thesis, aforesaid geometries have been necessary for different simulations. In general, 1D and 2D injections have been uses for test–cases. In the case of 3D, the sphere and semi-sphere are typically used for a respiration cycle where inhaled particles belong to the environment. The cone, otherwise, can be used to simulate spray injections.

In the case a new geometry is needed, this can be quite easily developed. As future work, it is also proposed to implement an injection option by selecting a boundary.

3.3.1.2 Particle outlet

Once particles have been injected, there exist three factors which provoke a particle to be removed from simulation:

A particle gets deposited when it touches a boundary considered as a wall. Wall boundaries can have three different behaviors. First, it can get deposited if radius of particles is larger than distance to wall. This way, a particle is removed from the simulation and written in post-process file as will be explained in 3.4. Second, particle bounces back. Particles moment and kinetic energy is conserved, considering it elastic. Third and final, wall's

boundary can have slip conditions. In this case, projection of velocity in the direction of the wall is maintained and keeps moving at this velocity sliding through the wall.

A particle gets out of the domain through a boundary considered outlet. A particle is removed and time, coordinates and particles ID (i.e., a global number which is different for each particle which allow to identify them) are written in an output file.

Simulation is over. The simulation can finish because the final time is reached or because all particles are out of the domain. In this last case, it can be a problem if few particles remain indefinitely in the domain. This can happen, for example, when a particle is very close to a wall, being fluid velocity almost null. If so, the simulation can have a problem, becoming a too long simulation, only becauase few statiscal irrelevant number of particles.

3.3.2 Particle inclusion test

To find the location of a particle, the so-called point-location problem needs to be solved. This problem consists in finding which element a particle belongs to (what it is called host element). For this purpose a particle inclusion test algorithm is built. The algorithm is composed of four main steps: three consecutive filters are applied, followed by the evaluation of the iso-parametric coordinates of the point within the hosting element.

- *Filter 1: A bin or oct–tree or first negihbour elements.* A list of host element candidates is created using a bin or oct–tree strategy Houzeaux y Codina (2003).

  Because of computational and numerical restrictions (deeper explained in section 6), a particle can only cross one element at each step. It means, at the initial injection of a particle it is necessary to use bounding boxes as explained in next subsection 3.3.2.1. After this, the list of candidates is only formed by direct neighbours of the previous host element.

- *Filter 2: Bounding box of the element.* The list of candidate elements is looped. Only if the point belongs to the element bounding box, next filter is applied.

- *Filter 3: The inclusion test.* An inclusion test method based on ray casting Ramsey et al. (2004) is used to check if the candidate element is the host element. This method is based on counting ray intersections with edges or nodes and apply the odd/even parity rule (if the number of intersections is odd, the point belongs to the element).

- *Calculation: Iso-parametric coordinates.* Once the host element is known, the iso-parametric coordinates of the point inside the element can be calculated using a Newton-Raphson (NR) iterator (explained in subsection 3.3.3.2).

In figure 3.3, a flowchart is shown outlining the aforesaid steps.

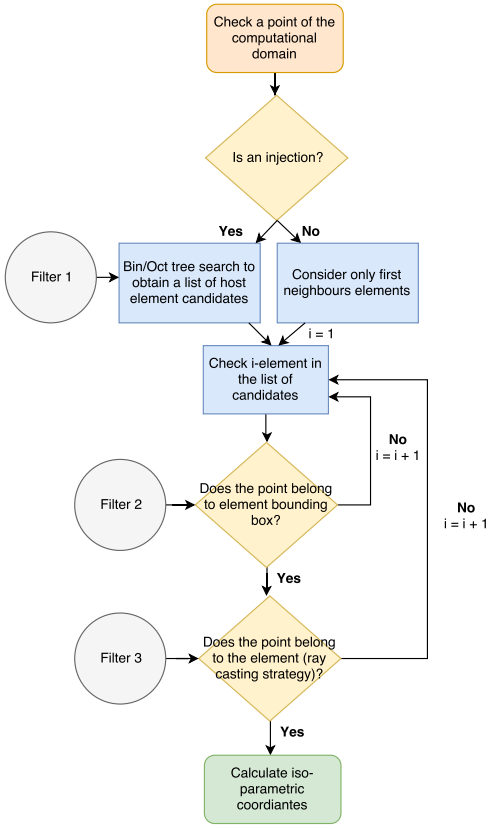


Figure 3.3: Flowchart of the inclusion test steps.

Next subsections are not ordered in the same order these filters and the calculus ara applied, as far as previous knowledge about iso-parametric coordinates is necessary (section 3.3.3) before detailing the ray intersection test.

3.3.2.1 The element search strategy

For a faster search of an element inside the computational domain (meshes can be formed by tens of millions of elements), elements are grouped in larger boxes called bounding box.

Therefore, the element search strategy (ESS) is based on a bin or an oct-tree strategy (Houzeaux y Codina, 2003).This is decomposed in two steps,

the pre-process (constructing the tree–like structure) and the process (range searching, which is explained in section 3.3.2).

During the pre-process, the computational domain is embedded in a box. The algorithm recursively partition each box into smaller boxes, until the box contains less than a prescribed number of elements. The bin strategy equally divides each box into a prescribed number of smaller boxes; the oct-tree strategy, otherwise, divides each box into different sizes smaller boxes (in $3D$ each box is typically divided into 8 smaller boxes). Oct–tree strategy in in general preferred for anisotropic meshes because we don't impose a maximum number of elements per box that limit the number of elements to test.

3.3.3 Natural coordinate system and shape functions

The finite element method considered in this thesis uses Lagrange functions to construct the element-wise interpolation shape functions. As mentioned earlier, the shape function of a node is equal to one on the node and 0 on the others, thus providing exact interpolated value on the nodes. In two-dimensional analysis, the simplest elements the triangle and the quadrilateral, which provide linear or bilinear interpolations, respectively. The three-dimensional elements which provide such interpolations, are the tetrahedra and hexahedra, respectively. Higher order elements can be constructed as well by introducing additional degrees of freedom (on the edges and interior of the elements, as well as on faces for 3D elements). Such elements will be introduced in 3.3.4.

The mesh is, in general, described in the Cartesian coordinate system, referred to as the global coordinate system. However, in the finite element method, a coordinate transformation is usually carried out to express quantities in a reference element, called the iso-parametric element. The natural coordinate system attached to this iso-parametric element is defined in such a way that local coordinates have values not exceeding unity. In addition, Lagrange shape functions are constructed on such elements, and thus the shape functions are never explicitly defined in the global coordinate system.

Figure 3.4 shows the most typical elements in natural coordinates.

If we work with natural coordinates, shape functions will allow us to carry out two type of transformations:

- Interpolate the velocity field from the nodes to any point inside the element.
- Convert global to natural coordinates and vice–versa.

If this is so, the element is of a type called iso-parametric. This elections facilitates to calculate integrals. In table 3.1 it is shown the limits of natural

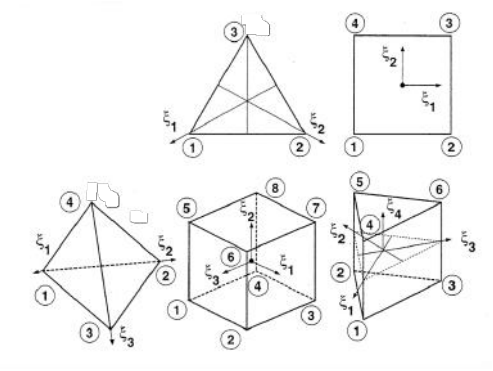


Figure 3.4: Iso-parametric elements coordinates: triangle, quadrilateral, tetrahedron, hexahedron and pentahedron.

Table 3.1: Limits of natural coordinates inside iso-parametric elements.

Topology	**Parametric Domain**
Triangle	$\xi_1, \xi_2 \in [0,1],\ \xi_1 + \xi_2 \leq 1$
Quadrilateral	$\xi_1, \xi_2 \in [-1,1]$
Tetrahedron	$\xi_1, \xi_2, \xi_3 \in [0,1],\ \xi_1 + \xi_2 + \xi_3 \leq 1$
Hexahedron	$\xi_1, \xi_2, \xi_3 \in [-1,1]$
Pentahedron	$\xi_1, \xi_2, \xi_3 \in [0,1],\ \xi_1 + \xi_2 + \xi_3 = 1,\ \xi_4 \in [-1,1]$
Pyramid	$\xi_2 \in [-1,1],\ \frac{-(1-\xi_2)}{2} \leq \xi_1,\ \xi_3 \leq \frac{1-\xi_2}{2}$

coordinate inside the most typical iso-parametric elements.

Let $\boldsymbol{\xi}$ be the vector representing the iso-parametric coordinates defined in an element. Given the shape functions of iso-parametric coordinates $N_k(\boldsymbol{\xi})$, shown in table 3.1, iso-parametric $\boldsymbol{\xi}$ coordinates can be transformed to global coordinates and vice-versa solving

$$\boldsymbol{\xi} = \sum_{k=1}^{N_e} N_k(\boldsymbol{\xi})\boldsymbol{x}, \tag{3.15}$$

where indexes k and N_e mean the node of the element and the total number of nodes respectively.

3.3.3.1 Transforming global coordinate system to natural coordinate system in linear elements

For linear elements (e.g., linear triangles o tetrahedron), the transformation from $\boldsymbol{\xi}$ to $\boldsymbol{x}$ is straight forward by isolating $\boldsymbol{x}$. For example, in $3D$ (the

procedure is the same for any dimension) the shape function reads

$$N_k = a_k\xi^1 + b_k\xi^2 + c_k\xi^3 + d_k, \tag{3.16}$$

where a_k, b_k, c_k and d_k are the derivative of the shape functions in function of $\boldsymbol{\xi}$ such that

$$(a_k, b_k, c_k) = \frac{\partial N_k}{d\xi^i} \quad \forall i \in 1, 2, 3. \ , \tag{3.17}$$

while d_k is the independent constant. In the particular case of the linear elements, these values (a_k, b_k, c_k) are constants. According to equation 3.15, we can write

$$\boldsymbol{x} = \sum_k (a_k\xi^1 + b_k\xi^2 + c_k\xi^3 + d_k)\boldsymbol{x}_k, \tag{3.18}$$

so

$$\boldsymbol{x} = \Big(\sum_k a_k\boldsymbol{x}_k\Big)\xi^1 + \Big(\sum_k b_k\boldsymbol{x}_k\Big)\xi^2 + \Big(\sum_k c_k\boldsymbol{x}_k\Big)\xi^3 + \Big(\sum_k d_k\boldsymbol{x}_k\Big). \tag{3.19}$$

Writing this using compact notation, we finally obtain

$$\boldsymbol{J}\boldsymbol{\xi} = \boldsymbol{x} - \sum_k d_k\boldsymbol{x}_k \tag{3.20}$$

where $\boldsymbol{J}$ represents the Jacobian matrix in function of a_k, b_k, c_k and $\boldsymbol{x}_k$. In the case of the tetrahedron with coordinates in the 4–nodes $(x_1, y_1, z_1), ..., (x_4, y_4, z_4)$, the Jacobian reads

$$\boldsymbol{J}_{tetrahedron} = \begin{bmatrix} -x_1 + x_2 & -x_1 + x_3 & -x_1 + x_4 \\ -y_1 + x_2 & -y_1 + y_3 & -y_1 + y_4 \\ -z_1 + z_2 & -z_1 + z_3 & -z_1 + z_4 \end{bmatrix}, \tag{3.21}$$

and while these nodes form a $3D$ tetrahedron (i.e., nodes are not coplanar), this is always invertible. Hence, doing the opposite transformation (from $\boldsymbol{x}$ to $\boldsymbol{\xi}$) only requires to invert the matrix $\boldsymbol{J}$,

$$\boldsymbol{\xi} = \boldsymbol{J}^{-1}(\boldsymbol{x} - \boldsymbol{x}_1), \tag{3.22}$$

which can be done analytically.

3.3.3.2 Newton-Raphson to transform global coordinate system to natural coordinate system in non–linear elements

In the case the element is not linear, the Jacobian matrix will depend on $\boldsymbol{\xi}$. Even though the transformation from natural to global coordinates keeps being trivial, inverting the transformation matrix (to compute the other way

transformation) is not anymore. For this reason, a Newton–Raphson (NR) is considered to solve the system $\boldsymbol{f} = 0$, such that

$$\boldsymbol{f} = \sum_{k=1}^{N_e} N_k(\boldsymbol{\xi})\boldsymbol{x}_k - \boldsymbol{x}, \tag{3.23}$$

where $\boldsymbol{x}$ is the exact position of the particle. Thus, applying the NR and using $\Delta\boldsymbol{\xi} = \boldsymbol{\xi}^{j+1} - \boldsymbol{\xi}^{j}$ we can write

$$\boldsymbol{f}^{j+1}(\boldsymbol{\xi}) = f^{j}(\boldsymbol{\xi}) + \boldsymbol{J}\Delta\boldsymbol{\xi}, \tag{3.24}$$

where

$$\boldsymbol{J} = \nabla_{\boldsymbol{\xi}}\boldsymbol{f}^{j}(\boldsymbol{\xi}), \tag{3.25}$$

being $\boldsymbol{J}$ the Jacobian matrix. Then, requiring that $\boldsymbol{f}^{j+1}(\xi) = 0$, we obtain

$$\boldsymbol{\xi}^{j+1} = \boldsymbol{\xi}^{j} - \boldsymbol{J}^{-1}\boldsymbol{f}^{j}. \tag{3.26}$$

As mentioned above, equation 3.24 is required to be iterated until reaching the zero. We can highlight that the proceadure explained for linear elements in previous subsection 3.3.3.1 could be considered as a NR with only one iteration.

This transformation is required after executing the ray cast intersection algorithm explained in next subsection 3.3.3.3, which allows us to find out the host element. But even with this information, we initially only have the global coordinates and herein the aforesaid conversion is required. Only after ξ^i is obtained with the NR, we are able to interpolate our unknowns, such as $\boldsymbol{u}_f$ (stored at the vertices) to the exact point where the particle is located. This process will be deeply explained in subsection 5.5.1.

NR makes sense only if $|\boldsymbol{J}| \neq 0$. This is always the case whenever $\boldsymbol{x}$ is inisde the element. This is why we perform the ray tracing strategy mentioned above. A particular case is the apex of the pyramid.

Pyramid apex issue with Newton-Raphson. Given the Jacobian calculated on equation 3.25, as mentioned above, it is not desired $|\boldsymbol{J}| = 0$, in which case $\boldsymbol{J}$ would not be invertible. It is common to include into the code a tolerance to avoid this kind of problems (e.g., if $\boldsymbol{J} < \mathit{toler}$ then remove this particle, or use a prescribed small Jacobian or in the worst of the case, stop the simulation within an error message which can facilitate future debugging). This tolerance is commonly a really small number such as the interval $10^{-8} \geq \mathit{toler} \geq 10^{-12}$. The election of this tolerance is often arbitrary. Typically, it is just chosen a small enough tolerance which does not alter the final result.

In the specific case of the pyramid three-dimensional element, close to the apex a very small Jacobian (defined in equation 3.25) is obtained. In

picture 3.5 is represented how the value of the Jacabian of a regular pyramid varies in fuction of its height, from its base to its appex. As shown, the minimum of the Jacobian is obtained at the appex, being there $\boldsymbol{J} = 0$ and, as a consequence, non–invertible.

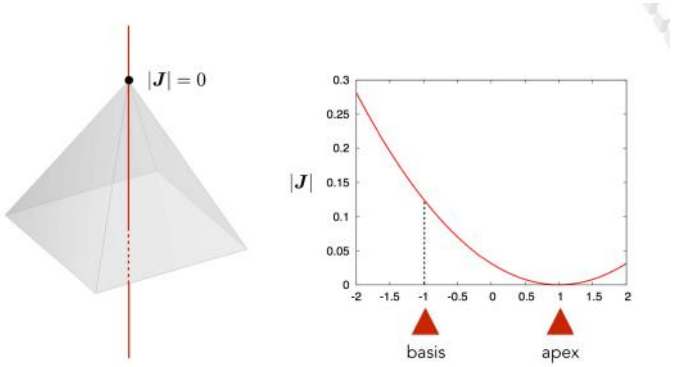


Figure 3.5: Values of $\boldsymbol{J}$ in a regular pyramid from base to appex.

Herein, the solution (using the iterative NR defined in equation 3.24) converges in just two iteration, plotted on graph 3.6. But even in the first iteration the Jacobian value is smaller than 10^{-12} which is the smallest tolerance proposed in the aforesaid interval of possible tolerance election.

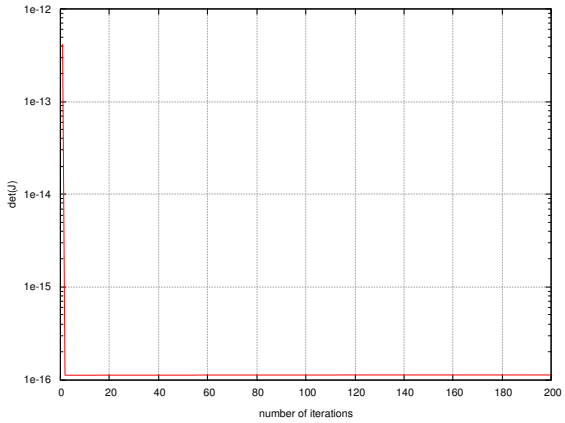


Figure 3.6: Value of $\boldsymbol{J}$ close to the pyramid appex in function of the iterations.

Hence, a wrong election of the tolerance can cause computational lost of particle or even stop the whole simulation. Before finding out this error, if the Jacobian was too small a prescribed diagonal small Jacobian was used instead. That caused a particle in the apex never being able to find its

host element, entering into an infinite loop. An infinite loop causes the whole simulation paralyzing, thus it is preferable to remove this particle instead. Although this problem was fixed and in the moment of writing this thesis, no particle was removed because of computational reasons, in section 6.2.1 it is explained how a particle is prefereable to be removed instead of conditionating the whole simulation as a security measure.

3.3.3.3 Ray intersection strategy for non-linear elements

Once a reduced number of candidate elements to host the particle is obtained, because of the bounding box or because the particle can only move one element through, a ray casting strategy is used (but for linear elements, which only require to obey conditions from previous table 3.1).

This strategy consists in casting a ray which length is much larger than element characteristic length ($l \gg L$). The origin of this ray is in the point (or particle) location of the feasible host element we want to check. The formula of this parametric line is given by

$$\boldsymbol{x}(\lambda) = \boldsymbol{x}_0 + \lambda \boldsymbol{r}. \tag{3.27}$$

Any point $\boldsymbol{x}$ along the ray can be expressed by a specific λ value which corresponds to the parametric distance along the direction vector $\boldsymbol{r}$ with origin $\boldsymbol{x}_0$. Specifically in Lagrangian particle transport: $\boldsymbol{x}_0 = \boldsymbol{x}_p$.

If the ray crosses once a face of the element, this particle is inside, otherwise if it crosses two faces, it is outside. Generalizing we can use the parity rule: odd number of intersections mean inside, even number of intersection outside. However, we must be careful if the ray intersects a vertex. A vertex has more than a face, so this rule would be faked. For this reason, if the ray intersects a vertex, a new ray with a different direction is casted until vertex intersection does not occur.

Checking if the ray is intersecting a face is done using different procedures in function of the geometry of the face. Sorting these procedures from cheaper to most expensive:

Triangle faces. If the face is a triangle, we can check if the point is coplanar with simple algebraic manipulations.

Quadrilateral faces. If, otherwise, the face is a parallelogram, as far as it is formed by 4 points, there exist more than an unique plane, so the previous strategy is no longer valid. When points are not coplanar, this is not considered a parallelogram, only a bilinear face. For this form, an algorithm called ray bilinear patch intersection described by (Ramsey et al., 2004) is used. Briefly, this algorithm generates bilinear patches formed as a combination of four possible non-coplanar points $(\boldsymbol{x}_{00}, \boldsymbol{x}_{01}, \boldsymbol{x}_{10}, \boldsymbol{x}_{11})$ which

fulfill equation 3.27 for $\lambda \geq 0$. These points can generate a bilinear patch equation by wighting them with two parameter $(u, v) \in [0, 1]^2$. Finally, the interaction must be calculated, being u, v and λ the unknowns.

3.3.4 High order methods

One of the main features of using higher order methods is that in the case the exact solution of a partial differential equation (PDE) is smooth and has no singularities in the domain, then the approximation calculated converges exponentially with the order of the approximating polynomial (Babuska et al., 1981; Szabo y Babuska, 1991). As a consequence, it has been shown that high-order methods provide better accuracy with lower computational cost than low-order methods in a wide range of applications (Vos et al., 2010; Cantwell et al., 2011; Löhner, 2011; Huerta et al., 2012; Wang et al., 2013).

This assumption so that the convergence rate for high order methods is obtained is that also the geometry is represented with high order accuracy. Hence, the boundary faces must be curved to match the domain boundaries with the accuracy determined by the order of the solution approximation (Gargallo Peiró, 2014).

Therefore, if a quadratic order method is required, also quadratic elements will become necessary.

3.3.5 Quadratic elements

When quadratic elements are used (examples of quadratic elements are shown of figure 3.7), the aforesaid intersection strategy from section 3.3.3.3 becomes insufficient, requiring a new methodology.

Figure 3.7: Two examples of quadratic elements: on the left, a 20-node brick, on the right a 10-node tetrahedron.

The use of higher order methods and elements is still pretty novel and not very common in the literature yet (Gargallo Peiró, 2014). Quadratic elements can be used to describe specially complex and curved geometries, like biological structures, e.g. respiratory system airways. Higher order elements allow to adopt a curved surface using fewer elements than with first order elements. However, quadratic elements also imply more algebraic complexity, as far as shape functions $N_k(\xi^i)$ have crossed and second order terms. Until now, it was probably computationally cheaper generating finer meshes using

first order elements, than coarser meshes with higher order elements. But with new architectures where loading can be more computationally expensive than processing, this may change. Another possible advantage of quadratic elements is decreasing the truncation error of Navier-Stokes equations 4.1 and 4.2 when the fluid is solved by DNS, getting closer to the resolution we theoretically should obtain given the Kolmogorov scale defined further by equation 4.9.

To check the intersection between a ray and a curved plane, no coplanar technique can be used as above. For this reason, we propose a innovative proceeding using a NR to find intersection by solving the root of the norm of the element's face and ray. This procedure will allow us to work in a one dimension space $\Re^{ndime} \rightarrow \Re$.

Given the equation of the ray 3.27, we apply a NR to find the intersection between this ray and the quadratic plane formed by $N_k(\xi^i)$.

$$\left\| \sum_{k=1}^{N_e} N_k(\xi^i) x_k^i - (\lambda \mathbf{r} + x_0) \right\| = 0. \tag{3.28}$$

Herein we are obtaining the natural coordinates position of the intersection with each face. Likewise with first order elements, we can check if ξ^i obey the natural coordinates value conditions to belong to the crossing face. If it does, it means the ray is crossing that face. It is important to highlight that tangent intersections must not be counted; otherwise, the parity rule is broken.

A quick test to check the algorithm consists in injecting particles randomly in a box a larger enough to bound a mesh formed by only one element, a quadratic tetrahedron (10 nodes). Thousands of particles are injected using a Monte–Carlo (MC) stochastic procedure, as a result we can graphically draw the element with particles accepted as shown in 3.8.

3.3.6 Future work with high order elements

As future work, this new technique should be more deeply tested using complex geometries and studying this double NR convergence, as this is still found in a very early development state.

By way of example, a quick test with a quadratic triangle to check the convergence of the first NR (the one in charge of obtaining iso-parametric coordinates) can be generating a grid of points that cover the whole element such as shown in figure 3.9. Here, a convex quadratic triangle is shown and we prove that the NR converges very quick because of its quadratic convergence (much more than other tested algorithms such as gradient descent method, which can require even more than 1000 iterations).

However, if we repeat the same test with a quadratic concave triangle, as shown in figure 3.10, the convergence in this case depends on the initial guess

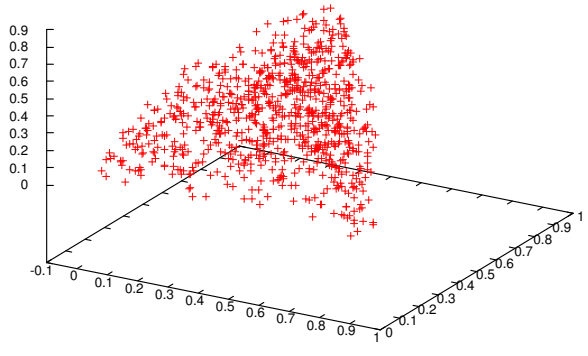
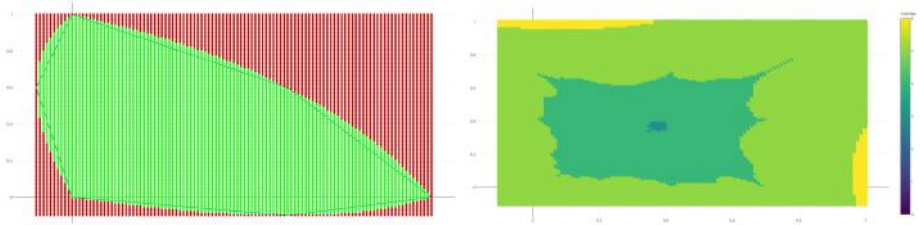


Figure 3.8: Quadratic tetrahedron element test inclusion using MC stochastic injection.

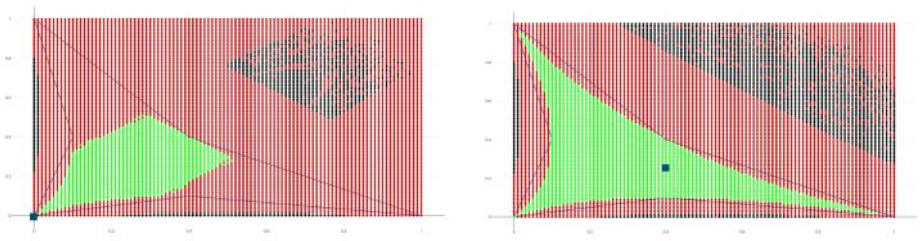


(a) (b)

Figure 3.9: (a) Green points mean that the point converged inside the element. Red points that converged outside the element. (b) The maximum number of iterations using a NR is 6.

of the NR. If the first guess is $(\xi^1, \xi^2) = (0, 0)$, as shown in 3.10(a), many points theoretically inside of the element are obtained as if they were outside. Othertwise, if the first guess is $(\xi^1, \xi^2) = (1/3, 1/3)$, as shown in 3.10(b), the result is much more close to the reality. In both cases, outside the element, there can be found black points, which mean that NR did not converge after 1000 iterations. Obviously, the dependence on the initial guess is not a good news for the method, so further work must be done here.

As a conclusion, we have presented a still very early-state algorithm based on two iterative methods (i.e., two consecutive NR). Although the NR tends to converge very quick it can strongly depend on the initial guess.



(a) (b)

Figure 3.10: (a) Starting from one of the nodes of the triangle (dark blue square) many of inside points are found as outside. (b) Starting from the center (dark blue square) it converges to an accurate result.

In addition, it is possible that using two consecutive iterative methods for computing the point–location problem millions of times does not allow to obtain a better performance using quadratic elements if particle transport is involved.

If a better performance could be proven, any high order level can be reach with this methodology.

3.4 Post-process: studying the obtained results

Once simulation is over, it's time post-process its results and obtain conclusions. Accurate results or a brilliant HPC performance have non-sense if after the simulation it gets complicated to extract information, visualize results or compare data. Smart post-process outputs can be as important as the rest of the simulation.

3.4.1 Particles trajectory

Describing particle trajectories may not be as trivial as it could be at first look. The trajectory of every particle is kept every global time step with particle's ID, coordinates and velocity in a .csv format file and it also kept the number of particles crossing an element every global time step in its nodes. Both formats have its pros and cons.

On the one hand, .csv format allows to draw individual trajectories, which results very visual. On the other hand, when millions of particles are simulated and hundred of thousands steps are computed, file can weigh the order of terabytes TB. This complicates its visualization because the high amount of memory required. The simplest solution is to visualize only a representative sample (e.g. taking only a percentage of the total particles

randomly). A more complex one, is to build a distributed system for post-process and visualization. For example, using Paraview, an open-source visualization tool, a Hadoop system was built inside in order to be able to use map/reduce technique with the csv file (Artigues et al., 2015).

Another way of analyzing the particle distribution consists in using classical postprocess programs to visualize nodal values. For example, to compute the nodal density of particles, particles are accumulated in elements, and then the values on the elements is projected on the nodes, using a L^2 projection.

3.4.1.1 Deposition

Studying the deposition is of great interest in the respiratory system simulations. Being able to check where drugs (e.g. nasal spray) get deposited and filtrate to blood, how breathed polluted particles affect, sampling deposited particles in function of its diameter or the deposition efficiency of the system are some of the examples which make this statistic crucial.

When a particle touches a wall (diameter smaller than distance to wall) with deposition boundary condition, this particles is removed from simulation and time, coordinates and ID are written as a csv file.

3.4.1.2 Residence time

Some topics like combustion or ventilation focus its interest in the study of the total time particles stay in a certain region, this is called the residence time. To compute it and in order to computationally simplify it, the time is kept in the nodes and accumulated for every particle. When a particle crosses an element, in order to simplify the calculus, half of the time is added to the previous element, and the other half to the new one.

3.4.1.3 Aditional statitstics

Some additional statistics are written as an output in order to understand and study the behaviour of our algorithm or how fluid complexity can affect the convergence. These are:

- Accumulative number of NR iterations in element's node to find out which regions have a bigger complexity to converge.
- Number of adaptive time steps (sub-steps) necessary to reach the global time step.
- Mean time of the sub-steps.

Chapter 4

Fluid Solver

Science is a differential equation.
Religion is a boundary condition.

A. Turing

Summary: This chapter describes how the fluid, which is one of the main elements of particle transport, is solved. Modeling the fluid implies solving Navier-Stokes (NS) equations (section 4.1). These equations arise from applying Newton's second law to fluid motion, making the assumption that the stress in the fluid is the sum of a diffusing viscous term and a pressure term.

It has not yet been proven that in three dimensions, there are always solutions, or in the case they do exist, then they are smooth. As NS equations almost never have an analytical solution, this means that discretizing the space and the time is necessary for its resolution, and only approximate solutions can be sought. Here, there are different approaches in the literature (section 4.2).

4.1 Governing equations in respiratory system air simulation

In the case of the respiratory system, the fluid in the airways is air and it is considered as an incompressible flow.

The air in the respiratory system airways, as shown in previous section 2.2, can be considered to behave as a fluid with relatively low velocity and incompressible fields. Thus, a flow with constant density ρ_f and viscosity μ_f is considered. In the specific case of air at room temperature ($T \simeq 300K$), these values are $\rho_f = 1.18415kg/m^3$ and $\mu_f = 1.85505 \cdot 10^{-5}Pa \cdot s$, whereas

the flow unknowns are obtained by solving the incompressible NS equations

$$\rho_f \frac{\partial \boldsymbol{u}_f}{\partial t} + \rho_f(\boldsymbol{u}_f \cdot \nabla)\boldsymbol{u}_f - \nabla \cdot [2\mu_f \boldsymbol{\epsilon}(\boldsymbol{u}_f)] + \nabla p = 0, \tag{4.1}$$

$$\nabla \cdot \boldsymbol{u}_f = 0, \tag{4.2}$$

where $\boldsymbol{u}_f$ and p are the fluid velocity and pressure, whereas the velocity strain rate $\epsilon(\boldsymbol{u}_f)$ is defined as

$$\epsilon(\boldsymbol{u}_f) = \frac{1}{2}[\nabla \boldsymbol{u}_f + (\nabla \boldsymbol{u}_f)^t], \tag{4.3}$$

which is a symmetric tensor, super-index t meaning the transposition.

In fluid mechanics, in order to characterize the flow patterns in fluid flow simulations, a dimensionless quantity called the Reynolds number is defined (or in our case we will specify it as Reynolds number of the fluid as Re_f, this way we can differentiate it from other future Reynolds numbers applied to particles)

$$Re_f = \frac{\boldsymbol{u}_f L}{\nu_f}, \tag{4.4}$$

where $\boldsymbol{u}_f$, L and ν_f mean fluid velocity, the characteristic length and kinematic viscosity, which is defined as $\nu_f = \frac{\mu_f}{\rho_f}$. Reynolds number of the flow quantifies the importance of inertial forces with respect to viscous forces.

The Reynolds number of the fluid in the respiratory system has values characterized in the transition between laminar and turbulent flow as detailed in depth in (Calmet et al., 2016). The biggest Reynolds numbers are found in the larynge, where the energy spectrum of stream-wise velocity fluctuations at different locations behave as shown in figure 4.2. As represented in figure 4.1 in the zone of highest stream-wise velocity fluctuations (point 7), Re_f is occasionally large enough to need the inclusion of some kind of turbulence model, as we will see later in subsection 4.2. Specifically, in points 7, 8 and 9, Re_f is valued as:

- *Point* 7: $2.2 \cdot 10^5$,
- *Point* 8: $1.0 \cdot 10^5$,
- *Point* 9: $0.5 \cdot 10^5$.

4.1.1 Numerical model to solve Navier-Stokes equations

Spacial numerical discretization. The numerical model for solving these equations is based on a FEM (which is explained in more detail in the chapter 3) stabilized by a Variational Multi-Scale (VMS) method first proposed by (Hughes, 1995). The formulation is obtained by splitting the

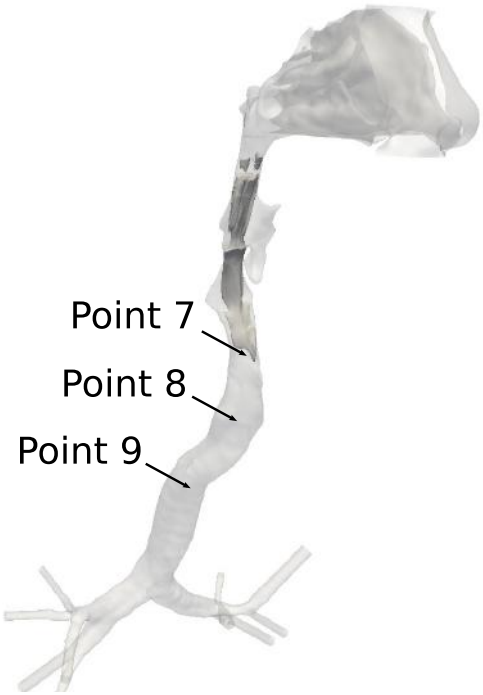


Figure 4.1: The laryngeal jet and the location of the three points.

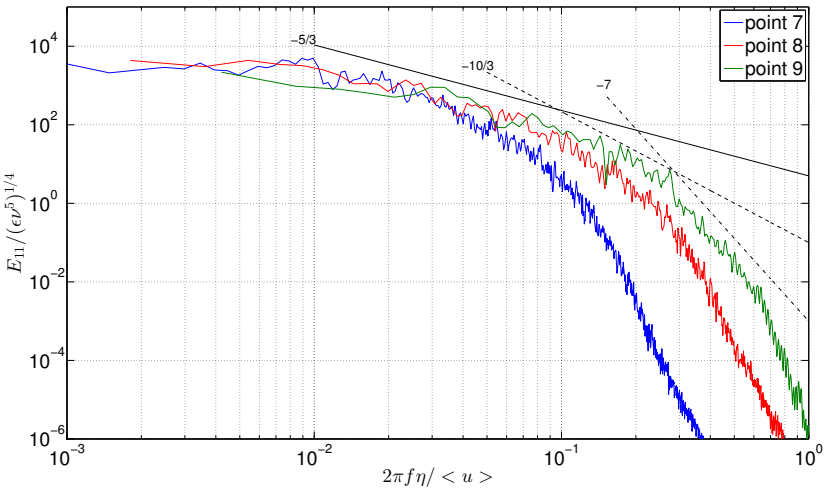


Figure 4.2: Normalized energy spectrum of stream-wise velocity fluctuations at three different locations (shown in the previous figure) downstream of the jet.

unknowns $\boldsymbol{u}_f$ and p into the grid and subgrid scale (SGS) components, such that

$$\boldsymbol{u}_f = \boldsymbol{u}_f^h + \boldsymbol{u}_f', \tag{4.5}$$

$$p = p^h + p'. \tag{4.6}$$

Here, the Galerkin method is applied to $\boldsymbol{u}_f$ and p and the subgrid scales are approximated using an algebraic model. The SGS velocity is, in addition, tracked in time and in space, thereby giving more accuracy and more stability to the numerical model. The VMS formulation is thoroughly described in (Pozorski y Apte, 2009), and the solution of the corresponding algebraic system is presented in (Houzeaux et al., 2011a) and its parallelization in (Vázquez et al., 2016).

Time numerical discretization. Time discretization from step n to $n+1, n+2...$, given a time step δt_f is carried out using the implicit method called a backward differentiation formula with order 2 (BDF2).

In BDF2 scheme the time derivative of the velocity is approximated as

$$\rho_f \frac{\partial \boldsymbol{u}_f}{\partial t} \simeq \frac{\rho_f}{2\delta t_f}(3\boldsymbol{u}_f^{n+1} - 4\boldsymbol{u}_f^n + \boldsymbol{u}_f^{n-1}), \tag{4.7}$$

where the time step δt_f can either be prescribed or computed automatically from the critical time step (Houzeaux et al., 2009).

4.1.2 Solution strategy

A fractional step scheme is used to solve the incompressible Navier-Stokes equations. This scheme is implemented at the algebraic level and converges to the monolithic solution. It is equivalent to using a fractional step scheme or an iterative method for the Schur pressure complement system (Houzeaux et al., 2011b). Next follows an Orthomin(1) iteration that minimizes the Schur complement residual at each solver iteration by dynamically computing a factor when the updating step is introduced.

Algebraic system. After space and time discretization, the following algebraic system must be solved at each time step:

$$\begin{bmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_u \\ \mathbf{b}_p \end{bmatrix}, \tag{4.8}$$

where

- $\mathbf{A}_{uu}$: Galerkin momentum, pressure subgrid scale, velocity subgrid scale (SUPG like term);

- $\mathbf{A}_{up}$: Galerkin pressure gradient, velocity subgrid scale;
- $\mathbf{A}_{pu}$: Galerkin velocity divergence, velocity subgrid scale;
- $\mathbf{A}_{pp}$: pressure stabilization: for that reason, no additional stabilization properties are sought by splitting.

This system has to be solved at each non-linear iteration of each time step, until convergence. There are two main options for solving system 4.8. Resolving it monolithically or by fractional scheme, by splitting the momentum and continuity.

Algebraic split strategy. In this thesis, an algebraic split strategy will be considered. First, the Schur complement pressure equation is extracted:

- **i) Extract pressure Schur complement**

  1-. $\mathbf{Sp} = \mathbf{b}_s$,

  2-. $\mathbf{S} := \mathbf{A}_{pp} - \mathbf{A}_{pu}\mathbf{A}_{uu}^{-1}\mathbf{A}_{up}$,

  3-. $\mathbf{b}_s := \mathbf{b}_p - \mathbf{A}_{pu}\mathbf{A}_{uu}^{-1}\mathbf{b}_u$.

- **ii) Solve with Orthomin(1)**

  1-. Solve momentum equation $\mathbf{A}_{uu}\mathbf{u}_f^{k+1} = \mathbf{b}_u - \mathbf{A}_{up}\mathbf{p}^k$,

  2-. Compute Schur complement residual $\mathbf{r}^k = [\mathbf{b}_p - \mathbf{A}_{pu}\mathbf{u}^{k+1}] - \mathbf{A}_{pp}p^k$,

  3-. Solve continuity equation $\mathbf{Qz} = \mathbf{r}^k$,

  4-. Solve momentum equation $\mathbf{A}_{uu}\mathbf{v} = \mathbf{A}_{up}\mathbf{z}$,

  5-. Compute $\mathbf{x} = \mathbf{A}_{pp}\mathbf{z} - \mathbf{A}_{pu}\mathbf{v}$,

  6-. Compute $\alpha =< \mathbf{r}^k, \mathbf{x} > / < \mathbf{x}, \mathbf{x} >$,

  7-. Update velocity and pressure:

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \alpha\mathbf{z},$$

$$\mathbf{u}_f^{k+2} = \mathbf{u}_f^{k+1} - \alpha\mathbf{v}.$$

This converges into the same solution as the monolithic system.

The algorithm described above is an Orthomin(1), required for solving the momentum equation twice and the continuity equation once. This is called the momentum preserving Orthomin(1) method, because after one shot of this algorithm the momentum is preserved. Another option could be building a continuity preserving version. In this case, the continuity equation would be solved twice, and a correction step would be needed to enforce the continuity of convergence properties. The algorithm and its performance are extensively described in (Houzeaux et al., 2011b).

4.2 Turbulence modeling

According to Bradshaw: *turbulence is a three dimensional time-dependent motion in which vortex stretching causes velocity fluctuation to spread to all wavelengths* (Bradshaw, 2013). Mathematically, the non-linear terms of equations 4.1 and 4.2 are responsible of this process. The importance of these non-linear terms are directly dependent on the Reynolds number of the flow Re_f. A higher Re_f implies more non-linearities, and as a consequence a more turbulent flow.

Different solutions are proposed in order to be able to describe accurately the chaotic changes in pressure and flow velocity (in other words, the turbulence) at different levels and computational costs. In the literature there are three main approaches to turbulence modelling.

- *Direct numerical simulation (DNS):*

  NS equations are numerically solved without any turbulence model. This means that the whole range of spatial and temporal turbulence scales is resolved. Therefore it is the most precise approach in terms of resolution of equations. However, it is extremely computationally expensive to solve the smallest length scales, due to the large number of points required for this purpose, it requires extremely fine meshes. Let us define Kolmogorov scale η determined by

  $$\eta = \left(\frac{\nu_f^3}{\epsilon_k}\right)^{1/4}, \tag{4.9}$$

  where ν_f and ϵ_k are the kinematic viscosity and the rate of kinetic energy dissipation. In order to satisfy the resolution requirements, the number of points N along a given mesh direction with increments h, must be $Nh > L$. If so, the integral scale is contained within the computational domain and also $h \leq \eta$, so that the Kolmogorov scale can be resolved.

- *Large eddy simulation (LES):*

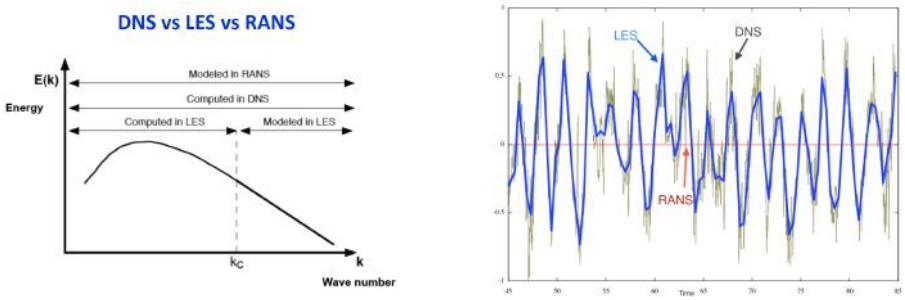
  Initially proposed by (Smagorinsky, 1963) to simulate atmospheric air currents, and later developed deeper by (Deardorff, 1970), LES is nowadays applied to a wide variety of engineering applications. It resolves a very wide range of time and length scales, all of which affect the flow field. Such a resolution can be achieved with DNS but in the case of LES, with a cheaper computational cost ignoring the smallest length scales using low-pass filtering. The filtering operation removes scales associated with high frequencies, and the operation can be interpreted in Fourier space. Given a scalar spatial and temporal field $\phi(\boldsymbol{x}, t)$, its Fourier transformation is $\phi(\hat{\boldsymbol{k}}, \omega)$ where $\boldsymbol{k}$ and ω mean

wave number and temporal frequency. Thus, the resolved sub-filter scales represent the scales with wave number larger than the cutoff wave number k_c.

- *Reynolds-averaged Navier-Stokes (RANS):*

  This is a time-averaged solution to the Navier-Stokes equations. Instantaneous quantity is decomposed into its time-averaged and fluctuating quantities. This idea was first proposed by Osborne Reynolds (Reynolds, 1895).

To summarize, the main differences between DNS, LES and RANS are graphically represented in charts 4.3.



(a) (b)

Figure 4.3: (a) LES computes above k_c and models beyond it. RANS models the whole range whereas DNS computes the whole range of wave numbers. (b) RANS averages over time.

In our particular case, the VMS method, which will be the most common way to solve the fluid in this thesis, can be considered, somehow, an implicit large eddy simulation (iLES), as described in (Grinstein et al., 2007). iLES methods capture the energy-containing and inertial ranges of turbulent flows, while their own intrinsic dissipation acts as a subgrid model (Aspden et al., 2009).

A discussion can be found in the literature about the limits of iLES achieving to behave as a DNS method (Aspden et al., 2009). If moderate Reynolds numbers are considered and meshes result fine enough to resolve all the Kolmogorov scales, as explained in more detail in a coauthored paper (Calmet et al., 2018) (which hypothesis is applied to respiratory system simulations), iLES can be considered in the same range as a

wavenumber than DNS. This hypothesis relays on wide importance because of the assumption particles transport will not require a diffusion model, as explained in section 5.5.1.

Chapter 5

Particles transport physics

If I could remember the names of these particles, I would have been a botanist.

E. Fermi

Summary: After solving the air unknowns, the moment has come to solve particle transport. There are many different approaches to solve particle transport, and these may depend on the application. Our goal is to simulate the respiratory system airways, but with the possibility of quickly being able to adapt it to any other purpose that may require particle transport. In this chapter, first the main compromises taken for a better performance of the code will be exposed (section 5.1). Next, the Lagrangian frame of reference chosen in this thesis will be compared to the Eulerian one (section 5.2). Finally, the required forces affecting particles in respiratory system simulations will be analyzed (section 5.3) within the importance of the relaxation time value (section 5.4) and the particles diffusion model used (section 5.5).

5.1 Particle transport approaches

The physics involved in particle transport can be split in two main instances: resolution of the fluid on the one hand and physics affecting particles on the other, always after the fluid is solved and variables such as its velocity and pressure (or temperature in more specific cases) are obtained.

In order to be efficient in the particle transport modelling some compromises must be adopted. The application of the model and a proper understanding of the literature and physics involving this application will mark which approach is ideal in each case. In this section, the main hypotheses assumed are briefly exposed.

- *Point particles.* Although radius, volume, or even shape, are taken into account for some calculus, point particles don't differentiate which part of them is submitted to the forces (e.g. an off-center force which might cause a pair force difference generating rotation). This hypothesis is valid when sufficiently small particles are computed compared to the flow streamlines isotropy at particle scale (Udwadia y Kalaba, 2007). External forces behavior are detailed in section 5.3.

- *Transient flows.* Transient flow allows facing a wider range of problems, including turbulent flows with very short time scales. Subgrid scales (SGS) are not taken into account for transporting particles as explained in subsection 5.5.1.

- *One-way coupling.* One-way coupling means that the fluid interacts with the particles, but the particles have sufficiently small momentum that does not interact with the fluid. This assumption makes some simulations, like combustion, unreliable when the fraction of particles is about the same order or superior than the fraction of liquid (Doisneau et al., 2013).

- *No mutual interactions.* Particles don't interact with each other, meaning they don't collide or present interaction forces. When collisions can be neglected is explained in section 5.5.2.

5.2 Frame of reference

There are two frames of reference in the literature to describe the particles transport: Eulerian and Lagrangian (Zhang y Cheng, 2007).

On the one hand, the first one solves a Partial Differential Equation (PDE), obtaining the concentration and velocity of particles. For "light" particles following the fluid velocity, this approach involves a PDE's per particle type to solve for the concentration. For "heavy" particles, the number of PDEs is then four, as one has to account for the particle velocity as well. On the other hand, the Lagrangian approach solves an Ordinary Differential Equation (ODE) for each individual particle.

The Eulerian approach is suitable and convenient to estimate detailed spatial and temporal distributions of particle concentrations and the particle residence time. Most recent numerical works of the Eulerian approach use the advection-diffusion equation with the gravitational settling to calculate particle concentrations (Zhao et al., 2004) and the particle residence time (Danckwerts, 1953; Ghirelli y Leckner, 2004).

An example of an application, without an optimal frame of reference selection, is icing on wings. Both frameworks, Lagrangian (Villedieu et al., 2012) and Eulerian (Habashi et al., 2006) are used in the literature to

Table 5.1: Differences between Lagrangian and Eulerian frameworks

	Eulerian	**Lagrangian**
Equations	Partial differential equations (PDEs). If pure transport: One PDE is solved. If external forces: Four PDEs are solved. Each type of particles (e.g. different diameters) require to solve PDEs separately.	Ordinary differential equations (ODEs). No difference in the number of equations for pure transport or external forces. Each particle is solved separately.
Injection	Global Boundary conditions	Local Individual for each particle
Parallelization	PDEs regular way parallelization	Load balance problem Migration between subdomains domain

simulate this. However, one of the studies of general interest is the shadow zone on the back of the wing, where there is a very low circulation of particles. Eulerian framework becomes the common option for this concentration purpose (Kim et al., 2013).

The Lagrangian approach considers a finite number of particles and track trajectories. This option enables representing dynamic characteristics of particles such as external forces, and accurately provide detailed spatial and temporal information of single particle trajectories and dispersion history (Li y Ahmadi, 1992). But at the same time, it makes it difficult to estimate the particle residence time. The major complication is that particles released at different locations and times inside a given domain would depart this domain at different times. This results in different residence times for individual particles (Chang et al., 2013). In addition, as explained for the icing example, in the cases where spatial statistics are important, injection must be done using many particles and many injections around the domain, highly increasing the computational cost.

Particles injections are solved in a different way. In the Eulerian approach it is done using the boundary conditions, which make more suitable global injections. However, the Lagrangian approach allows injecting locally using spatial coordinates.

The parallelization techniques for Eulerian and Lagrangian methods are completely different. The Eulerian approach parallelization is done in the regular way PDEs are parallelized, whereas the Lagrangian approach can become more critical because of the migration of particles between subdomains and load balance issues, when they are concentrated in specific regions (Houzeaux et al., 2016).

Finally, the Eulerian approach can take into account more easily sub–grid scales (SGS) calculated in the Gauss points of the element (Guerra et al., 2013). More detailed information about SGS will be given in subsection 5.5.1

To sum up, table 5.1 shows the main differences between both frameworks.

5.3 Forces involved

The predominant forces exerted by the fluid on the particles strongly depend on the problem to be solved. As we consider point particles and one-way coupling, an accurate force balance cannot be established and therefore the only option is to combine a series of forces devised ad-hoc into a total force $\boldsymbol{F}_p$.

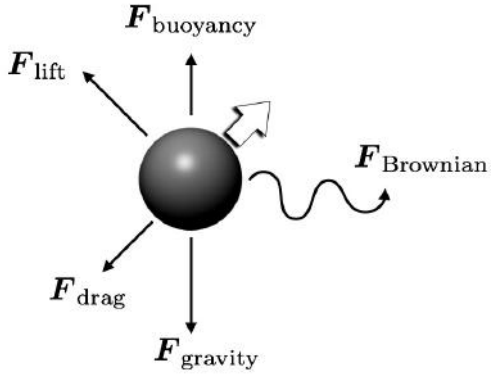


Figure 5.1: Picture of external forces affecting a single particle

A non-exhaustive list, considered in this thesis, is: drag force, lift force, gravity, buoyancy or Brownian diffusion.

The aforesaid forces will be considered in this thesis, but other forces, like the Van der Waals force or Magnus force, just to cite a few, could be necessary in other scenarios.

Here, let $\boldsymbol{a}_p$, $\boldsymbol{u}_p$, $\boldsymbol{x}_p$ and m_p be the acceleration, velocity, position and mass of particle p. According to Newton's second law:

$$\boldsymbol{F}_p = m_p \boldsymbol{a}_p = m_p \frac{d^2 \boldsymbol{x}_p}{dt^2}. \tag{5.1}$$

We are now going to describe the different forces considered in this thesis.

5.3.1 Drag force

This force represents the viscous and pressure forces exerted by the fluid and acts in the opposite direction of the relative velocity of the particle with the fluid. This force is defined as

$$|\boldsymbol{F}_D| = \frac{1}{2} \rho_f C_D A_p \|\boldsymbol{u}_f - \boldsymbol{u}_p\|^2, \tag{5.2}$$

where C_D and A_p are the drag coefficient and projection area of the particle in the direction of the motion. In the case of a sphere

$$A_p = \frac{\pi d_p^2}{4}, \tag{5.3}$$

being d_p the particle's diameter.

Dimensional analysis. Equation 5.2 can be rewritten as

$$\boldsymbol{F}_D = |\boldsymbol{F}_D| \frac{\boldsymbol{u}_f - \boldsymbol{u}_p}{\|\boldsymbol{u}_f - \boldsymbol{u}_p\|}. \tag{5.4}$$

Herein, the drag force variables involved are relative speed norm $\|\boldsymbol{u}_f - \boldsymbol{u}_p\|$, fluid density ρ_f, viscosity of the fluid μ_f, size of the body in function of its frontal area A_p and drag force $|\boldsymbol{F}_D|$. According to the algorithm of the Buckingham π theorem (Buckingham, 1914), these 5 parameters can be reduced to two dimensionless parameters:

- Drag coefficient C_D and
- Particle Reynolds number Re_p

For this purpose, we will consider a yet-unknown function f_a such as

$$f_a(|\boldsymbol{F}_D|, \|\boldsymbol{u}_f - \boldsymbol{u}_p\|, A_p, \rho_f, \mu_f) = 0. \tag{5.5}$$

The right-hand side is zero in any system of units. So it should be possible to express the relationship described by f_a in terms of only dimensionless groups.

First, Re_p reads

$$Re_p = \frac{\|\boldsymbol{u}_f - \boldsymbol{u}_p\| d_p}{\nu_f}, \tag{5.6}$$

where $\nu_f = \frac{\mu_f}{\rho_f}$ and is known as kinematic viscosity.

Next, C_D reads

$$C_D = \frac{|\boldsymbol{F}_D|}{\frac{1}{2}\rho_f A_p \|\boldsymbol{u}_f - \boldsymbol{u}_p\|^2}. \tag{5.7}$$

Now, f_a can be replaced by f_b such as

$$f_b\left(\frac{|\boldsymbol{F}_D|}{\frac{1}{2}\rho_f A_p \|\boldsymbol{u}_f - \boldsymbol{u}_p\|^2}, \frac{\rho_f \|\boldsymbol{u}_f - \boldsymbol{u}_p\| \sqrt{A_p}}{\mu_f}\right) = 0. \tag{5.8}$$

Herein, the only unknown is $\boldsymbol{F}_D$, it is possible to express it as

$$\frac{|\boldsymbol{F}_D|}{\frac{1}{2}\rho_f A_p \|\boldsymbol{u}_f - \boldsymbol{u}_p\|^2} = f_c\left(\frac{\rho_f \|\boldsymbol{u}_f - \boldsymbol{u}_p\| \sqrt{A_p}}{\mu_f}\right), \tag{5.9}$$

or

$$|\boldsymbol{F}_D| = \frac{1}{2}\rho_f A_p \|\boldsymbol{u}_f - \boldsymbol{u}_p\|^2 f_c(Re_p), \tag{5.10}$$

and with

$$C_D = f_c(Re_p). \tag{5.11}$$

5.3.1.1 Drag coefficient

From the previous dimensional analysis, we know that the drag coefficient C_D must depend on the particle Reynolds number Re_p. For the motion of a smooth sphere in the Stokes regime, which can be described as a type of fluid flow where advective inertial forces are small compared to viscous forces and $Re_p \ll 1$, the drag coefficient can be easily formulated, by neglecting the effects of the inertia terms of Navier-Stokes equations 4.1-4.2 (Yang et al., 2015), as

$$C_D = \frac{24}{Re_p}, \tag{5.12}$$

which is valid for most of the particle transport simulations in respiratory system simulations as far as particles tend to align with fluid. Obviously, when particles follow the flow stream lines, its relativity velocity norm $\|\boldsymbol{u}_f - \boldsymbol{u}_p\|$ becomes small and consequently particles Reynolds too.

However, this is not valid for higher Reynolds numbers. Literature usually differentiates four different regions for the type of fluid flow: the aforesaid Stokes regime where drag force decreases inversely proportional to Reynolds number as shown in equation 5.12, Newton's law regime ($10^3 < Re_p < 10^5$), where drag coefficient is independent of Reynolds number ($C_D \simeq 0.44$), the intermediate region between Stokes and Newton regions and the boundary layer separation at a very high Reynolds number ($Re_p > 10^5$) (Norouzi et al., 2016). Different drag coefficients are proposed in the literature (Ganser, 1993; Cheng y Nguyen, 2010; Turton y Levenspiel, 1986; Wilson y Huang, 1979; Arastoopour et al., 1982) as shown in figure 5.2 in order to be able to evaluate the drag coefficient throughout the main three regions where this thesis models particles transport. Allow us to note that figure 5.2 does not show the drag crisis that is found in a Reyndols number between 10^5 and 10^6, but in the respiratory system such a large Reynolds number is not found.

In this thesis, Ganser's formulation will be the chosen one. Based on the equation 5.6 in (Ganser, 1993), the author defines drag coefficient as

$$C_D = \frac{24}{Re_p K_1}[1 + 0.1118(Re_p K_1 K_2)^{0.65657}] + \frac{0.4305 K_2}{1 + \frac{3305}{Re_p K_1 K_2}}, \tag{5.13}$$

where K_1 and K_2 are the shape functions that define how spherical the particle is, and are defined as

$$K_1 = \frac{1}{3} + \frac{2}{3\sqrt{\phi}}, \tag{5.14}$$

and

$$K_2 = 10^{1.8148(-log\phi)^{0.5743}}, \tag{5.15}$$

being ϕ the spherification value. If particles are spheres, then $\phi = 1 \rightarrow K_1 = K_2 = 1$. In figure 5.3 can be appreciated how C_D varies in function

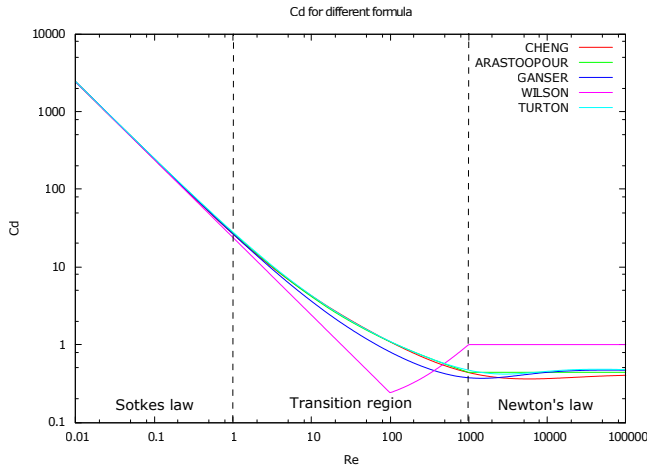


Figure 5.2: Comparison of different drag coefficients proposed in the literature (Ganser, 1993; Cheng y Nguyen, 2010; Turton y Levenspiel, 1986; Wilson y Huang, 1979; Arastoopour et al., 1982)

of particles shape showing that, at a fixed Reyndols number, drag increases with decreasing ϕ.

It is important to remark that equation 5.13 is valid for

$$Re_p K_1 K_2 \leq 10^5. \tag{5.16}$$

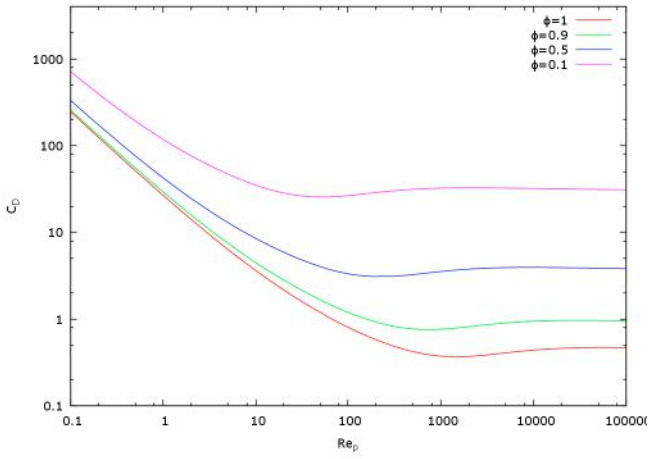


Figure 5.3: Ganser's drag coefficient in function of shapes values

5.3.1.2 Cunningham slip correction factor

Inside the drag formula, 5.2, can also be added as a correction for noncontinuum effects on small particles, with the Cunningham slip correction factor C_{slip},

$$C_{slip} = 1 + \frac{2\lambda_{MFP}}{d_p}(1.142 + 0.558e^{-0.999\frac{d_p}{2\lambda}}), \tag{5.17}$$

where λ_{MFP} is the mean free path of the air, at room temperature $6.64{\cdot}10^{-8}$.

The Cunningham slip correction factor allows predicting the drag force on a particle moving inside a fluid with Knudsen number (Kn) between the continuum regime and free molecular flow. The Knudsen number is defined as

$$Kn = \frac{\lambda_{MFP}}{L}, \tag{5.18}$$

where L is the representative physical length scale (typically the diameter d_p in the case of particles transport).

Particle Knudsen number Kn estimates the slip effect and classifies the different regimes as (Neumann y Rohrmann, 2012):

- Continuum: $Kn < 0.015$,
- Slip flow: $0.015 < Kn < 0.15$,
- Transitional: $0.15 < Kn < 4.5$,
- Free molecule: $Kn > 4.5$.

The mathematical reason why this correction factor becomes necessary is that the derivation of Stokes Law assumes a nonslip condition which is no longer correct at high Kn.

In this thesis we will find our simulations between slip flow (microparticles) and a free molecule (nanoparticles) regime.

5.3.1.3 Drag force overview

It is possible to rewrite the drag force equation 5.2 in function of d_p and Re_p, which are commonly significative variables in the simulations in this thesis, and also including C_{slip} correction. Finally obtaining

$$\boldsymbol{F}_D = \frac{1}{8}\pi\mu_f d_p C_{slip} C_D Re_p(\boldsymbol{u}_f - \boldsymbol{u}_p). \tag{5.19}$$

5.3.2 Lift force

The lift force is normal to the drag force and represents non-symmetric effects not accounted by the drag force: non-uniform flow, non-spherical particles, particle rotation, etc. According to (Drew y Lahey, 1993) for a low Reynolds number flow past a spherical particle, the lift coefficient can be defined as

$$\boldsymbol{F}_{lift} = -C_l \rho_f \alpha_p (\boldsymbol{u}_f - \boldsymbol{u}_p) \times (\nabla \times \boldsymbol{u}_p), \tag{5.20}$$

where C_l means lift coefficient and the secondary phase volume fraction is

$$\alpha_p = \frac{\dot{\gamma} d_p}{2\|\boldsymbol{u}_f - \boldsymbol{u}_p\|}, \tag{5.21}$$

whereas the shear rate is

$$\dot{\gamma} = \frac{d\boldsymbol{u}_f}{dt}. \tag{5.22}$$

Furthermore, the lift coefficient can be defined in function of vorticity Reynolds number Re_ω, similarly to drag coefficient in previous subsection 5.3.1 as $C_l(Re_\omega)$. In this case, the dimensionless number is given by

$$Re_\omega = \frac{\|\nabla \times \boldsymbol{u}_f\| d_p^2}{\nu_f}, \tag{5.23}$$

In general, the lift force remains small for nanoparticles, being possible to neglect it in this case, but becomes significant for microparticles (Li y Ahmadi, 1992).

Saffman lift coefficient. According to (Saffman, 1965), for low Reynolds number flow past a spherical particle, the lift coefficient can be defined as

$$C_l = \frac{3}{2\pi\sqrt{Re_\omega}} C, \tag{5.24}$$

for $0 \leq Re_p \leq Re_\omega \leq 1$ and $C = 6.46$.

Mei lift coefficient. In (Mei et al., 1994), the authors extended the model to a higher range of particle Reynolds numbers. The Saffman-Mei model is empirically represented as

$$C = \begin{cases} 6.46 f(Re_p, Re_\omega) & if \quad Re_p \leq 40 \\ , & \\ 6.46 \cdot 0.0524\sqrt{\beta Re_p} & if \quad 40 < Re_p < 100, \end{cases} \tag{5.25}$$

where

$$\beta = \frac{Re_\omega}{2Re_p}, \tag{5.26}$$

and

$$f(Re_p, Re_\omega) = (1 - 0.3314\sqrt{\beta}) e^{-0.1 Re_p} + 0.3314\sqrt{\beta}. \tag{5.27}$$

Lift force near wall. When a particle is placed close enough to a wall, it is strongly influenced by lift force due to near wall shear. According to (Zheng y Silber-Li, 2009), the lift force is dominant at a range of $2 < z^+ < 6$, where the dimensionless distances z^+ are defined as

$$z^+ = z/d_p, \tag{5.28}$$

Here, z represents the distance from the wall. This force can strongly affect the deposition, underestimating it if it is not taken into account (Li y Ahmadi, 1992).

5.3.3 Gravity and buoyancy

These forces act together as soon as there exist a density difference between the fluid and the particle, which can be represented as

$$\boldsymbol{F}_g + \boldsymbol{F}_b = \rho_f \boldsymbol{g} V_p (1 - \frac{\rho_f}{\rho_p}), \tag{5.29}$$

where $\boldsymbol{g}$ is the gravity vector where the norm is $9.81 m/s$, whereas V_p is the volume of the particle.

5.3.4 Brownian motion

Molecules of the fluid constantly collide against particles. When particles are small enough, the dispersion provoked by this phenomenon may be noticeable and it is known as Brownian motion. From the Eulerian point of view, this force would appear as a diffusion term in the equation. In the Lagrangian framework it can be applied as a random displacement or as a random force.

This phenomena was first observed in 1827 by botanist Robert Brown and explained by Albert Einstein (A.Einstein, 1903), calculating the diffusion by using a normal distribution with the mean $\mu = 0$ and variance $\sigma^2 = 2Dt$, where D and t are diffusivity and time. Hence, the expression of the diffusion over time can be described as

$$\rho(x,t) = \frac{N}{\sqrt{4\pi D t}} e^{-\frac{x^2}{4Dt}}, \tag{5.30}$$

where ρ, x, t, N and D are density of particles, position, time, number of particles and diffusivity respectively.

The generation of non-correlated random numbers with mean zero is crucial in order to simulate Brownian motion correctly. This is why, Brownian motion implementation using Parallel Pseudo Random Numbers Generator (PPRNG) will be expanded in an extended section 7.4 about Brownian motion, to the extent this topic requires special attention.

5.4 Particles relaxation time and Stokes number

5.4.0.1 Relaxation time

Particle relaxation time τ_p is a measure of the return time of a perturbed system to equilibrium. Hence, in the case of transported particles, it means the time each particle needs to react to flow until equilibrium between both is reached. Thus, assuming particle transport is mainly caused by drag force, and given

$$\boldsymbol{a}_p = \frac{d\boldsymbol{u}_p}{dt} = \frac{\boldsymbol{F}_D}{m_p}, \tag{5.31}$$

where

$$m_p = \rho_p V_p, \tag{5.32}$$

while $V_p = \frac{1}{6}\pi d_p^3$ using equation 5.19. Now, acceleration can be rewritten as

$$\frac{d\boldsymbol{u}_p}{dt} = \frac{\boldsymbol{u}_f - \boldsymbol{u}_p}{\tau_p}, \tag{5.33}$$

where

$$\tau_p = \frac{\alpha_D}{C_D Re_p}, \tag{5.34}$$

and

$$\alpha_D = \frac{4}{3}\frac{\rho_p d_p^2 C_{slip}}{\mu_f}. \tag{5.35}$$

However, it is common to find in the literature τ_p described in the approximation of considering a Stokes flow ($Re_p \ll 1$) for which $C_D = \frac{24}{Re_p}$, in that case, Stokes relaxation time reads

$$\tau_p^{St} = \frac{C_{slip}\rho_p d_p^2}{18\mu_f}. \tag{5.36}$$

A small example for checking particle behavior in function of its own τ_p is proposed. In order to simplify, we will consider on the one hand $\boldsymbol{u}_f = 0$. Given that equation 5.33 is written in function of time such as

$$\boldsymbol{u}_p(t) = \boldsymbol{u}_p^0 e^{-\frac{t}{\tau_p}}, \tag{5.37}$$

where $\boldsymbol{u}_p^0$ is the initial particle velocity.

On the other hand, we consider three representative types of particles in this thesis: a nanoparticle, a microparticle and a tennis ball size particle, just to give an idea of human-scale behavior. Particles have water density $\rho_p = 10^3 kg/m^3$, while tennis ball has an approximately density of $\rho_p = 450 kg/m^3$ and its diameter is about $d_{tennisball} = 6.8cm$. Thus, these relaxation times are:

- Nanoparticles: $\tau_p = 1 \cdot 10^{-9} s$.
- Microparticles: $\tau_p = 4 \cdot 10^{-6} s$.
- Tennis ball: $\tau_p = 6300 s$.

Whereas initial particle velocity is set to $\left|\boldsymbol{u}_p^0\right| = 1 m/s$ and fluid remains in rest. It is important to notice the importance of the Cunningham correction factor in nanoparticles. While in micropartciles the factor is close to $C_{slip} \simeq 1$, in the nanoparticle case $C_{slip} \simeq 225$.

This way, as shown in figure 5.4, for smaller particles, there is a quicker particle velocity decay until reaching fluid velocity, in this case zero. On a more human scale like the tennis ball the decay time is of thousands of seconds, but for nanoscale it is extremely quick. In other words, the smaller the particle becomes, smaller time steps will be required in the simulation in order to capture all the physics affecting them.

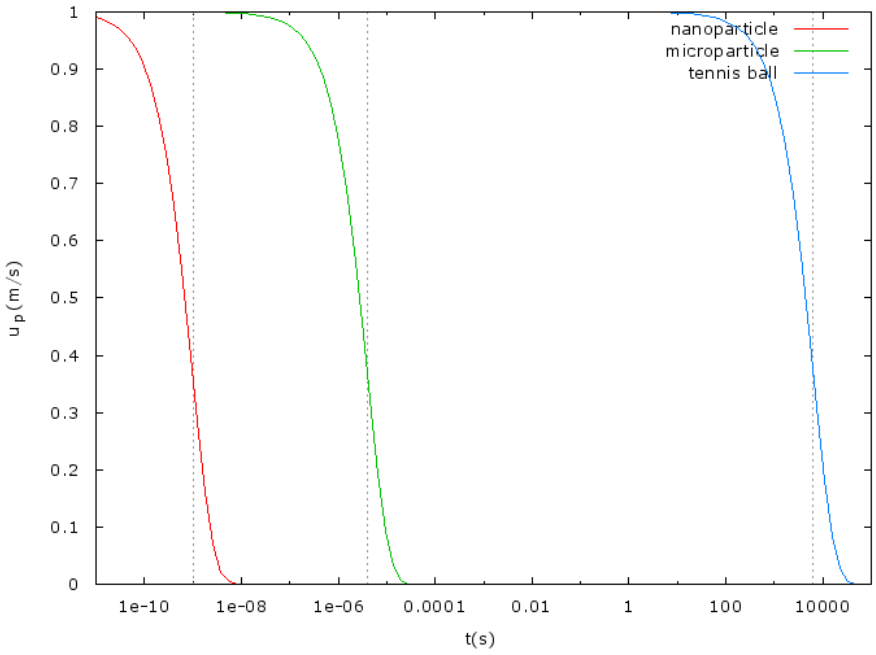


Figure 5.4: Comparison of decay time with different τ_p. Vertical discontinuous lines represent the particular relaxation time.

In future section 6.3.2 a real case simulation will be presented. Here, being able to capture the decay time becomes absolutely necessary in order to obtain accurate results.

5.4.0.2 Stokes number

Use of the dimensionless Stokes number St is often found in literature to characterize the behavior of particles suspended in a fluid flow. For example, it is used to describe the deposition in a bend in function of St in (Pui et al., 1987), a benchmark this thesis uses to validate our results in chapters 6 and 8. A particle with a low Stokes number follows fluid streamlines (perfect advection), while a particle with a large Stokes number is dominated by its inertia and tends to continue along its initial trajectory.

The Stokes number is defined as the ratio of the characteristic time of a particle (or droplet) to a characteristic time of the flow or of an obstacle. In other words, it is the ratio of particle relaxation time to fluid time scale (Kulick et al., 1994). Again, defining the characteristic dimension of the obstacle L, we can describe St as

$$St = \frac{\tau_p \|\boldsymbol{u}_f\|}{L}. \tag{5.38}$$

For a better understanding of what the Stokes number means, let us consider a constant fluid velocity $\|\hat{\boldsymbol{u}}_f\|$. Now, a time scale can be described used as a characteristic length particle diameter such as $\hat{t} = \frac{d_p}{\|\hat{\boldsymbol{u}}_f\|}$. Next, let us consider all the variables involved in a drag differential equation 5.33 as $\|\boldsymbol{u}_p\| = \|\hat{\boldsymbol{u}}_f\|\|\boldsymbol{u}_p^*\|$, $\|\boldsymbol{u}_f\| = \|\hat{\boldsymbol{u}}_f\|\|\boldsymbol{u}_f^*\|$, $t = \hat{t}t^*$ and $\tau = \hat{\tau}\tau^*$, where $\hat{\tau}$ can be considered the relaxation time in a Stokes regime as shown on equation 5.36. Hence,

$$\frac{\hat{\boldsymbol{u}}_f}{\hat{t}} \frac{d\boldsymbol{u}_p^*}{dt^*} = \frac{\hat{\boldsymbol{u}}_f}{\hat{\tau}\tau^*}(\boldsymbol{u}_f^* - \boldsymbol{u}_p^*), \tag{5.39}$$

$$\frac{d\boldsymbol{u}_p^*}{dt^*} = \frac{1}{St} \frac{\boldsymbol{u}_f^* - \boldsymbol{u}_p^*}{\tau^*}, \tag{5.40}$$

where $St = \frac{d_p}{\|\hat{\boldsymbol{u}}_f\|\hat{\tau}}$, which finally can be defined as

$$St = \frac{C_{slip}\rho_p d_p \|\boldsymbol{u}_f\|}{18\mu_f}, \tag{5.41}$$

which is the same result we would obtain mixing equation 5.36 and 5.38 using $L = d_p$. Checking equation 5.40, it can be noticed that for small Stokes numbers (e.g. small diameter or low fluid velocity) the variation of $\boldsymbol{u}_p^*$ will become important, whereas for large Stokes numbers, it will tend to zero, meaning we are not in the Stokes regime and thus this is not valid.

5.5 Particles diffusion modeling

When LES or RANS models are used, literature proposes different approximations to simulate particle diffusion due to turbulence employing

stochastic methodologies. From the least computationally intensive to the most computationally intensive, according to (Loth, 2000), these models can be arranged as: Discontinuous random walk DRW models, continuous random walk CRW models, and Langevin stochastic differential equation SDE models.

On the one hand, DRW and CRW models both require a drift correction in non-homogenous flows. On the other, SDE models do not need such a correction because their first-moment is equal to the Eulerian momentum equation (MacInnes y Bracco, 1992). However, the SDE model demands the Reynolds-stress transport description of the turbulence and according to (Shirolkar et al., 1996; Loth, 2000), it probably has not been tested as extensively as the other models.

For the DRW Lagrangian simulations, the turbulence is assumed to be isotropic and the fluctuation a Gaussian probability distribution (Gosman y Loannides, 1983). It means the eddy is assumed to have a constant velocity perturbation for as long as the particle is interacting with it. This perturbation is formed in the subgrid scale by using a randomly chosen velocity perturbation $\boldsymbol{u}'_f$ combined with the local mean (or resolved) velocity immediately surrounding the fluid point location $\bar{\boldsymbol{u}}_f$. Considering equation 4.5, it is carried out

$$\boldsymbol{u}_f^{DRW} = \boldsymbol{u}'_f + \bar{\boldsymbol{u}}_f. \tag{5.42}$$

DRW has been widely used in engineering problems (Elghobashi, 1994; Crowe et al., 2011) with a good performance for complex flows. Nevertheless, one of the main problems with the DRW model is that it employs step-function type perturbations yielding infinite continuous-fluid accelerations (Loth, 2000). In addition, as mentioned above, we must assume isotropic turbulence. The CRW model avoids these problems by correlating the turbulence statistics in time with stochastic sampling for obtaining finite fluid accelerations (MacInnes y Bracco, 1992).

The eddy lifetime τ_Λ and the particle-eddy interaction time τ_{int}, which can be calculated as shown in equation 5.43, defined in (Csanady, 1963) for turbulent flows.

$$\frac{1}{\tau_{int}} = \frac{1}{\tau_\Lambda} + \frac{1}{\tau_{tra}}. \tag{5.43}$$

Where τ_{tra} means the time particles need to traverse the eddy. Hence, the continuous-fluid acceleration along the fluid path can be approximated by CRW model as

$$\frac{d\boldsymbol{u}_f^{CRW}}{dt} \simeq \frac{\tau_{int}}{\tau_\Lambda}\frac{\boldsymbol{u}'_f(t+\delta t) - \boldsymbol{u}'_f(t)}{\delta t} + \frac{\bar{\boldsymbol{u}}_f[x_p(t+\delta t)] - \bar{\boldsymbol{u}}_f[x_p(\delta t)]}{\delta t}. \tag{5.44}$$

When particle density is similar to the fluid density (e.g. bubbles), CRW enebles computing vorticity fluctuations, which can be used for lift

formulation. DRW, however, is often used under the opposite condition, when particles density is much greater than a fluid one $\frac{\rho_p}{\rho_f} \gg 1$.

Applying DRW and CRW models will remain as a future work. During this thesis it will be assumed that iLES can be assimilated to a DNS solution (as explained in previous section 4.2), not requiring as a consequence any dissipation model. The inclusion of CRW models in iLES applied to particles transport can be of special interest for studying how it can affect the calculation of the vorticity on particles and, consequently, the behavior of lift force.

5.5.1 Element interpolation at particle position

The fluid velocity is interpolated element-wise, once the element and the associated shape functions are obtained at any particle location. In this thesis, the sub-grid scale SGS velocity is not considered in the drag law. In VMS methods, the SGS is generally discontinuous and tracked at the element integration points, which makes its interpolation difficult at any location inside an element. Figure 5.5 illustrates the different locations where the unknowns are obtained.

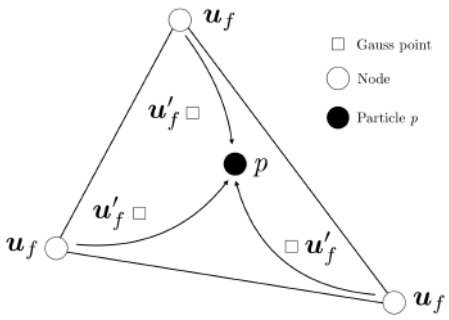


Figure 5.5: Linear interpolation of velocity at particle p position using nodal velocity u_f^h. Subgrid scale velocity $\boldsymbol{u}'_f$ computed at the Gauss point is neglected.

Several options are possible. On the one hand, to obtain a continuous SGS, the SGS is first projected on the finite element space and then interpolated at the particle position. On the other hand, to maintain the discontinuous character of the SGS, this one can be extrapolated from the Gauss points to the nodes and then interpolated at the particle position. When considering an Eulerian approach however, the SGS velocity can easily be accounted for in the advection term, as described in (Guerra et al., 2013).

5.5.2 Negligible collision criteria

During this thesis the collision between particles will be neglected and it will remain for future work to implement it. The criteria to neglect them must depend on the concentration of particles $n_p = \frac{\#particles}{V}$, the size of particles d_p, the particle velocity $\boldsymbol{u}_p$ and the turbulence of the fluid or its time scale τ_Λ, or the equivalent to the particle transition inside the eddy. In the previous subsection 5.5, the particle-eddy interaction time τ_{int} was described (check equation 5.43) for the case of turbulent flow. In the case of laminar flow, this can be replaced by a macroscopic time scale τ_L based on a characteristic length L. Deeper discussion about the characteristic length will be discussed in the next section 6 for applying it to an adaptive time step scheme.

According to (Loth, 2000), the number of collisions per unit time and per particle can be approximated by three main factors:

- 1) The product of n_p.
- 2) The inter-particle relative velocity
- 3) The area swept out by a single particle.

The inter-particle relative velocity in turbulence far from initial conditions can be upper-bounded by the particle terminal velocity (V_{term}), while the swept area, to which particle to particle interactions are limited, can be estimated by the value πd_p^2.

According to these mentioned factors, the criteria for negligible particle collision in turbulence can be written as

$$n_p \|\boldsymbol{u}_p^{ter}\| \pi d_p^2 \ll \frac{1}{max(\tau_{int}, \tau_p)}, \tag{5.45}$$

where the terminal velocity of the particle reads

$$\boldsymbol{u}_p^{ter} = \sqrt{\frac{2m_p \boldsymbol{g}}{\rho_p A C_D}}, \tag{5.46}$$

thus, for a sphere particle

$$\boldsymbol{u}_p^{ter} = \sqrt{\frac{4d_p}{3C_D}}. \tag{5.47}$$

In a Lagrangian approach, n_p is determined by the number of particles injected, which can vary depending on the requirements of the problem. More particles imply more computational cost, so frequently the number of particles injected are in a balance between being significative for the statistics but with an affordable computational cost. To include some

typical numbers in our simulations, allow us to suppose an initial injection of 100000 microparticles in a sphere of $0.03m^3$ evolving a human nose, we have $n_p \simeq 3000000$ particles per m^3. As mentioned on subsection 5.4.0.1, microparticles have a $\tau_p = 4 \cdot 10^{-6}s$, thus $\frac{1}{\tau_p} = 250000$ while $n_p \pi d_p^2 \simeq 10^{-5}$, so a very large $\|\boldsymbol{u}_p^{ter}\|$ would be need in order not to obey criteria 5.45, which is extremely unlikely in respiratory system simulations.

Chapter 6

Particles transport numerical model

To those who ask what the infinitely small quantity in mathematics is, we answer that it is actually zero. Hence there are not so many mysteries hidden in this concept as they are usually believed to be.

L. Euler

Summary: In previous chapter 5, the physics affecting particle transport was exposed, defining the external forces. However, these forces act along time, making necessary the integration of the position, velocity and acceleration over time to obtain the particle trajectory at every moment. In this chapter we discuss what types of integration schemes exist in the literature and which is the most suitable for our application (section 6.1). In addition, an adaptive time step strategy is included to help to control the error and reach convergence 6.2. To conclude, results are first presented with simple mathematical cases and later using physical benchmarks.

6.1 Time integration schemes

There is a large variety of integration schemes for solving the particle transport. Some examples, starting with single step methods, are Euler's method or the midpoint rule, both compared with a particle tracking application in (Teitzel et al., 1997). However, these kinds of methods are not very accurate and are difficult to converge when the particles that have

accelerations that are too large. One of the most common schemes found in the literature are Runge–Kutta methods (RK_p), where p denotes the order of the integration scheme. Authors like (Boozer y Kuo-Petravic, 1981; Press et al., 1992) use this method for particle transport and others like (Darquenne y Paiva, 1994) use it, more particularly, for aerosol transport in human lungs. But some problems are also reported when RK_p are used for small particles (e.g. nanoparticles), as shown by (Longest y Xi, 2007). These same authors use a Runge–Kutta, except for particles with $d_p < 400nm$, in which case an analytic integration scheme is employed. This is why we opt for a more adaptable integration scheme: A Newmark-β integration scheme.

6.1.1 Newmark-β time integration scheme

The Newmark-β is a semi-implicit integration method primarily developed for the numerical evaluation of the dynamic response of structure in finite element analysis Newmark (1959); Hilber et al. (1977); Wood et al. (1981); Chung y Hulbert (1993). Later, these methods have also been applied to first order differential equations Jansen et al. (2000), Schweizerhof et al. (2004b). We consider a Newton-Raphson NR iterator for solving the implicit dependence, first proposed by Park (1975).

In the Newmark-β scheme, the actualizations of the velocity $\boldsymbol{u}_p^{n+1}$ and position $\boldsymbol{x}_p^{n+1}$ of the next time step are given by two equations:

$$\boldsymbol{u}_p^{n+1} = \boldsymbol{u}_p^n + [(1-\gamma)\boldsymbol{a}_p^n + \gamma \boldsymbol{a}_p^{n+1}]\delta t, \tag{6.1}$$

$$\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \boldsymbol{u}_p^n \delta t + \frac{\delta t^2}{2}[(1-2\beta)\boldsymbol{a}_p^n + 2\beta \boldsymbol{a}_p^{n+1}], \tag{6.2}$$

where β and γ are constants which determine if the method is implicit or explicit, its accuracy and its stability. For $\gamma = \frac{1}{2}$, the method is at least second-order accurate. While $\beta = \frac{1}{4}$ yields the constant average acceleration method.

6.1.1.1 Linearization

Some forces depend on the particle and fluid velocities (e.g. drag force or lift force) and thus the position. Equation (6.1) is therefore strongly non-linear. In this work, an inner Newton-Raphson (NR) iterator is used to converge the particle velocity equation. Once the velocity is obtained, the position is then updated using Equation (6.2). The NR method can be defined as a root-finding algorithm that uses the first term of the Taylor series of a function in the vicinity of a suspected root.

Let $\boldsymbol{f}(\boldsymbol{u}_p^{n+1})$ be the function whose root is desired using equation (6.1)

$$\boldsymbol{f}(\boldsymbol{u}_p^{n+1}) = -\boldsymbol{u}_p^{n+1} + \boldsymbol{u}_p^n + [(1-\gamma)\boldsymbol{a}_p^n + \gamma \boldsymbol{a}_p^{n+1}]\delta t, \tag{6.3}$$

and the Jacobian $\boldsymbol{J}$ defined as

$$\boldsymbol{J} = \nabla_{\boldsymbol{u}_p} \boldsymbol{f}(\boldsymbol{u}_p^{n+1}) = -\boldsymbol{I} + \gamma \frac{d\boldsymbol{a}_p}{d\boldsymbol{u}_p}\bigg|_{n+1} \delta t, \tag{6.4}$$

where $\boldsymbol{I}$ is the identity matrix.

Finally, the Newton-Raphson is described in this case by

$$\boldsymbol{u}_p^{n+1,k+1} = \boldsymbol{u}_p^{n+1,k} - \boldsymbol{w}(\boldsymbol{u}_p^{n+1,k}), \tag{6.5}$$

where the subindex k means the k-iteration of the NR and

$$\boldsymbol{w}(\boldsymbol{u}_p^{n+1,k}) = \boldsymbol{J}^{-1} \boldsymbol{f}(\boldsymbol{u}_p^{n+1,k}). \tag{6.6}$$

6.1.1.2 Convergence criterion

Next convergence criterion defined as

$$\frac{\|\boldsymbol{w}(\boldsymbol{u}_p^{n+1,k})\|}{\|\boldsymbol{u}_p^{n+1,k}\|} < \epsilon_c, \tag{6.7}$$

is imposed. ϵ_c is defined as the tolerance residual norm, or in other words, the desired precision in the convergence.

The integration scheme can be summarized in algorithm 1.

Algorithm 1 Integration scheme algorithm.

```
for each particle p do
    Interpolate from nodes u_f(x_p^n)
    ε_c = 10^6; k = 0; maxiter = 10
    while (||w||/||u|| > ε_c) and (k ≤ maxiter) do
        Sum all forces
        Second Newton's law: a_p^{n+1,k} = F_p^{n+1,k}/m_p
        Compute u_p^{n+1,k} using Newmark-β + Newton-Rapshon
        Calculate ε_c
        k = k + 1
    end while
    Update position x_p^{n+1}
end for
```

Algorithm 1 includes the heaviest computational cost of the Lagrangian particle transport calculation and it is easily parallelizable splitting it into threads and processes in a more global way as shown in co-authored paper (Houzeaux et al., 2016) and in more detail in section 7.2.1.

6.1.1.3 Jacobian calculus

It is important to study the behavior of our Jacobian $\boldsymbol{J}$, as far as in the case of its determinant is equal to zero, the Jacobian will not be invertible and hence the NR method is not applicable. Apart from this, inverting the Jacobian can have a high computational cost when many different forces are involved. This is why first we are going to calculate the exact Jacobian if only drag force is taken into account $\boldsymbol{J}^D$. Next, we will explain our Jacobian approximation $\hat{\boldsymbol{J}}$ for more costly cases.

Drag force exact Jacobian. We will consider the acceleration $\boldsymbol{a}_p$ depends uniquely on drag force, previously defined by equation 5.2 in section 5. In that section, the acceleration of a particle was also defined when drag force was the only external force in function of relaxation time τ_p as

$$\boldsymbol{a}_p = \frac{(\boldsymbol{u}_f - \boldsymbol{u}_p)}{\tau_p}, \tag{6.8}$$

where τ_p depends on Re_p and C_D, as described by equations 5.34 and 5.35.

According to this, the Jacobian components J^D_{ij} (only using drag force) can be calculated. In order to simplify the formulation, superindex indicating the time step n and $n+1$ will be removed in this calculus, as far, as n-time step variables are removed when we derivate respect $\boldsymbol{u}_p^{n+1}$. Thus, the components are

$$J^D_{ij} = \frac{\partial f_i}{\partial u_{p,j}} = \frac{\partial}{\partial u_{p,j}}[-u_{p,i} + \gamma(u_{f,i} - u_{p,i})\frac{\delta t}{\tau_p}], \tag{6.9}$$

and applying derivative's chain rule, we obtain

$$J^D_{ij} = -(1+\gamma\alpha_D^{-1}C_D Re_p \delta t)\delta_{ij} + \gamma\alpha_D^{-1}\delta t \frac{\partial(Re_p C_D)}{\partial Re_p}\frac{\partial Re_p}{\partial u_{p,j}}(u_{f,i} - u_{p,i}), \tag{6.10}$$

where

$$\delta_{ij} = \begin{cases} 1, \text{ if } i = j, \\ \\ 0, \text{ if } i \neq j. \end{cases} \tag{6.11}$$

In order to remove some constants found in equation 6.10, let us define κ as

$$\kappa = \gamma\frac{\delta t}{\tau_p}, \tag{6.12}$$

where $\kappa > 0$ and ideally $\kappa \leq 1$ as far as δt is expected to be lower than τ_p if the reaction of the particle requires to be measured.

Next, derivatives are developed, such as

$$\frac{\partial Re_p}{\partial u_{p,j}} = -\frac{\Delta u_j}{\|\Delta \boldsymbol{u}\|^2} Re_p, \tag{6.13}$$

where $\Delta \boldsymbol{u} = \boldsymbol{u}_f - \boldsymbol{u}_p$, while Δu_j means the difference only in the component j. The $\frac{\partial(Re_p C_D)}{\partial Re_p}$ derivative, on the other hand, depends on the different possible formulas available in the literature, as explained in detail in section 5.3.1. In aforesaid section, it is shown in figure 5.2 that C_D is always a decreasing function (except in (Wilson y Huang, 1979) formula for very high Reynolds numbers). Again, for simplicity reasons, let us define ϕ as

$$\phi = \frac{1}{C_D} \frac{\partial(Re_p C_D)}{\partial Re_p}, \tag{6.14}$$

or if preferred

$$\phi = Re_p \frac{\partial}{\partial Re_p} \ln(Re_p C_D), \tag{6.15}$$

which shows that ϕ is a function containing the derivative of ln which can never become zero.

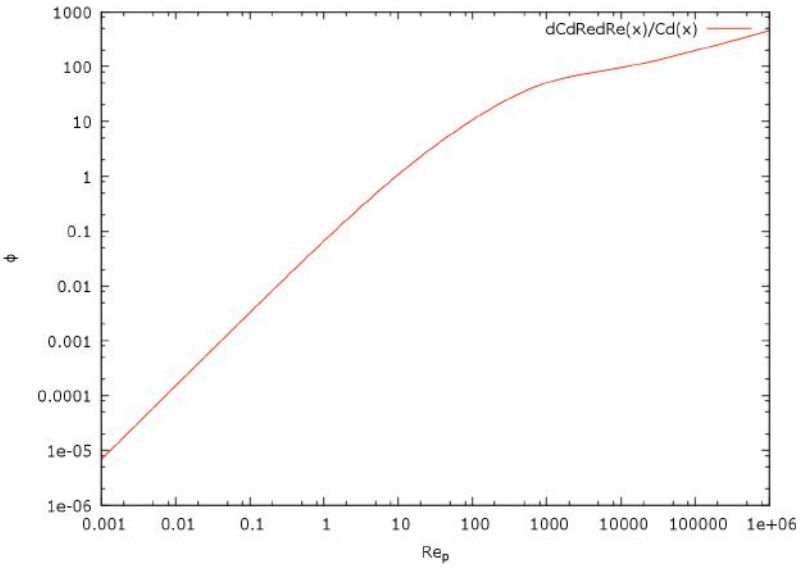


Figure 6.1: Behavior of ϕ

As shown in figure 6.1, where ϕ is plotted using Ganser's drag coefficient (Ganser, 1993), this function is always positive and increasing.

Finally, a shorter expression of the Jacobian can be written as

$$J_{ij}^D = -(1+\kappa)\delta_{ij} - \kappa\phi \frac{\Delta u_i \Delta u_j}{\|\Delta \boldsymbol{u}\|^2}. \tag{6.16}$$

If the determinant of the Jacobian is different from zero $\det(\boldsymbol{J}^D) \neq 0$, it is invertible. In order to check its behavior, let us consider a 2D problem.

In this case, the diagonal is defined by

$$J^D_{1,1} = -(1+\kappa) - \kappa\phi\frac{\Delta u_1^2}{\|\Delta \boldsymbol{u}\|^2}, \tag{6.17}$$

$$J^D_{2,2} = -(1+\kappa) - \kappa\phi\frac{\Delta u_2^2}{\|\Delta \boldsymbol{u}\|^2}, \tag{6.18}$$

whereas the extra-diagonal is given by

$$J^D_{1,2} = J^D_{2,1} = -\kappa\phi\frac{\Delta u_1 \Delta u_2}{\|\Delta \boldsymbol{u}\|^2}. \tag{6.19}$$

The Jacobian is thus symmetric, as will happen in the 3D case.

Keeping on the 2D problem, the determinant of the Jacobian can be calculated as

$$\begin{aligned}\det(\boldsymbol{J}^D) = (1+\kappa)^2 + (1+\kappa)\kappa\phi\frac{\Delta u_1^2 + \Delta u_2^2}{\|\Delta \boldsymbol{u}\|^2} \\ +\kappa^2\phi^2\frac{\Delta u_1^2 \Delta u_2^2}{\|\Delta \boldsymbol{u}\|^4} - \kappa^2\phi^2\frac{\Delta u_1^2 \Delta u_2^2}{\|\Delta \boldsymbol{u}\|^4},\end{aligned} \tag{6.20}$$

so after simplifications are applied, the determinant is finally given by

$$\det(\boldsymbol{J}^D) = (1+\kappa)^2 + (1+\kappa)\kappa\phi, \tag{6.21}$$

and generalizing it for nd-dimension, the determinant reads

$$\det(\boldsymbol{J}^D) = (-1)^{nd}(1+\kappa)^{nd-1}(1+\kappa+\kappa\phi). \tag{6.22}$$

As far as all the constants are positive in equation 6.22, the Jacobian, independently of its dimension, may not be equal to zero

$$\det(\boldsymbol{J}^D) \neq 0, \tag{6.23}$$

so to summarize, it will be always positive or negative depending on its dimension

$$\delta_{ij} = \begin{cases} \det(\boldsymbol{J}^D) > 0, \text{ if } 2D, \\ \\ \det(\boldsymbol{J}^D) < 0, \text{ if } 3D. \end{cases} \tag{6.24}$$

It is therefore always invertible.

Jacobian approximation discussion. In this work, we approximate the Jacobian by its diagonal $\boldsymbol{J} \sim \text{diag}(\boldsymbol{J})$, considering only the drag force terms. The rest of the forces like lift, gravity, buoyancy or Brownian force are included in the acceleration term $\boldsymbol{a}_p^{n+1}$ from equation 6.3.

This assumption could be considered a good solution for computational performance if the number of iterations needed to converge are not too many. This point is proven using some mathematical cases, shown in subsection 6.3.1, in figures 6.6 or 6.12.

Nevertheless, an interesting topic for discussion may be if it is really necessary to adopt the diagonal terms approximation, especially when lift force is involved. Hence, a future work comparing computational times and error ratio using both configurations (diagonal and full matrix) is proposed.

In addition, studying the convergence of the NR can also be an interesting topic.

6.2 Adaptive Time Step (ATS)

Let us define δt_f as the fluid time step. The straightest way to solve particle transport is using the same time interval for fluid and particles $\delta t_f = \delta t_p$, where δt_p is the time step of the particles. On the contrary, an independent δt_p is desired, some assumptions must be considered. In our case, a linear variation of the fluid solution during δt_f. If so, intermediate solutions can be interpolated, allowing smaller intervals $\delta t_p < \delta t_f$. The only restriction about the new time steps is that particle final time must be the same than fluid, as the example of figure 6.2 shows. There are three reasons to have a smaller particle time step than a fluid time:

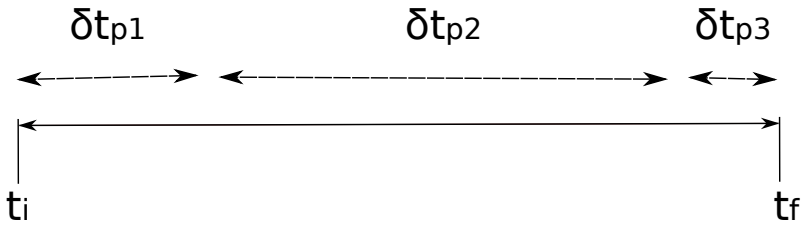


Figure 6.2: Scheme of adaptive time step. Particles can adopt a smaller time step than the fluid.

One element per time step. Particles cannot cross more than one element from one time step to another, because only the first neighborhood list of elements is looped (for accuracy and parallelizing reasons as further detailed in section 7). Otherwise, time step is automatically decreased.

There is also a mathematical reason to allow a particle to cross only one element each time step. We must compute the right linear variation fo the velocity from one element ot another.

Reaching convergence of the integration scheme. Time step is decreased when convergence in Newmark-β is not reached. The convergence factor can be controlled before the simulation starts, previously choosing the value of the residual norm ϵ_c defined in equation (6.7).

Control the error time of the integration scheme. Before the simulation starts, the user must define the maximum acceptable error due to the discretization $\epsilon_{err} \ll 1$. This error is normalized using a characteristic length L. The election of L is in some way arbitrary and may vary depending on the properties of the problem. The discussion will be deeper developed during next subsection 6.2.1. But, at the same time, in some problems a good choice can become key, as will be shown in the examples below. Some of the proposed characteristic lengths are the diameter of the particle d_p, the length of the element h or the instant velocity of the particle $|\boldsymbol{u}_p|$ multiplied by a characteristic time τ.

In order to estimate the new time step δt_{err}, let $\boldsymbol{x}_{p,exa}^{n+1}$ be the exact solution found by applying Taylor series:

$$\boldsymbol{x}_{p,exa}^{n+1} = \boldsymbol{x}_p^n + \boldsymbol{u}_p^n \delta t + \frac{1}{2}\boldsymbol{a}_p^n \delta t^2 + \frac{1}{6}\frac{d\boldsymbol{a}_p^n}{dt}\delta t^3 + \mathcal{O}(\delta t^4). \tag{6.25}$$

Subtracting the equation (6.25) and (6.2) and neglecting $\mathcal{O}(\delta t^4)$, one obtains

$$\boldsymbol{x}_{p,exa}^{n+1} - \boldsymbol{x}_p^{n+1} = \beta(\boldsymbol{a}_p^{n+1} - \boldsymbol{a}_p^n)\delta t^2 - \frac{1}{6}\frac{d\boldsymbol{a}_p^n}{dt}\delta t^3. \tag{6.26}$$

We consider the following approximation for $\frac{d\boldsymbol{a}_p^n}{dt}$,

$$\frac{d\boldsymbol{a}_p^n}{dt} \simeq \frac{\boldsymbol{a}_p^{n+1} - \boldsymbol{a}_p^n}{\delta t}, \tag{6.27}$$

we require the difference $\boldsymbol{x}_{p,exa}^{n+1} - \boldsymbol{x}_p^{n+1}$ to be lower than a given characteristic length, as mentioned earlier. Thus we have that

$$\max|\boldsymbol{x}_{p,exa}^{n+1} - \boldsymbol{x}_p^{n+1}| = \epsilon_{err} L, \tag{6.28}$$

where $\max|\boldsymbol{x}_{p,exa}^{n+1} - \boldsymbol{x}_p^{n+1}|$ means the maximum difference in absolute value of any of the dimensions. Isolating δt, it is finally obtained

$$\delta t_{err} = \sqrt{\frac{\epsilon_{err} L}{(\beta - 1/6)\max|\boldsymbol{a}_p^{n+1} - \boldsymbol{a}_p^n|}}. \tag{6.29}$$

Obviously, equation (6.29) is only reasonable for the choice $\beta \neq 1/6$. It is interesting to take a look at the limiting behavior of this formula: $\frac{d\boldsymbol{a_p}}{dt} \to 0$ or $\frac{d\boldsymbol{a_p}}{dt} \to \infty$. Furthermore, notice that if $\boldsymbol{a}_p = ct.$, the exact solution is obtained regardless of the time step (e.g. parabolic movement when $\boldsymbol{a}_p = \boldsymbol{g}$, as shown on figure 6.3). Some researchers like Schweizerhof et al. (2004a) or Schweizerhof et al. (2004b) arrive to a similar discretization error formula for a Newmark-β integration scheme.

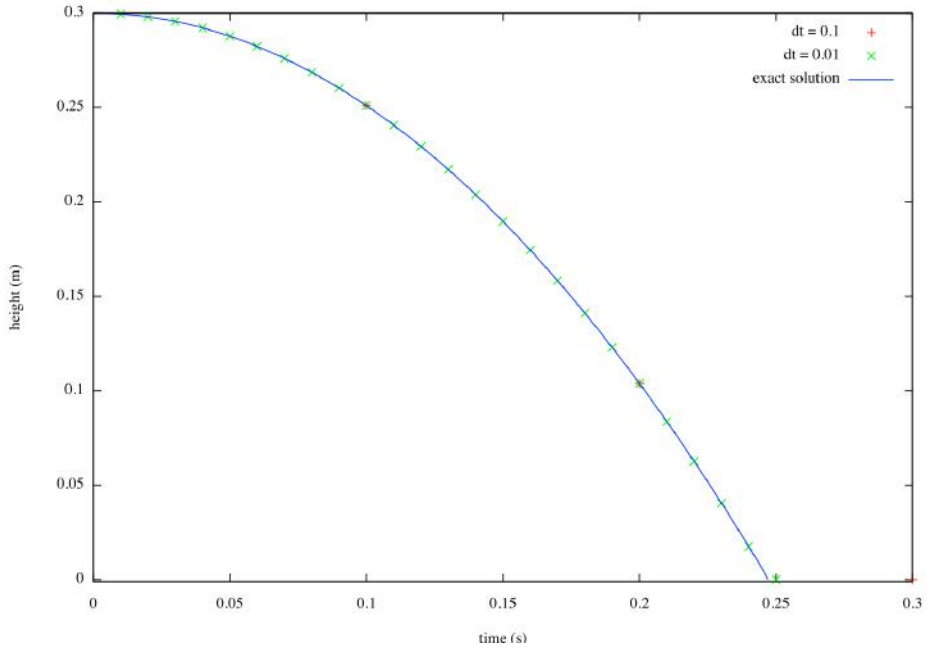


Figure 6.3: A particle with initial velocity $\boldsymbol{u}_p^0 = (1, 0)m/s$ is only affected by gravity. Independently of the time step, the simulated particle follows the exact trajectory line. However, deposition is not reached at the same time because of its fixed time step. Interpolation is thus necessary to recover the right time.

Finally, let us define the accuracy α as

$$\alpha = \frac{\delta t_{err}}{\delta t}. \tag{6.30}$$

The new time interval is only accepted if $\alpha > 0.9$. Otherwise, the process is repeated using $\delta t_{new} = \delta t$. Contrarily, a relaxation variable $s > 1$ is responsible of smoothly incrementing the δt of the next time step if possible, in order to optimize the number of steps computed:

$$\delta t^{n+1} = s\delta t. \tag{6.31}$$

6.2.1 Discussion of discretization error and characteristic length

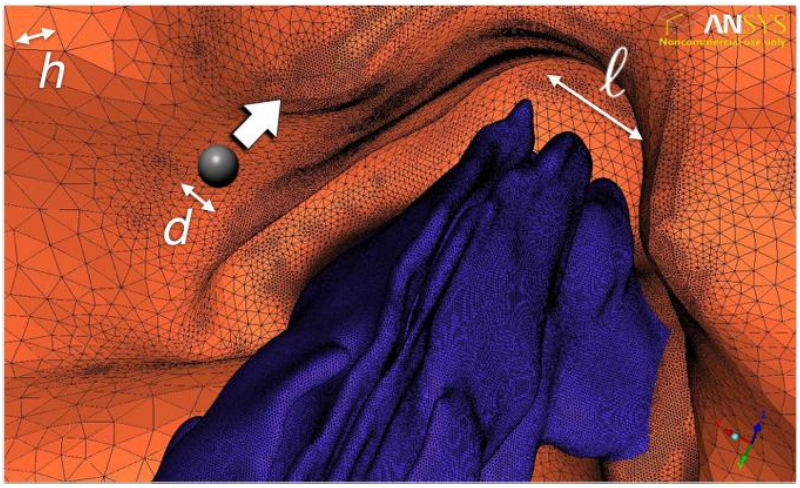


Figure 6.4: Which is the right characteristic length?

The time step of a particle δt will directly depend on two factors (more or less arbitrary). First, the error the user is willing to assume, in other words, the selection of the value of discretization error ϵ_{err}. Second, the characteristic length L the user considers more suitable for the problem.

As shown in figure 6.4, there may be different eligible L in the same problem. In the respiratory system, for instance, this could be the diameter of the particle d_p, the characteristic length of the element or the characteristic length of the obstacle such as the airway diameter.

The first option, d_p, has the main advantage that the response time or characteristic time of the particle τ_p (previously defined in section 5.4 by equation 5.36) depends on the square of its diameter. It means, smaller particles will react quicker to the fluid, so smaller time steps will be necessary to correctly track them. However, we typically simulate particles whith diameters that can be on the order of nanometers. According to equation 6.29, the time of the particles is proportional to $\delta t_p \propto L^{1/2}$. Hence, the smallest particles will have a really small time step and lots of sub-iterations will be required before reaching global time step.

If, otherwise, the chosen L is the characteristic length of the element, it means the result, somehow, depends on the mesh twice. The first one, because of the restrictions of the adaptive time step was one element per element as we previously explained; the second one because of the chosen L. Nevertheless, in the respiratory system, meshes are usually finer close to the wall in order to obtain better deposition results, which are of much interest

for medical purposes. Therefore, this option can become an extra-help.

Finally, choosing the option of the characteristic length of the obstacle brings a high computational cost if it must be calculated for every particle, every step. On the other hand, it is clear that in small channels, fluid velocity fields can be higher and deposition more recurrent, so smaller time steps can be helpful.

Security measure to avoid possible bugs or infinite loops. It is also important to remark that, for computational reasons, there are a maximum number of iterations a particle can do before reaching the global time step. This way, it is ensured that a single particle can not be able to consume too much computational resources requiring too many steps to reach it (next time step can not start until all particles are over) or even entering into an infinite loop because of a bad numerical behavior, which would paralyze the whole simulation because of a single particle. This maximum number of iterations can be chosen as a prescribed number before starting the simulation or using

$$maximum\,number\,iterations = \boldsymbol{int}\left(\frac{\delta t_f}{\delta t_{min}}\right), \tag{6.32}$$

where ***int*** means converting the result into an integer and δt_{min} the minimum time step allowed to a particle (set before starting the simulation). If a particle needs more iterations it means there must be an error and that a particle entered into an infinite loop. In the unlikely case it could occur, this particle is preferred to be removed instead of parallelize the simulation. At the end of the simulation a statistic of lost particles because of the numerical reasons is shown, so it can be checked if the number of lost particles is relevant.

In section 3.3.3.2, we explained a source of problems that made the code enter into an infinite loop without the maximum number of iterations measure set. At the time of writing this thesis, no particle had been lost for computational or numerical reasons. However, this limit of total iterations can be considered as a necessary security measure.

6.3 Results with adaptive time step (ATS)

In order to test the complete algorithm, it means a Newmark-β with an inner NR plus and ATS, two types of studies will be done. First, some mathematical cases will evaluate the algorithm in a ground level behavior. Next, a real case will be proposed and benchmarked.

6.3.1 Mathematical cases

Different mathematical cases in $1D$ and $2D$ have been created. The main goals of these study cases are:

i) Analyzing the convergence of the integration scheme.

ii) Comparing the behavior of non-adaptive and adaptive time steps.

These examples will be simplified as much as possible for a better understanding of the proposed scheme. Only one small particle with $\rho_p = 1000kg/m^3$, $d_p = 10^{-9}m$ transported in a fluid by drag force F_d, which will be the only force affecting the particle, will be simulated. As the particle is extremely small, drag force will quickly accelerate the particle until reaching the fluid velocity u_f, so the particle will approximately follow the same path as the fluid. In this way, we will be able to approximate the fluid pathlines as valid for the same particles path. In addition, characteristic length will always be particles diameter $L = d_p$.

- *Case I: Gaussian function*

Let us define a 1D domain, where a particle is at rest until it is submitted to an abrupt acceleration. The path the particle must follow is a Gaussian from $t_i = 0s$ to $t_f = 1s$, where the mean value is $\mu = 0.5$ and the standard deviation $\sigma = \frac{0.01}{\sqrt{2}}$, obtaining a function such that

$$\boldsymbol{x}_p = 0.6e^{\left(-\frac{t-0.5}{0.02}\right)^2} \tag{6.33}$$

In order to get the particle following the Gaussian bell, the derivative $\boldsymbol{u}_f = \frac{d\boldsymbol{x}_p}{dt}$ must be imposed for fluid velocity

$$\boldsymbol{u}_f = -3000(t-0.5)e^{-2500(t-0.5)^2} \tag{6.34}$$

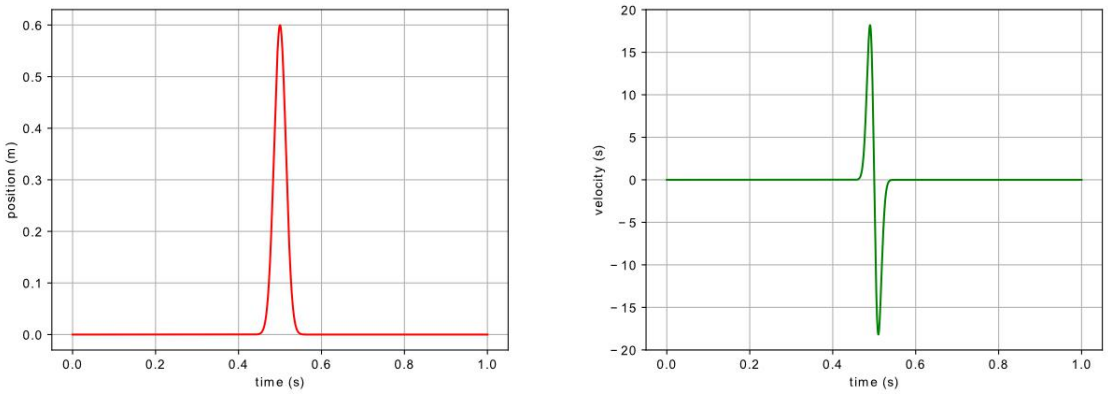


Figure 6.5: (a) Gaussian function. (b) Velocity fluid function.

This first experiment focuses on the convergence of the NR. Let us define the value of the tolerance (equation 6.7) $\epsilon_c = 10^{-12}$ and a fixed time step $\delta t = 10^{-3}s$

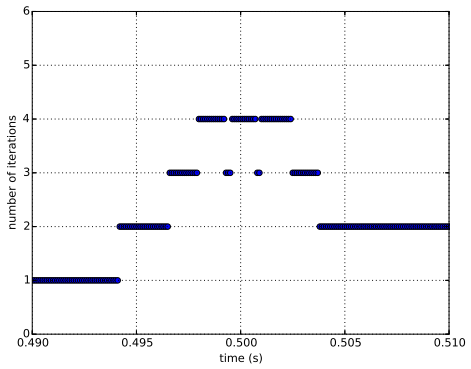


Figure 6.6: Number of NR iterations needed to obtain the convergence.

Figure 6.6 shows that the number of maximum iterations needed for convergence are 4, just when the particle is close to the maximum of the gauss function. The plot only shows the zone of interest when the particle starts to be accelerated. Before that, ϵ_c remains null (converging in the first iteration). Figure 6.7 shows how ϵ_c varies during iterations until it reaches the desired value.

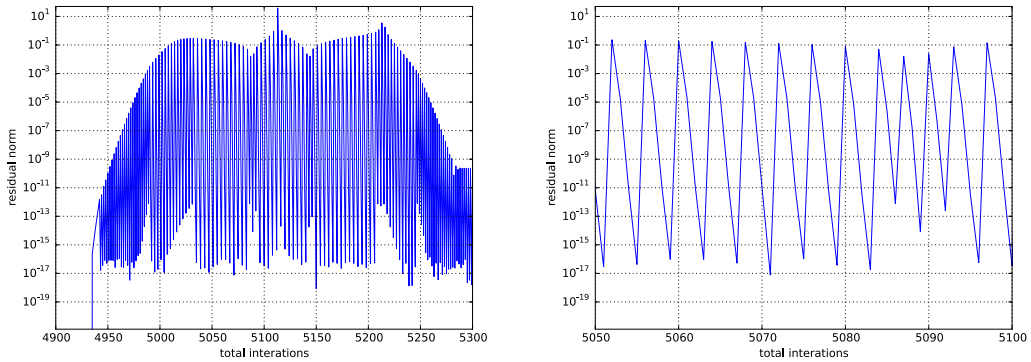


Figure 6.7: (a) Residual norm during the period the particles gets accelerated. (b) Same as left, but zoomed when particle approaches the maximum of the bell.

The second experiment with the Gaussian function shows the optimization of the algorithm when the adaptive time step is applied. The time step chosen in this case is larger: $\delta t = 0.1s$. So that the width of the gauss function is smaller than the time step. This is why, the particle doesn't

notice this change of velocity and rests in rest during the full simulation as shown in 6.8 (a).

Allow us to consider the adaptive time step now, defining the characteristic length as the diameter of the particle $L = d_p$. Figure 6.8 (b) shows that this same particle does notice the acceleration.

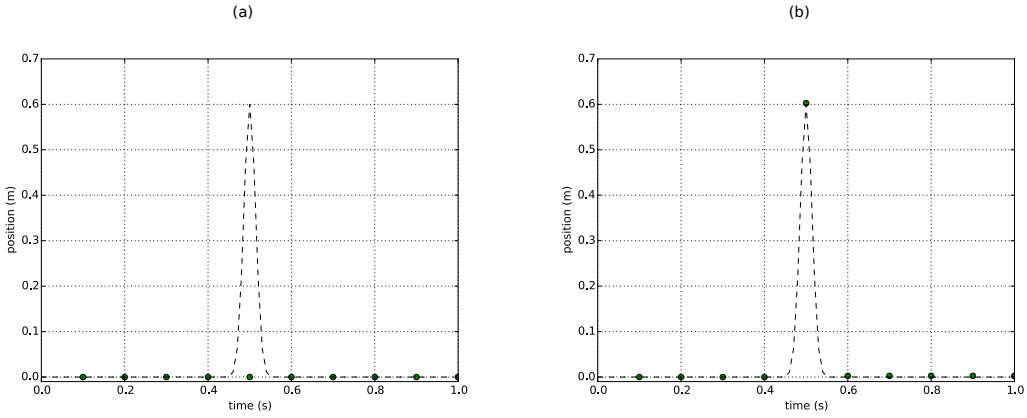


Figure 6.8: (a) Particle trajectory given a Gaussian function if adaptive time step is off. (b) Particle trajectory given a Gaussian function if adaptive time step is on.

Figure 6.9 shows the position of the particle every time step when using the adaptive. The initial δt is reduced to the minimum time step δt_{min} imposed by the user, in this case $\delta t_{min} = 10^{-12}$. As the particle notices the velocity remains constant, δt is quickly increased until it notices the Gaussian function and decreases δt again.

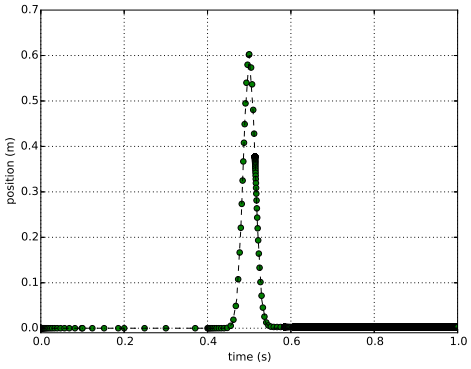


Figure 6.9: Gaussian function plotting every step of the particle when adaptive time step is applied

- *Case II: Sinusoidal function*

Let us define a $1D$ problem where a particle follows a sinusoidal path from $t_i = 0s$, $t_f = 1s$. The imposed velocity field must be the derivative of the time respect x_p such as 6.36 graphically represents in figure 6.10.

$$x_p = 0.0125sin(20t), \tag{6.35}$$

$$u_f = 0.25cos(20t). \tag{6.36}$$

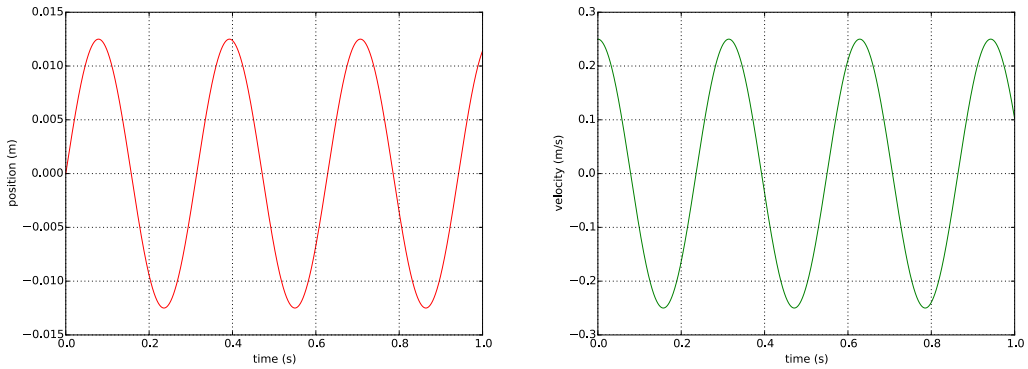


Figure 6.10: (a) Sinusoidal position function. (b) Velocity fluid function.

As in the previous example, first of all the convergence is studied. For this reason a fixed time step is chosen $\delta t = 0.001s$. As shown in figure 6.12, the number of iterations needed to reach the convergence always oscillate between 2 and 3 in a cyclic way as expected in a sinusoidal function and the maximum residual norm obtained is $\epsilon_c = 108.96$. A zoomed plot of when the maximum residual is reached is represented in figure 6.11.

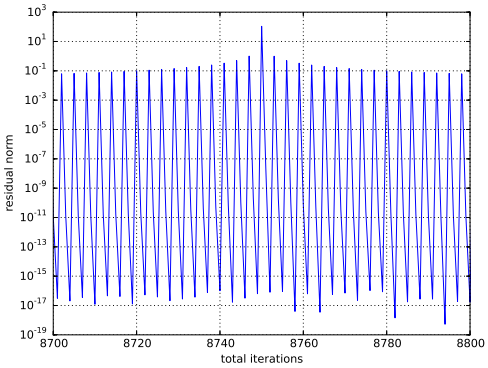


Figure 6.11: Zoomed plot of the residual norm in sinusoidal function.

Secondly, when studying the adaptive time step a larger fixed $\delta t = 0.1s$ is used. Figure 6.13 (a) shows the result without using adaptive time step. As

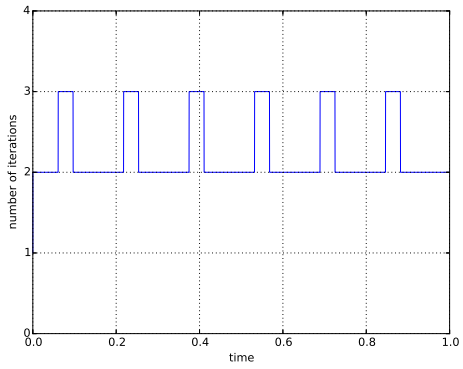


Figure 6.12: Number of iterations needed in order to obtain the convergence.

the frequency of the function is too high for the chosen δt, the particle has troubles when following the right path. On the other hand, once adaptive time step is on, with $L = d_p$, as shown in figure 6.13 (b), the particle agrees with the exact trajectory.

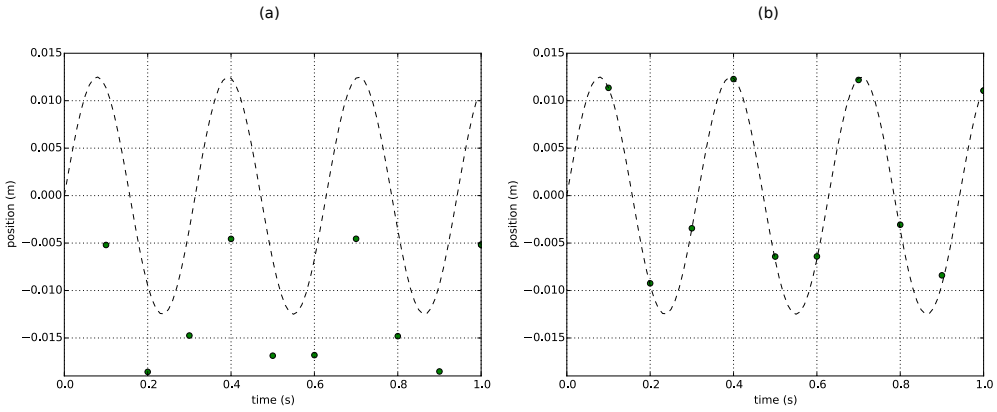


Figure 6.13: (a) Particle's trajectory given a sinusoidal function if adaptive time step off. (b) Particle's trajectory given a sinusoidal function if adaptive time step on.

- *Case III: Swirl*

We will consider a $2D$ problem, where a rotational fluid with a constant swirling velocity field is shown in figure 6.14. The particle is initially injected at a certain distance from the center. The fluid must transport the particle into a closed circumference to obtain the exact solution. In order to obtain this result, the pure transport model can be used instead of the force model, considering an infinitely small particle radius.

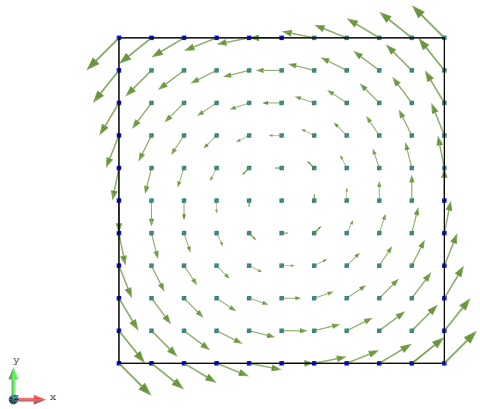


Figure 6.14: Mesh of the swirl case with a rotational fluid. Velocity fluid is computed on the nodes.

As shown in figure 6.15, using a constant time step, as the time step decreases, the particle trajectory fits better with a circle. If ATS is on, the circle always fits a closed circle.

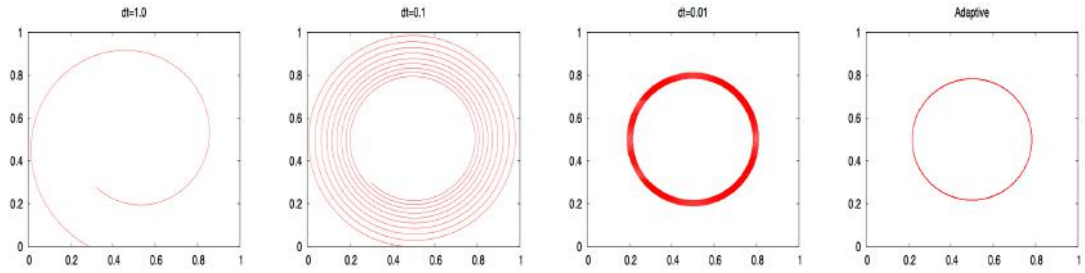


Figure 6.15: Comparison of swirl simulation with different and adaptive time steps.

6.3.2 Real case simulations

A real simulations is proposed now. The initial spray injection conditions will be simulated to study how ATS and NR help to simulate a particle velocity when particle is out of the Stokes regime. In addition, in chapter 8, section 8.3, a second real experiment (a bent pipe) studying particles deposition in function of its diameter will be simulated and results with other simulations found in the literature and empirical data will be compared to ours.

- *Non-Stokes regime: Spray injection*

In the literature there are some simulations of spray delivery in order to study the deposition of drug delivery in function of the size of the particle, the angle of the spray or the initial velocity of the spray.(Kimbell et al., 2007; Basu et al., 2017). According to them, typical particles diameter may vary between $d_p = 1\mu m - 500\mu m$ approximately, with a flow rate which can vary between $Q_{in} = 10L/min$ - $30L/min$ and an initial spray velocity between $u_p = 1m/s - 20m/s$.

In this problem, we are choosing the most extreme possible values, which will be the most difficult conditions to converge the solution. We chose $d_p = 1\mu m$ and $u_p = 20m/s$. Given these parameters, the relaxation time of the particle is $\tau_p \simeq 4 \cdot 10^{-6}$. Let us use a larger time step than τ_p, such as typical $\delta t = 10^{-5}s$, a generally valid time step to be able to properly simulate micro-particles in respiratory system. Unfortunately, in the very initial step of the injection of particles simulating a nasal spray, the difference $|u_f - u_p|$ is maximum (herein, $u_p \simeq 4.5m/s$) , thus particles are not under Stokes regime and $Re_p > 1$.

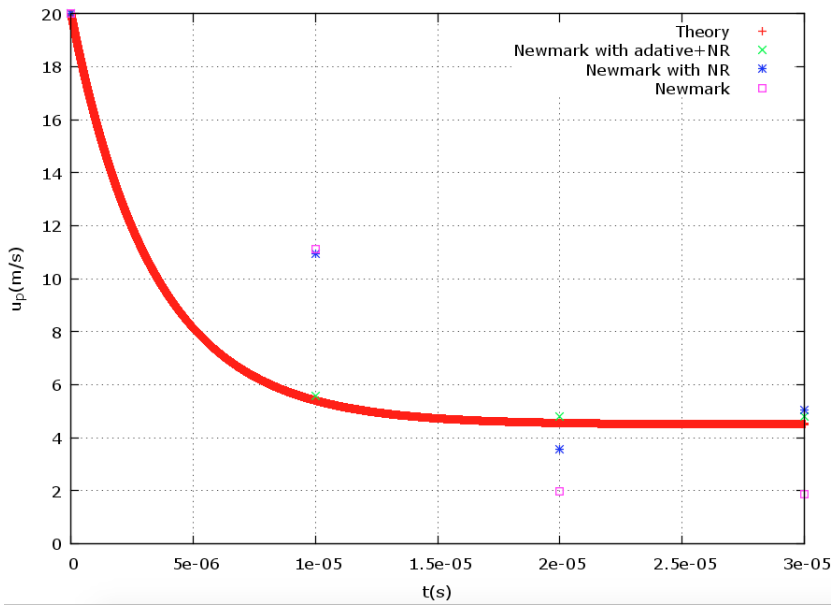


Figure 6.16: Behavior of relaxation time of u_p in a initial injection of a spray using different levels of configurations for the particles transport algorithm and comparative with the expected result

As shown in figure 6.16, using a $\delta t = 10^{-5}s$, only the configuration with ATS is capable of capturing the decay curve of particle velocity. The options without ATS do not, but after two more time steps, the simulation with NR converges again to the theoretical particle velocity.

Chapter 7

Parallelization of particle transport

The way the processor industry is going, is to add more and more cores, but nobody knows how to program those things. I mean, two, yeah; four, not really; eight, forget it.

S. Jobs

Summary: The particle transport in a fluid requires solving the fluid first, which is in charge of transporting particles, and next the particle transport itself. In an HPC context, finding out the most suitable and efficient computational setting (e.g., number and distribution of MPI processes) for solving the fluid and particle problems is not straightforward, as the fluid and particles have very different computational needs. While the fluid equations are solved in the whole computational domain, particles may have heterogenous concentrations. In this chapter, we propose a hybrid parallelization solution, together with a dynamic load balance strategy to enhance the overall simulation efficiency.

7.1 Strategies to simulate fluid and particle transport

A proper simulation of the respiratory system airways requires two main steps:

- Simulating the fluid.

- Once the fluid is solved, simulating the particle transport.

The air physics must be always solved before particle transport because particle motion depends on the air velocity. These two steps can be done using two different strategies:

- **Writing the air solution in the disk:** Solving the air and particle transport separately. This strategy requires writting the fluid solution on the disk.

- **Using RAM or MPI to send the air solution:** First solve the air and next the particle transport at each time step. The solution is communicated via Random-access memory (RAM) or Message Passing Interface (MPI). Herein, there are two different options: mono-code or multi-code, using a single instance of the code or two respectively, as explained further in section 7.2.

These two strategies are schematized in figure 7.1. The disk strategy is typically used for stationary flows and if the simulation must be repeated, recycling the same fluid solution several times. Alternatively, using RAM or MPI is typically used to solve transient flows.

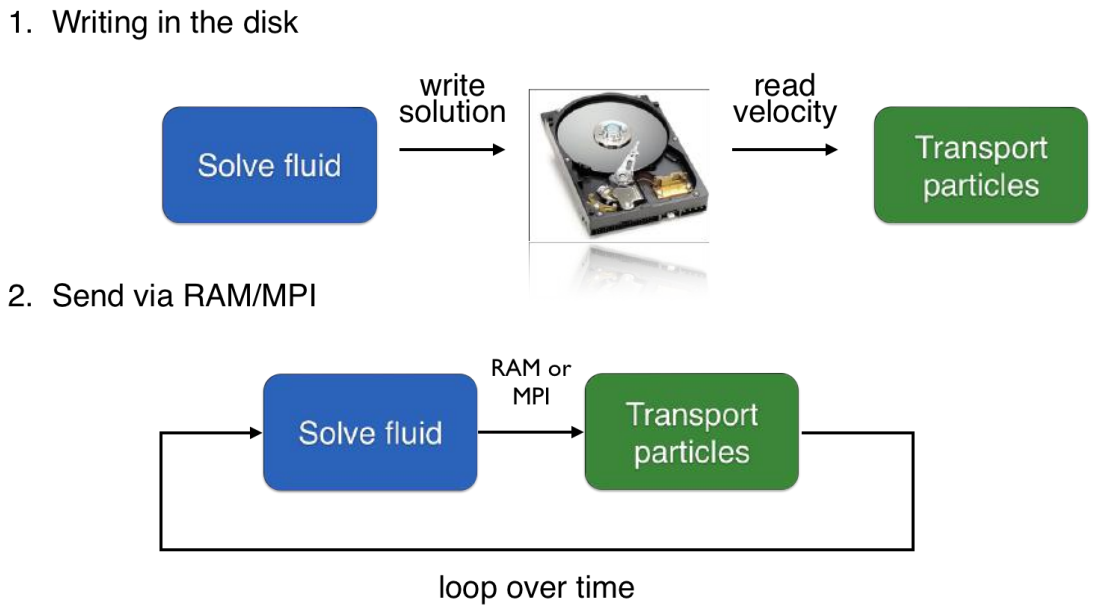


Figure 7.1: Two different strategies for computationally solving particle transport.

7.1.1 Stationary or transient flows

Using one of the strategies proposed above will mainly depend on the type of fluid: Stationary or transient.

Stationary flow. When the flow is stationary, only the steady-state solution is needed. Thus, once the fluid solution is converged, its solution can be passed to particle solver through files or through memory, as shown in figure 7.2. The first approach is obviously the preferred one when several experiments are undertaken (for example to transport different particle types, or to test different injection conditions).

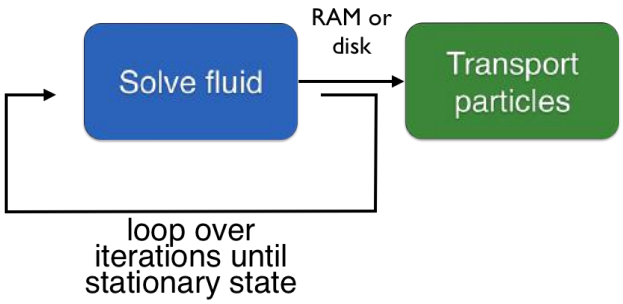


Figure 7.2: In the case of a stationary flow, first the stationary-state must be achieved.

Transient flow. The flows of interest in this thesis are transient. Thus, in order to capture the correct transient behavior of the particles, these should be transported from one time step to the next one. For this, the fluid velocity should be made available to the particle solver anytime. This is achieved either by communicating through RAM or using MPI. The latter one is considered when using the mono-code strategy while the last one is used together with the multi-code strategy, as we will describe in section 7.2.2. Writing on the disk is discarded, because the data collected can become too large as every time step the fluid solution is should be saved.

Let us mention another possible bottleneck of the simulation. When considering millions of particles, the RAM memory can also suppose a limitation, whenever the partitioning of the mesh used for particle transport is not properly done. As an example, let us consider 10^7 particles in one single MPI process. Transporting an individual particle requires about 20 real number variables (8 bytes), leading to a total memory equal to $10^7 \times 20 \times 8 = 2 \cdot 10^9 bytes = 2Gb$, without taking into account the mesh. Depending on the computational resources, we can thus quickly reach the memory limit of the core. This possible problem can thus be solved using

a specific partition for the particle transport, as allowed by the multi-code coupling.

7.2 HPC parallelization

In HPC, two main parallelization methodologies exists, for distributed memory and shared memory systems, respectively.

- **Distributed Memory:** Parallel processes do not share the memory space, so the only efficient way to move data from the address space of one process to that of another space is with a message passing (Saez et al., 2011). In the case of this thesis, we consider a mesh-based numerical method. The parallelization consists in assigning different portions of the mesh (the subdomains) to different processes. These subdomains of the mesh are created using a mesh partitioner (herein METIS), and they communicate through the MPI library.

- **Shared Memory:** Parallel processes share the memory space, so several processes can operate on the same data (Saez et al., 2011). The parallelization is done splitting a loop into chunks, each chunks being operated by different threads, running on different cores. This strategy is usually carried out by using Open Multi-Processing (OpenMP) library. Alternatively, we also use OmpSs library, developed in BSC, which expands OpenMP for supporting, among others techniques, asynchronous parallelism and GPUs. In the OmpSs case, loops are split into tasks. Both libraries have been used in this thesis.

Shared and distributed memory solutions are compatible. If they are both used in a single code, the methodology is referred to as a hybrid parallelization approach.

7.2.1 Particle transport parallelization

Distributed and shared memory parallelization can be used separately or together. In the case of a hybrid parallelization, the advantages of each methodology can provide a better performance. However, which proportion of processes and threads is the ideal one requires a deep performance study of the problem to be solved.

Fluid parallelization. The fluid can be efficiently solved by using MPI parallelization. Mesh partitioning is performed in such a way that the work is balanced. This can be relatevily well achieved when solving the NS equations with partitioners like METIS. However, this does not exclude OpenMP for specific improvements, as (Garcia-Gasulla et al., 2018) shows.

Particle parallelization. For independent particles and if there exists no race condition, OpenMP threads becomes an ideal parallelization method, as particles can be homogeneously distributed in threads, independently of their distribution in the computational domain. However, as particles coexist with the fluid, MPI parallelization is usually required too. For a better comprehension of the particle transport parallelization, let us firstly explain how the MPI and OpenMP parallelizations are done separately:

- **Parallelization using MPI:** A particle is located in one specific element of the mesh and therefore belongs to exclusively one subdomain. The mechanism to migrate one particle to another subdomain is through an MPI message whenever a particle falls into a halo element (an element from another subdomain sharing at least one node with an interface element) of the subdomain it belongs to. This is possible because we constrain the particles not to travel through more than one element from one time step to the next one as already explained in subsection 6.2. In figure 7.3, a hypothetical mesh divided in subdomains is shown together with halo elements used for migration.

- **Parallelization using OpenMP:** The transport of many particles consists of a loop over these particles. Each iteration of the loop is in charge of transporting a single particle, by evaluating the different forces it is submitted to and integrating its trajectory in time. The work of this loop can then be divided into chunks and distributed by OpenMP threads. As far as particles are independent and due to the existence of a single main loop, this shared memory parallelism is easy to put in place by using a single OpenMP pragma. OpenMP offers two possible scheduling for assigning the size of the chunks:

  - **Static:** If the scheduling is static, OpenMP, by default, divide the loop into equal-sized chunks or as equal as possible in the case where the number of loop iterations is not evenly divisible by the number of threads multiplied by the chunk size. A static schedule can be non-optimal, however. This is the case when the different iterations take different amounts of time.
  - **Dynamic:** It uses the internal work queue to give a chunk-sized block of loop iterations to each thread. When a thread is finished, it retrieves the next block of loop iterations from the top of the work queue. By default, the chunk size is 1. This scheduling type involves extra overhead.

Hybrid parallelization. MPI has a rigid parallelism. In the case of load unbalance, a mesh repartitioning and a particle redistribution would

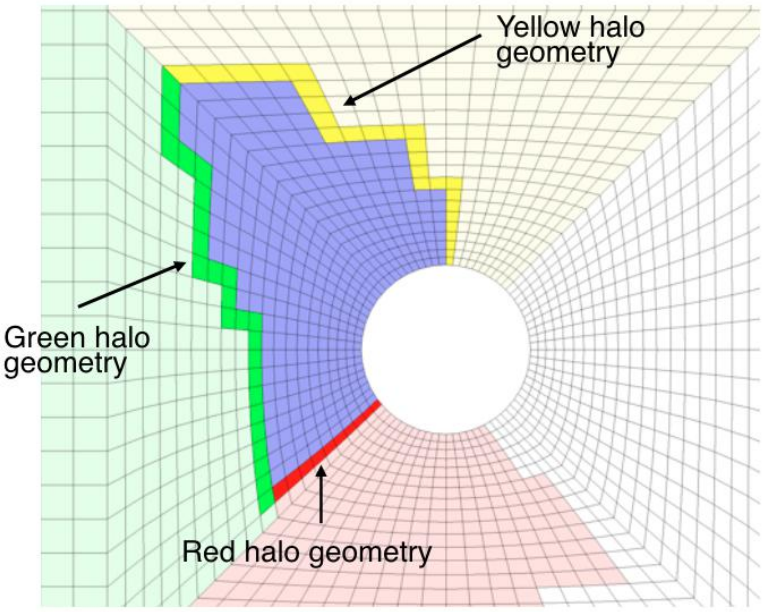


Figure 7.3: Blue subdomain is bordering red, green and yellow subdomains. In the case, a particle in blue subdomain changes subdomain, time step will be decreased if necessary, until it belongs to a halo element.

be necessary. This unbalance can be leveled by introducing shared memory parallelism. In order to employ both parallelization strategies, the OpenMP paradigm is used to parallelize the loop over particles inside each MPI process. The hybrid parallelization strategy is illustrated in Algorithm 2.

The variable N_{over} counts the number of particles that have reached their final time steps over all the MPI partitions (i.e., it is a global variable for all the partitions defining the total number of particles). Therefore, the MPI communication loop (from step 2 to step 34), is active until all particles have reached time step t_f, that is when $N_{over} = N_p$. Particles falling in halo elements are accumulated in an array and then sent to the corresponding neighbors at the end of the MPI loop.

The OpenMP paradigm used in this thesis to parallelize the loop over particles inside each MPI subdomains uses a dynamic scheduling with chunks of a certain size. The size of the chunk should be adjusted for each individual problem to obtain optimum efficiency. The chunk must be greater than the minimum to overrun the cost of the overhead but lower than a maximum to enable sufficient parallelism. The dynamic option is chosen because particles need different computational times to finish their computation. These differences can be caused by the numbers of iterations required to converge the Newton-Raphson, but mainly by the adaptive time step (ATS)

Algorithm 2 Hybrid parallel algorithm for the parallel transport of N_p particles.

```
1: N_over = 0, t_p = t_i for all particles p
2: while N_over ≠ N_p do
3:   !$OMP PARALLEL DO SCHEDULE (DYNAMIC,1000)
4:   !$OMP ...
5:   for Particles p in my subdomain do
6:     in_halo = .false.
7:     while t_p ≠ t_f and .not. in_halo do
8:       Update particle dynamical properties using the
         Newmark/Newton-Raphson scheme
9:       if particle p is in halo element then
10:        Save particle to be sent in an array
11:        in_halo = .true.
12:      else
13:        Update time step size δt_p
14:        Update time: t_p = t_p + δt_p
15:      end if
16:    end while
17:    if .not. in_halo then
18:      N_over = N_over + 1
19:    end if
20:  end for
21:  !$OMP END PARALLEL DO
22:  MPI_AllReduce(N_over)
23:  MPI_Send particles in halo elements to corresponding neighbors
24:  MPI_Recv particles from neighbors
25: end while
```

strategy, explained in previous chapter, in section 6.2.

Therefore, the hybrid paradigm is very efficient in terms of computational performance in the particle transport scenario. On the one hand, MPI parallelization is a good fist approach for the fluid, which computation can be distributed using partitioners like METIS or SCOTCH. On the other hand, particles concentration in the domain has no guarantee to be homogenous and they may be located in a very small portion of the computational domain. In this case, only few MPI processes would be working, so load is unbalanced. In the particle's approach, OpenMP allows to distribute a balanced number of particles per thread. In the present case, as particles are independent, OpenMP parallelization is very efficient.

7.2.2 Coupling and multi-code strategy

In this work, we only consider a one-way coupling, which means that the particles are transported by the fluid but have negligible effects on the fluid dynamics. Thus only the fluid velocity is needed to carry out the transport through the drag and lift laws, as explained in chapter 5. Two solutions will be analyzed and compared.

Firstly, synchronous coupling using the same instance of the code, and thus the same subdomain partitioning for the fluid and the particle solvers, as illustrated in Figure 7.4 (Top). In this case, the fluid equations cannot be solved while the particles are transported. The particle transport is therefore a synchronization point in the code execution.

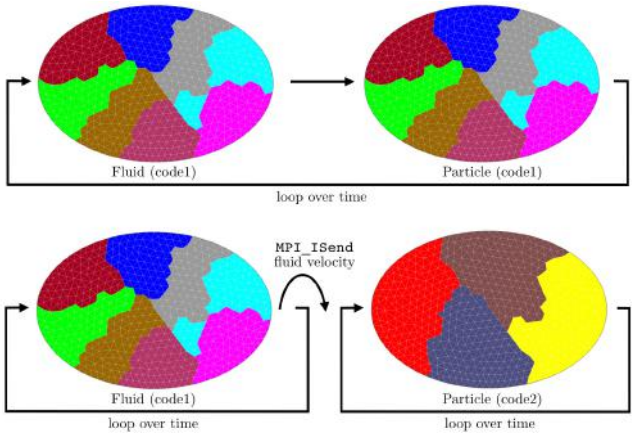


Figure 7.4: Fluid-Particle coupling. (Top) Synchronous coupling using the same code. (Bot.) Asynchronous coupling using two different codes, or two instances of the same code.

Secondly, in order to gain asynchronism, the fluid to particle coupling is also achieved via a multi-code strategy, as illustrated by Figure 7.4 (Bot.). In the present work, two instances of the code Alya are used. In this case, the fluid and particle domains are partitioned independently, with a different number of subdomains. At the end of a time step, once the velocity and pressure of the fluid are obtained, the velocity field is sent to the particle subdomains, via the `MPI_ISend` function, and received by the particle subdomains via the `MPI_IRecv` function. Then, once this communication is finalized, the fluid solver can proceed to the next time step and the particle solver can transport the particles. Through this asynchronism mechanism, we obtain an overlap of the fluid and particle calculations. Let us note that another advantage of this strategy is that load balance (explained in next subsection 7.2.3) at the MPI level can be achieved independently for both codes, thus adapting to the specific requirements of the two physics.

At this point, an important remark should be made. In order to fairly compare the synchronous and asynchronous couplings, one must consider a fix amount of computational resources. This implies that asynchronism is gained at the expense of using less resources for solving the fluid. In Section 7.3, we will analyze under which conditions the asynchronous strategy can be efficient.

7.2.3 Load Balance

Load balance is a measure of the work load distribution among the CPUs involved. A better load balance implies better efficiency. Mathematically, the efficiency can be described as

$$\text{eff} = \frac{\sum_{i=1}^{n} t_i}{n \cdot t_{max}} = \frac{t_{ave}}{t_{max}}, \tag{7.1}$$

where n, t_i, t_{max} and t_{ave} are respectively the number of CPUs, the time of work of each CPU or process, the maximum time of work among all the processes and the average time. This equation shows that if all the processes needed the same computational time, then eff $= 1$ and the load would be balanced. For a better comprehension of the load balance concept, let us imagine a hypothetical simulation.

- **Hypothetical unbalanced example (Part 1).** Figure 7.5 (Top) shows a hypothetical trace of a clearly unbalanced execution using 5 cores. Processor 1 works during 4s, while the others only work during only 1s, being the efficiency in this case eff $= 0.4$.

Load Balance in a hybrid parallelization. The hybrid parallelization approach explained in the previous subsection 7.2.1 may present two unsolved performance issues.

First, the load, specially in the particle code, is dramatically unbalanced. In the worst situation, all the particles may fall in a single subdomain. This is very likely to happen in the earliest time of the simulation, as particles are injected locally in the respiratory system simulations. Even if the particles were distributed in a homogeneous manner they will be migrating over time and may end up producing a load imbalance. But, not only the particle solver presents a work distribution problem. In (Garcia-Gasulla et al., 2018), the authors analyzed the load balance in some parts of a fluid code and propose a dynamic solution to solve it.

Second, when using the asynchronous coupling strategy, one must select the distribution of MPI processes between the fluid and particle codes. As we will see in Section 7.3, this decision have an important impact on the

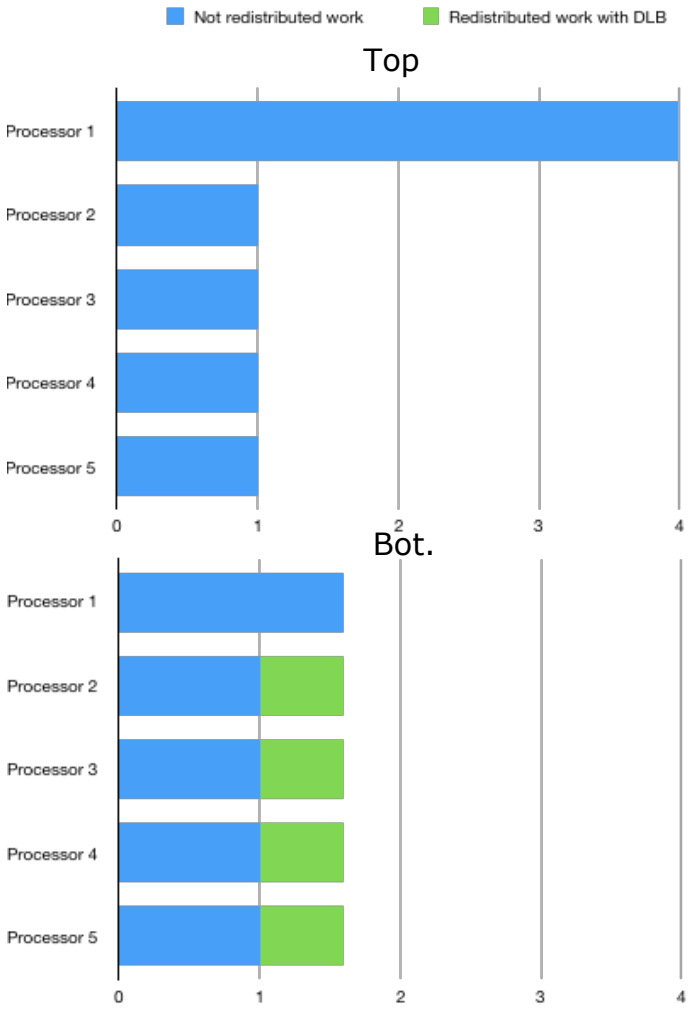


Figure 7.5: (Top) The trace of this hypothetical simulation shows an unbalanced execution as processor 1 works four times more than the other processors. (Bot.) The trace of this hypothetical simulation shows an ideally balanced execution with the help of DLB.

performance, and the only way of finding the optimal distribution is by evaluating each one. Moreover, as the load of the particles will change during the execution, the optimal distribution of MPI processes between fluid and particle codes can change as well.

To attack these problems we propose a dynamic solution, which is applied at runtime, using the Dynamic Load Balancing Library (DLB). DLB is a library developed in BSC and devoted to speed up hybrid parallel applications and maximize the utilization of computational resources (Garcia et al., 2009b). In other words, DLB is a dynamic library that can help parallel applications improving their load balance and hence, the efficiency

of the code. DLB is applied at runtime meaning that we do not need to analyze specific inputs or modify the application code. The philosophy of the library is to exploit the computational resources (i.e. CPUs or cores) of the MPI processes blocked in an MPI blocking call by other processes running on the same node, by spawning more threads of the second level of parallelism (i.e. in our case OpenMP).

When running with DLB, whenever an MPI process detects that it is not using its cores (i.e. it is waiting in a blocking MPI call), it will lend its resources to the system. Another process running on the same node can then use these cores and spawn more OpenMP threads to parallelize further the end of its computation. For a better comprehension of how DLB works, let us recover the previous hypothetical unbalanced simulation.

- **Hypothetical unbalanced example (Part 2).** As figure 7.5 (Bot.) shows, after 1s of computation, processes 2 to 5 have finished their own work after 1s while process 1 keeps working. Then, DLB allows the use of the available cores to process 1 to further split its work. In this ideal case, we then recover an optimum efficiency (eff = 1) and an execution time of 1.6s. The total time was thus reduced from 4s to 1.6s.

DLB applied to particle tranport. In the current scenario DLB will tackle three different problems:

- Load imbalance of fluid code;
- Load imbalance of particle code;
- Distribution of MPI processes among fluid and particle codes.

In Figure 7.6 we can see different traces of an execution of the synchronous code with 1024 MPI processes and 2 OpenMP threads each. The X axis represents time and each horizontal line is a thread (grouped by MPI process). In this traces the blue color represents computation of the fluid code, the green represents computation of the particle code and the orange global communications. The top image is the original execution, where we can clearly see the imbalance of the particle code, and less spectacular but also significant we can see the imbalance of the fluid code. In the middle image we can see the same execution with DLB. In the bottom figure we show a zoom of one of the nodes (the most loaded one and consequently the bottleneck). We can see that when the particle code is running (green), only one MPI process have computation to perform, and all the other MPI processes are waiting in a global communication (orange). At this point the process running the particles is able to use the 16 cores of the node, by spawning 16 OpenMP threads. The same observation can be made in the fluid code whenever imbalance is present.

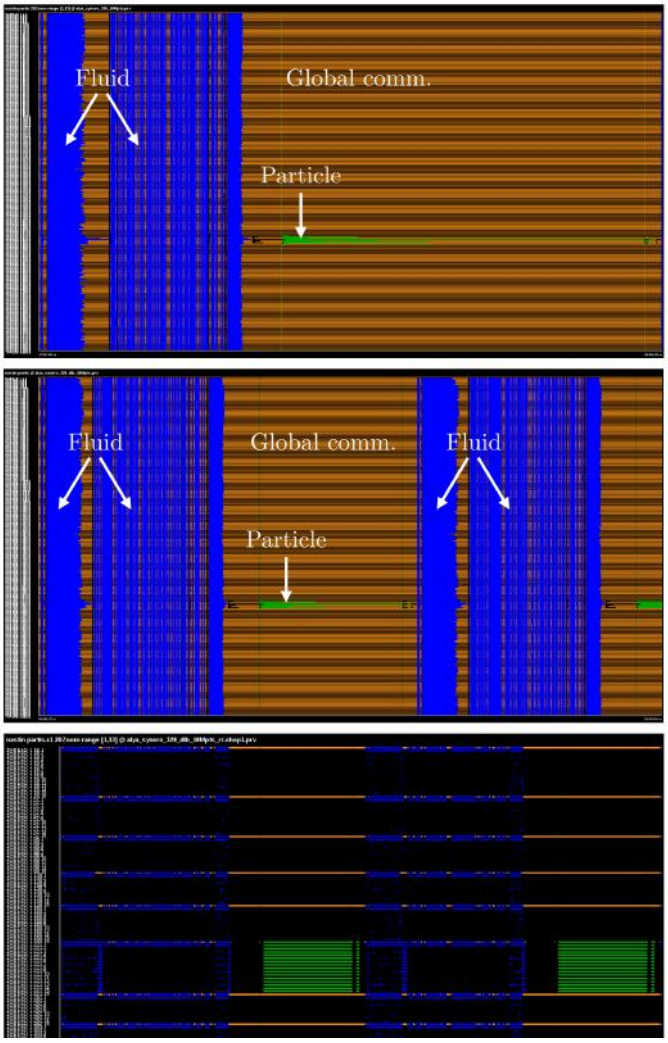


Figure 7.6: Synchronous coupling, 10M particles. (Top) Without DLB. (Mid.) With DLB. (Bot.) DLB zoomed.

7.2.4 Chronology of the parallelization of particle transport simulations

As shown on figure 7.7, firstly, Alya was used as a multi-physics mono-code only parallelized with MPI, but this way, as mentioned, particle transport simulations were absolutely unbalanced. Including OpenMP was the obvious next step, obtaining a hybrid parallelization. But this strategy was still too unbalanced, because if there exist an MPI task without any particle, OpenMP does not get activated there, so it is not helpful. For this reason, a multi-code strategy was proposed, being able to do different partitions for the fluid and for the particle transport. This strategy works at the

beginning of the simulation if the initial particle distribution is known, but the simulation gets unbalanced after some time as this distribution varies. Finally, a Dynamic Load Balance (DLB) was added to dynamically get adapted to the problem.

Figure 7.7: The balance of the code has improved in time including new parallelization and balancing tools.

7.3 Coupling results

To assess the performance of the proposed strategies, i.e. synchronous and asynchronous couplings together with dynamic load balance, we consider the particle transport in the respiratory system in a sniff situation. The objective of such simulations is to predict the deposition of the particles ejected from a medicinal spray inside the nasal cavity, large and small airways. Details on the meshing strategy and the simulation can be found in (Calmet et al., 2016). In the present case, the mesh is hybrid (tetrahedra, prisms, pyramids) and composed of 17M elements. Particles are injected in the vestibule of the nasal cavity. They are therefore concentrated in a small volume inside the vestibule and thus in few MPI tasks at the beginning of the run. Figure 7.8 shows the particles distribution in two MPI subdomain after injection, when using 256 subdomains. The figure shows also the connectivity graph of the subdomains. The nodes of the graph are located at the centers of gravity of the subdomains; the bars indicate the neighboring relations. The first time step exhibit therefore a very bad load balance and we will concentrate on them. As particles are transported and deposited, the load balance becomes higher and the relative weight of the particle solver with respect to the fluid solver decreases.

As an example of configuration, Figure 7.9 shows the connectivity graph of the MPI subdomains when using a total of $n_{fp} = 256$ cores, with $n_f = 192$, $n_p = 64$ for the asynchronous approach, where n_{fp}, n_f and n_p are the total number of cores and the dedicated numbers of cores to the fluid and to the particles respectively.

The simulations were carried out on Marenostrum 3 supercomputer, which nodes are composed of 2 Intel Sandy Bridge of 8 cores each. Several numerical and computational configurations have been considered:

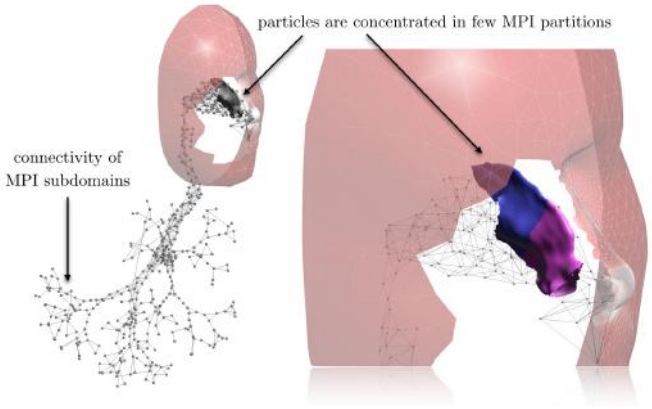


Figure 7.8: Confinement of particles in two MPI subdomains after injection, 256 subdomains.

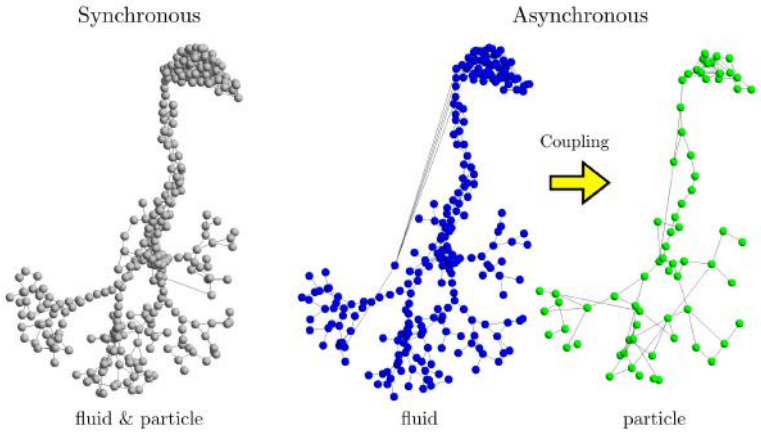


Figure 7.9: Connectivity graph of the MPI subdomains for the synchronous and asynchronous couplings with $n_{fp} = 256$, $n_f = 192$, $n_p = 64$.

- **16, 32, 64 computing nodes:** We will present the performance results obtained on 16, 32, 64 computing nodes of Marenostrum3 that correspond to 256, 512 and 1024 cores respectively.
- **DLB *vs* no DLB:** All the experiments have been executed with and without DLB, we will be able to compare the performance gain obtained when using the load balancing library.
- **Amount of particles:** We have considered two numbers of particles, 0.5 and 10 millions. This will allow us to compare the performance in a fluid dominating and particle dominating situation.
  - 0.5M particles → Fluid dominates

– 10M particles → Particle dominates

- **One core per MPI process:** For all the simulations, the MPI processes are started with one OpenMP thread. This means that OpenMP is exclusively used for load balance with DLB.

- **Average over 10 time steps:** We have executed 10 time steps starting with the particle injection in the vestibule of the nasal cavity. The performance results presented are the average execution time of the first 10 time steps.

- **Synchronous *vs* asynchronous:** We will compare the performance of the synchronous version with the asynchronous one. When running the synchronous code we will fill the nodes with MPI processes (i.e. 16 MPI ranks per node). In the case of the asynchronous code we will run different distribution of MPI ranks between fluids and particles.

  In all the cases we will compare executions using the same number of resources. Figure 7.10 illustrates the distribution of cores among the fluid and particle codes for both the synchronous and asynchronous approaches.

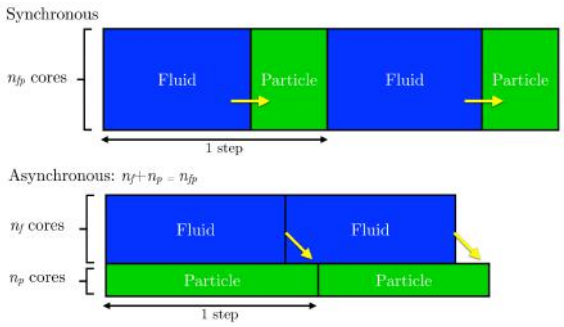


Figure 7.10: Distribution of cores for the synchronous (single-code) and asynchronous (two-code) couplings.

For the asynchronous executions, we have used four different proportions of MPI processes between the two codes (fluid and particle): we have considered 8 MPI processes for the fluid and 8 MPI processes for particle on each node (8 + 8), 12 + 4, 14 + 2 and 15 + 1. Table 7.1 shows a summary of the number of MPI processes used in each code in each for the possible configurations.

In Figure 7.11 we can see the average execution times for the different configurations. The first conclusion is that the performance when the computation is dominated by the fluid (0,5M particles, left hand side charts) differs from the one obtained by the particle dominated execution (10M

Per node	16 nodes		32 nodes		64 nodes	
	Synch	Asynch fluid + part.	Synch	Asynch fluid + part.	Synch	Asynch fluid + part.
8 + 8	256	128 + 128	512	256 + 256	1024	512 + 512
12 + 4	256	192 + 64	512	384 + 128	1024	768 + 256
14 + 2	256	224 + 32	512	448 + 64	1024	896 + 128
15 + 1	256	240 + 16	512	480 + 32	1024	960 + 64

Table 7.1: Distribution of MPI processes for fluid+particle, per node and in total.

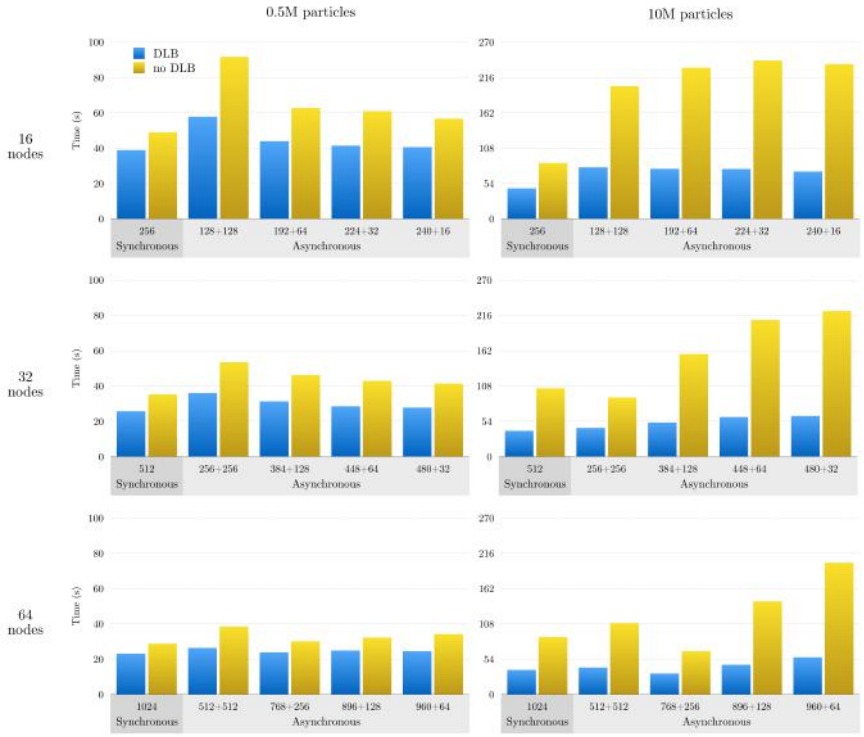


Figure 7.11: Timings. (From top to bottom: 16, 32, 64 nodes. (Left) 0.5M particles. (Right) 10M particles.

particles, right charts of the figure). This is due to the fact that the fluid and the particle codes have different scalability behaviors, while the fluid code scales quite well (Vázquez et al., 2016), the particle code has a poor scalability due to the high load imbalance. Therefore, the global scalability highly depends on which code dominates the computation.

We can see how DLB can help to improve the scalability by providing a better resource utilization. When simulating 10M of particles DLB can make the execution between a 45 and 72% faster (a speed up between 1.8 and 3.5 with the same number of resources). When running with 0.5M of particles, DLB can run between 20 and 32% faster (a speed up between 1.2 and 1.5 of the original code)

When analyzing the performance of the asynchronous version, we can see that it depends on the distribution of MPI processes among the fluids and the particles codes. Almost in all the cases we can find a configuration of MPI processes that perform better than the synchronous version. But at the same time when choosing any of the other configurations we can see how the performance drops, for some of the configurations the asynchronous code with a bad distribution can be two times slower than the synchronous one.

Finally we can observe that DLB is able to hide the performance problem when using a bad distribution of MPI processes among the codes (fluids and particles). The execution time when using DLB is almost constant independently of which version we are running (synchronous or asynchronous) and how many MPI processes for fluids and particles we are running.

To finish we would like to illustrate the behavior of the different configurations with some traces. In Figure 7.12 we can find the traces with 0.5M of particles, when the fluid code dominates the execution. We are using 328 MPI processes (256 + 72 for the asynchronous version) on 41 nodes with 2 OpenMP threads each MPI process (8 MPI processes per node). The colors of the trace represent the following:

Blue: Fluids code computation;

Green: Particle code computation;

Orange: Global communication;

Yellow: Point-to-point communication.

The X axis represents time, and each horizontal line one OpenMP thread (each two consecutive lines is a MPI process). For clarity we are showing a zoom of one node (the most loaded one, therefore, the bottleneck one) and one time step. From top to bottom we can find 4 different versions: synchronous coupling, synchronous coupling with DLB, asynchronous coupling and asynchronous coupling with DLB.

In this case, the performance of the synchronous and asynchronous versions (without DLB) are similar. We can observe that although the fluid code is slower, because it is running in less resources, 328*2 (656 cores) versus 256*2 (512 cores), the particle code can run in parallel with the fluid code and this slowdown is overcome.

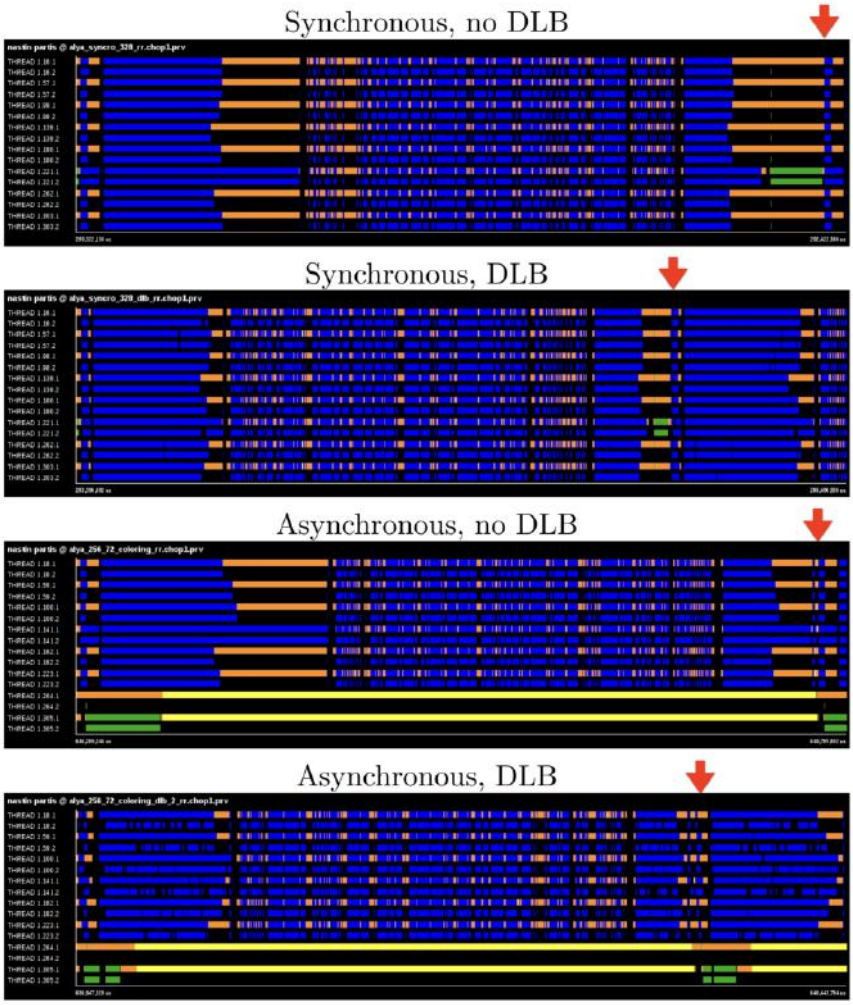


Figure 7.12: Comparison of different strategies, 0.5M particles

If we look at the executions with DLB, we can see how it can improve the performance of several parts of the fluid and particle codes computations.

In Figure 7.13, we can see the same executions but running with 10M particles, and a particle dominated scenario.

7.4 Brownian diffusion as a stochastic parallelizable process

When particles suspended in a fluid are sufficiently small not to being able to neglect their interactions with fluid molecules, suspended particles are transported because of constant collisions with fluid molecules. As far as

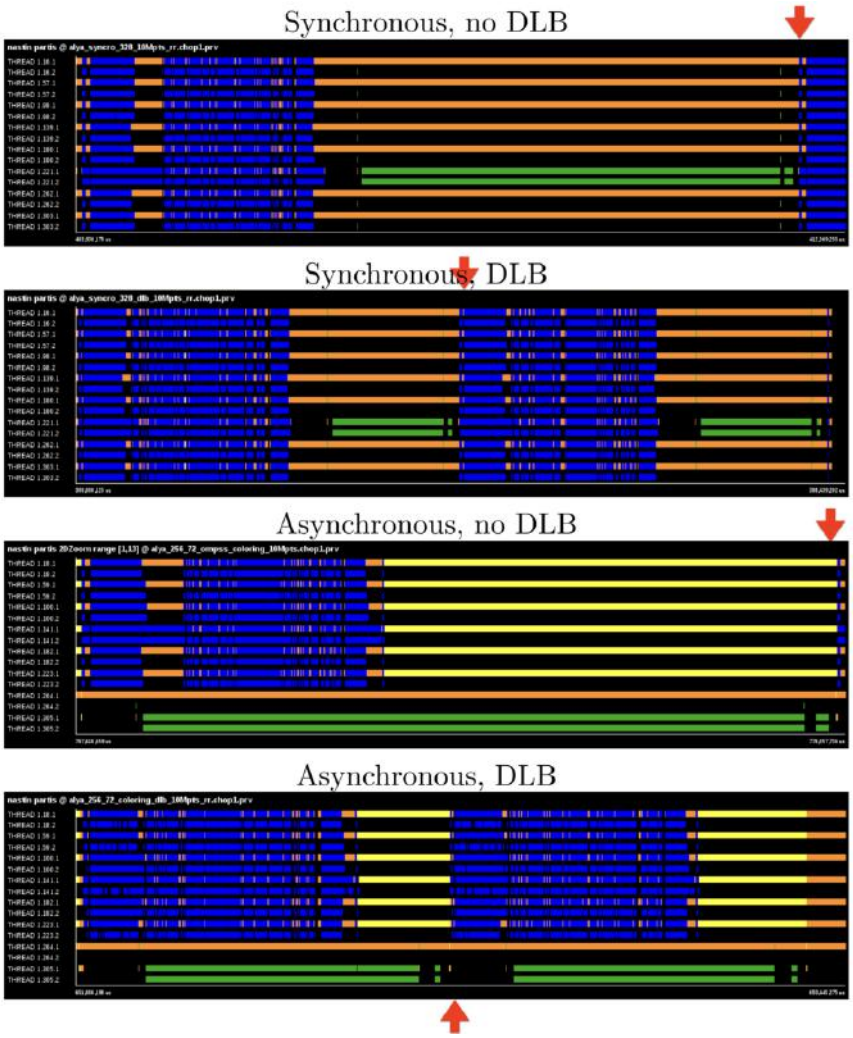


Figure 7.13: Comparison of different strategies, 10M particles

these collisions are continuous in time, they can be modeled as a stochastic process described by a normal distribution as shown in equation 7.2.

$$\rho(\boldsymbol{x}, t) = \frac{N}{\sqrt{4\pi D t}} e^{-\frac{\boldsymbol{x}^2}{4Dt}}, \tag{7.2}$$

where ρ, $\boldsymbol{x}$, t, N and D are the density of particles, position, time, number of particles and diffusivity respectively. This phenomena was first observed in 1827 by botanist Robert Brown and explained by Albert Einstein in 1905 (A.Einstein, 1903).

Brownian diffusion is applied, among others, in simulations of polymers (Filippidi et al., 2007), thermophoresis (Kuznetsov y Nield, 2010; Nield y Kuznetsov, 2009) or, in our case, nanoparticles deposition which is also

studied by (Shi et al., 2008b; Cheng et al., 1995). Any of these simulations or other which require simulating Brownian diffusion will need a random number generator (RNG) or, in the most of the cases a pseudorandom numbers generator (PRNG). PRNGs use iterative deterministic algorithms for producing a sequence of pseudo-random numbers that approximate a truly random sequence and they should be scalable and without data sharing or synchronization apart from an initialization phase (Coddington, 1997). In addition, our simulations are done in a HPC context with a distributed memory code, which means a parallel pseudorandom numbers generator (PPRNG) is demanded. Applying a good PPRNG is crucial in order to obtain good results and its implementation must be studied carefully.

Some studies have been done delving into parallelizing methods of the PRNG (LEcuyer et al., 2017) or studying the quality of PRNG when applied in parallel (Srinivasan et al., 2003). But these studies are very theoretical and don't take into account the unbalancing problems which are found in particles transport simulations, where particles could not be found equally distributed among processes. According to literature, there exist three main methods to obtain PPRNG: Leapfrog method, sequence splitting and independent sequences (Coddington, 1997).

As explained in previous chapter 7, two levels of parallelism are found in our code Alya within an hybrid shared/distributed memory architecture parallelized with MPI processes and OpenMP threads. In this hybrid scenario, (Teijeiro et al., 2013) studies how to optimally parallelize particles transport by Brownian diffusion, but only particles and transport by Brownian is taken into account, and other large-time CPU consumers parts of particles transport like fluid solver or other forces affecting particles are not.

7.4.1 Hybrid parallelization for Brownian diffusion

The parallelization of the particle transport is based on a hybrid MPI+OpenMP paradigm. This hybrid strategy was deeply explained in previous section 7. In this chapter, we maintain the same parallel structure but paying especial attention to the Brownian diffusion as a stochastic process, as shown in algorithm 3, which is the same than presented in section 7.2.1 but adding the Brownian motion calculation.

The variable N_{over} counts the number of particles that have reached their final time steps over all the MPI partitions. Therefore, the MPI communication loop (from step 2 to step 34) is active until all particles have reached time step t_f, that is, when $N_{over} = N_p$. As mentioned in section 7.2.1, particles falling in halo elements are accumulated in a stack and then sent to the corresponding neighbours at the end of the MPI loop. The OpenMP paradigm used to parallelize the loop over particles inside each MPI subdomains uses a dynamic scheduling with chunks of size 1000. The call to

Algorithm 3 Hybrid parallel algorithm for the Brownian diffusion of N_p particles.

$N_{over} = 0$, $t_p = t_i$ for all particles p
while $N_{over} \neq N_p$ **do**
  `!$OMP PARALLEL DO SCHEDULE (DYNAMIC,1000)`
  `!$OMP ...`
  for Particles p in my subdomain **do**
    $in\_halo = .false.$
    while $t_p \neq t_f$ and .not. in_halo **do**
      if first iteration of Particle p during δt_f **then**
        Call Gaussian random generator and calculate Brownian diffusion coefficient
      end if
      if Brownian force **then**
        Update particle dynamical properties (drag + Brownian forces) using the Newmark/Newton-Raphson scheme
      end if
      if Brownian displacement **then**
        Update particle dynamical properties (only drag force) using the Newmark/Newton-Raphson scheme
        Update particle position adding Brownian displacement outside the Newmark/Newton-Raphson scheme
      end if
      if particle p is in halo element **then**
        Save particle to be sent in stack
        $in\_halo = .true.$
      else
        Update time step size δt_p
        Update time: $t_p = t_p + \delta t_p$
      end if
    end while
    if .not. in_halo **then**
      $N_{over} = N_{over} + 1$
    end if
  end for
  `!$OMP END PARALLEL DO`
  `MPI_AllReduce(`N_{over}`)`
  `MPI_Send` particles in halo elements to corresponding neighbors
  `MPI_Recv` particles from neighbors
end while

Gaussian random generator 9) is done, independently of if its implemented as a force or as a displacement, inside the two layers of parallelization

paradigms. This call is only done during the first iteration of the global time step, although particles may have an smaller particular time step they will use the same number until the global time step is reached.

7.4.2 Random numbers generators in parallel

First of all, a list or stream of pseudo random numbers is needed. In the literature one of the PRNG more recommended is the lagged Fibonacci (7.3), especially when independent sequences are generated Coddington (1997); Aluru (1997); Srinivasan et al. (2003).

$$x_k = x_{k-p} \otimes x_{k-p-q} \bmod m, \tag{7.3}$$

where $\otimes$ denotes the operation such as $+,-,\times$. $m = 2^l$ for generating l bit random numbers. p is known as the lag of the generator and the seed for these generators is the first p random numbers.

Herein, there exist three main methods to generate PPRNG: the leapfrog method, sequence splitting and independent sequences.

The leapfrog method ideally generates the same sequence of random numbers for different number of processors. Let X_i be the i-th value of a random numbers sequence. For processor P of an N processor machine, generate the sub-sequence $X_P, X_{P+N}, X_{P+2N}, ...$ The potential problems with this method is guaranteeing uncorrelated elements in the sequence, which is more usual when the number of physical processors is power of 2.

Sequence splitting splits the sequence into non-overlapping contiguous sections, each generated by a different processor. The length of the sections L depends on the user, then processor P would generate the sequence $X_{PL}, X_{PL+1}, X_{PL+2}, ...$ This method can find out the same correlation problems than the method before, and it does not produce the same sequence for different numbers of processors. A part from that, in an unbalanced scenario like non homogenous distributed particles, where not processes will need different amount of random numbers (and this quantity can vary over time), splitting the sequence in the right range becomes a challenge by itself.

Finally, independent sequences consist in running the same sequential generator on each processor but with different initial seeds. The initialization of the seed on each processor is critical. Any correlation within the seeds could have terrible consequences in final results. Many default random generators use the CPU time as the seed, which is not suitable in parallel codes. If different processors call the seed at the same time, they will generate the same sequence. Even if initial seed is correctly randomized, there exist no guarantee that sequences will not overlap.

7.5 Results of Brownian motion

In this result section, our aim is to show a simplified theoretical case in section 7.5.1 where we will study:

- the optimum time step δt_p;
- differences between applying a random perturbation on force and on displacement;
- quality of PPRNG.

Using these results, we will be able to run a real patient simulation in section 8.4 in next chapter 8 with guarantees and compare our results to benchmark.

7.5.1 Random Walk case

In order to ensure a good behavior of the parallelized Brownian diffusion, a mathematical experiment is considered. Assuming that N particles start from the origin at the initial time $t = 0$ and based on equation 7.2, the diffusion equation has the solution given by

$$\rho(\boldsymbol{x}, t) = 1 - e^{-\frac{\boldsymbol{x}^2}{4Dt}}. \tag{7.4}$$

Different time steps δt are chosen in the interval $10^{-6} \leq \delta t \leq 5 \cdot 10^{-4}$. Each experiment with same time interval is repeated 10 times to minimize the random factor and the mean relative error $< e_r >$ is calculated comparing the experiment result with the theoretical one as

$$< e_r >= \frac{\|\boldsymbol{x}_m - \boldsymbol{x}_a\|}{\|\boldsymbol{x}_a\|}, \tag{7.5}$$

where $\boldsymbol{x}_m$ and $\boldsymbol{x}_a$ are the position measured in the experiment or simulation and the analytic position.

In addition, each experiment is run using an hybrid parallelization with 3 processes of MPI and 4 threads of OpenMP. The total number of particles is 2500 and the simulation is over when $t_f = 0.1s$ is reached.

Results are shown in Figure 7.14. Applying Brownian motion as a force or as a displacement gives similar results for small time steps $\delta t < 5 \cdot 10^{-5}$. In this case, applying the diffusion over the displacement obtains even more accurate results, but simultaneously, this method also starts to diverge sooner, whereas applying it over force keeps working fine when $\delta t < 3 \cdot 10^{-4}$. It must be noticed that the error using the force could not be calculated with $\delta t = 5 \cdot 10^{-3}$, because the Newmark-$\beta$ was not able to converge. It means, Brownian force can affect the stability of the integration scheme. According to this, the real patient simulation experiment from next section 8.4 will be done using a $\delta t = 10^{-5}s$ applying the diffusion over the displacement.

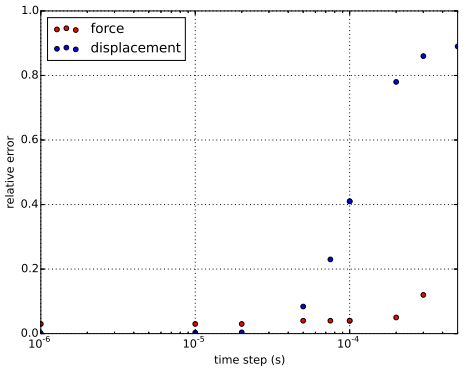


Figure 7.14: Comparison of the error in Brownian dissipation with different time steps between force and displacement.

Also using $\delta t = 10^{-5}s$ and 3 MPI processes + 4 OpenMP threads, let us repeat the experiment above to study the quality of our PPRNG. According to (Srinivasan et al., 2003; Kunuth, 1998), for random numbers in the interval $[0, 1)$, one can use the maximum-of-t test (Max$t$) to check its behaviour. Herein, t floating point numbers in aforesaid interval are generated and noted the largest number. We repeat this n times. The distribution of this largest number should be x^t. Hence, using the random walk case, this will be repeated the total number of time steps, in this case $n = 1000$ and t will be the number of particles multiplied by the dimension of the problem. If the dimension is 2, then $t = 5000$. With these numbers, figure 7.15 is obtained, where it is shown that the largest number of each step generated by our PPRNG fits with the power law x^t.

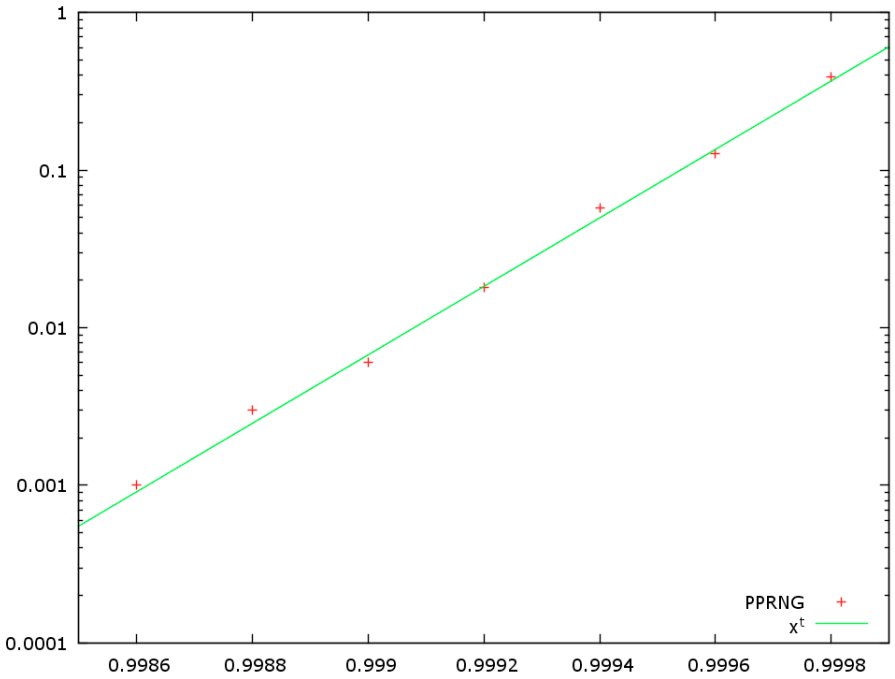


Figure 7.15: Largest random number generated in each time step must fit as a power law.

Chapter 8

Respiratory system simulations and results

It is better to have your head in the clouds, and know where you are... than to breathe the clearer atmosphere below them, and think that you are in paradise.

H.D. Thoreau

Summary: In chapter 2 medical applications of simulating respiratory system airways were exposed. Now, in this chapter, some results of these simulations are shown. First, the deposition convergence in the respiratory system is proven (section 8.1) in order to proceed to a three different real patients geometries simulation (section 8.2), work presented in (Calmet et al., 2018).

8.1 Deposition convergence in the respiratory system

In a collaboration with Imperial College (IC) of London, a real patient nose obtained by IC and Saint Mary's hospital of London was simulated.

The geometry was obtained via MRI and CT scan techniques. Once the mesh was generated, for an easier analysis of the results, four regions (zone I, zone II, zone III and zone IV) were defined as can be seen in figure 8.1. Each of these zones have differentiated fluid behaviors in the case of a sniff or drug delivery via spray, making particle deposition ratio change heterogeneously too. In addition, each zone has also different medical interests. Depending on the field of the study, like aforesaid medical purposes (e.g. nasal surgery, intranasal dug delivery...), the zone of most interesting can vary.

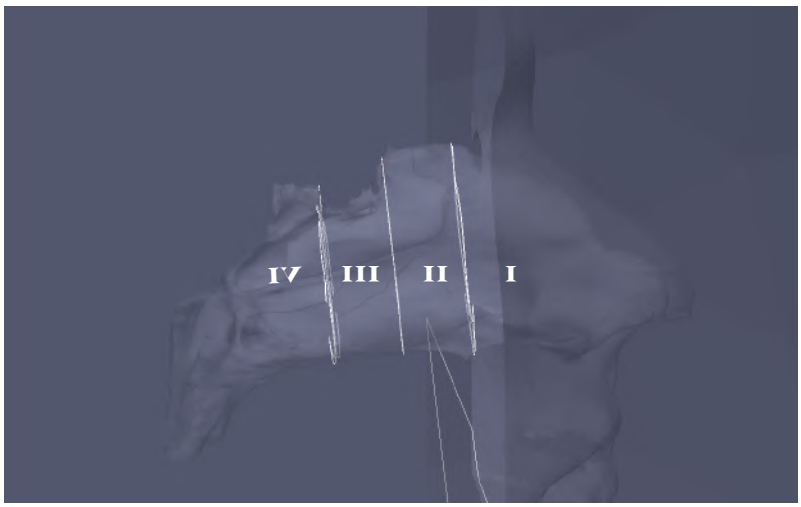


Figure 8.1: Splitting the nose domain in different zones for an easier study of particles deposition.

Firstly, to validate the code in this geometry and in particular the particles deposition statistics, two convergence experiments were done. The proposed geometry is used and a short sniff with particles is calculated, simulating for example drug delivery. Fluid initial properties (unsteady and laminar) are set to

$$\rho_f = 1.18415 kg/m^3,$$

$$\nu_f = 1.85508 \cdot 10^{-5} m^2/s,$$

$$Inlet\ flow\ = 30 L/min,$$

whereas particles setting is defined by forces involved (Gravity, Buoyancy and Drag) and particles properties

$$d_p = 1 \mu m$$

$$\rho_p = 1000 kg/m^3$$

In figure 8.2 the time to reach a constant deposition is evaluated, observing that the stationary regime in deposition terms is obtained approximately when $t \geq 0.06s$.

A second experiment is executed and shown in 8.3, where different number of particles are simulated, demonstrating that in the magnitude order of 10 thousands or 100 thousands the variation in deposition ratio remains insignificant.

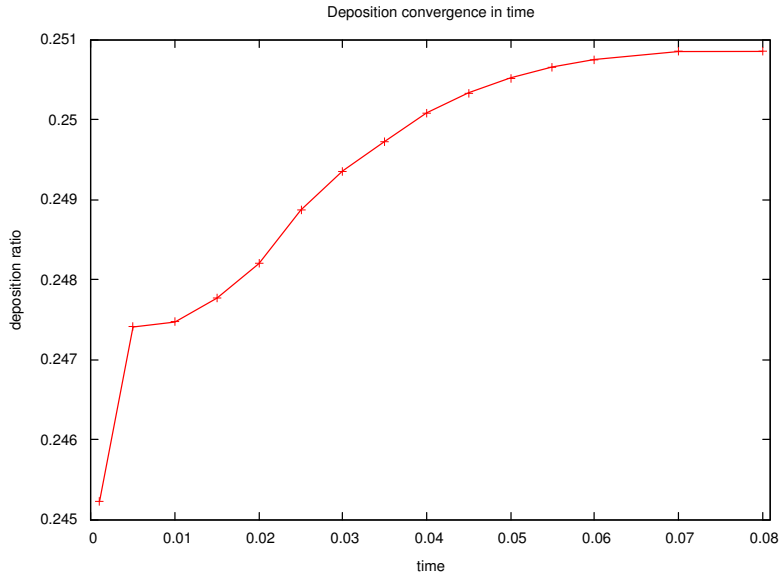


Figure 8.2: Checking necessary time to converge deposition in a sniff

8.2 Three subjects experiment

In coauthored paper (Calmet et al., 2018), results with three different nasal geometries are shown. Given the total deposited particles N_{dep} in the area of interest and the total number of initial particles N_{in} released at the nostrils, we define the deposition efficiency as

$$\nu = \frac{N_{dep}}{N_{in}}. \tag{8.1}$$

The inlet, situated in the extended nostril, has a constant velocity profile with a steady flow rate equal to $Q_{in} = 20L/min$. Two particles injectors are located at each nostril.

First, particle deposition in the nasal cavity is compared to experimental data reported by (Kelly et al., 2004a). The human nasal cast is based on the same MRI file it will be employed for the three subjects study and the same impaction parameter IP shown in next equation

$$IP = d_a^2 Q_{in}, \tag{8.2}$$

where d_a is the particle aerodynamic diameter. The aerodynamic diameter is that of a sphere with unit density ($1g/cm^3$) and its mass is equal to the mass of the actual particle.

Benchmark results are shown in figure 8.4. Here, it is proven an acceptable agreement between our simulation and the measurements of

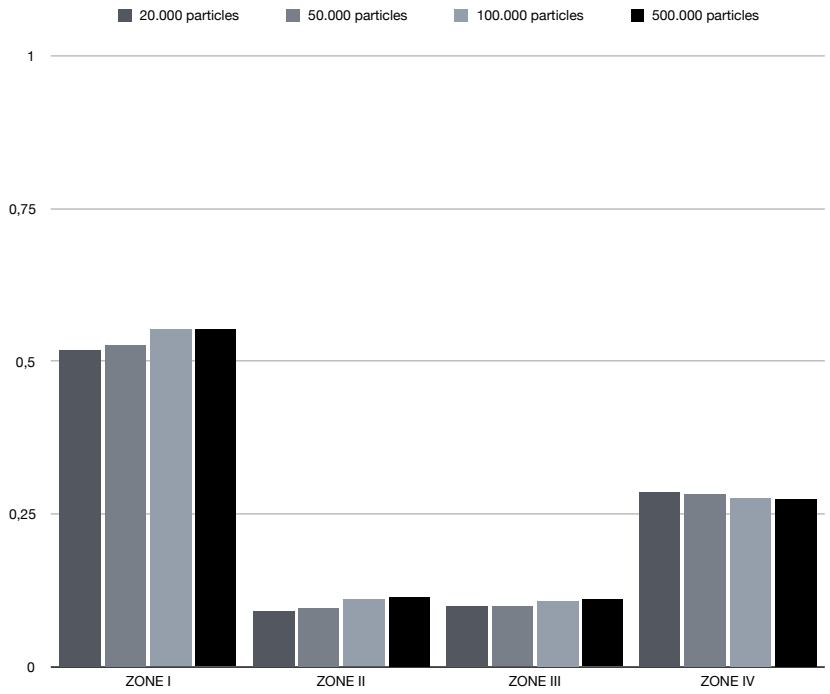


Figure 8.3: Ratio of deposition (over 1) convergence variance in different zones in function of initial number of particles

(Kelly et al., 2004a). However, differences in the deposition results are due to the coarser airways surface in the replica producing higher deposition efficiencies than the numerical model (Shi et al., 2007b).

In the experimental case of the three subjects shown in figure 8.5. As expected, maximum deposition occurred in the anterior section of the nasal airway for all three subjects (check figure 8.6, colored with blue surfaces). The deposition in the anterior surface is high for larger particles at lower inhalation flow rates. The deposition decreases for small particles with high inhalation flow rates and large particles with medium inhalation flow rates. The deposition in the inferior meatus is low in all subjects. The deposition in the olfactory region is very low for subject A and almost non-existent in subject B and subject C. Clearly, for direct delivery to the brain via the olfactory region, directional inhalation of nano-particles would be required.

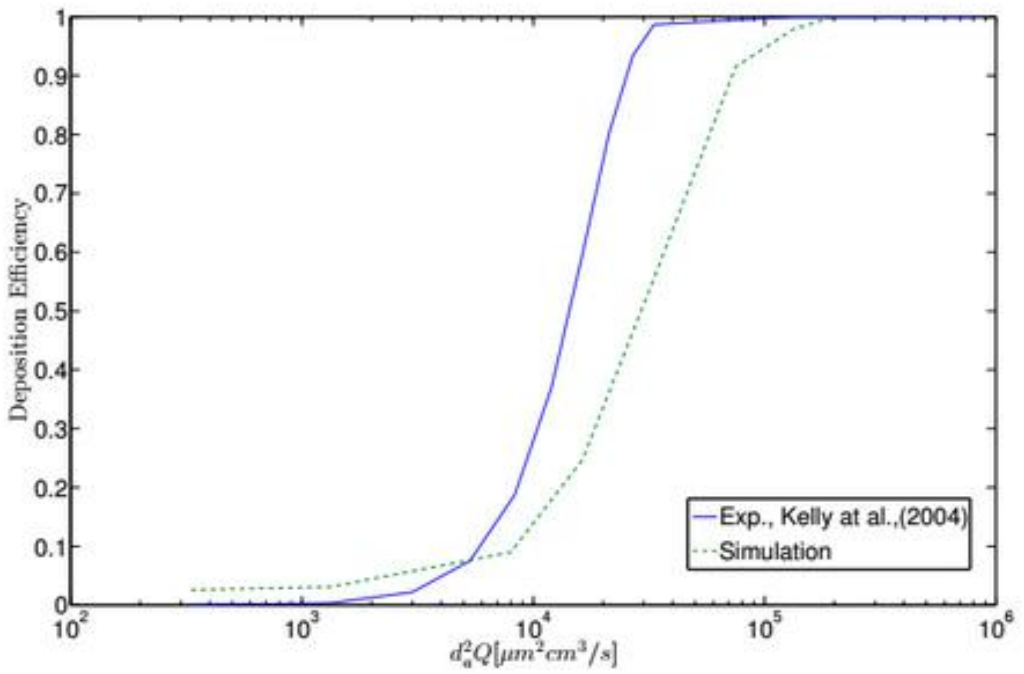


Figure 8.4: Particle deposition efficiency comparison between simulation and experiment

8.3 Experimental benchmark with adaptive time step (ATS): bent pipe

In this benchmark our goal is to validate the ATS using a real benchmark in order to compare experimental data provided by (Pui et al., 1987) with our simulations. In this experiment, a $90 < BA >$ bend pipe (elbow diameter = 24.25mm, pipe diameter = 8.51mm and a parabolic inlet condition imposed for the fluid). 12000 particles with different diameters ($d_p = 1 - 15\mu m$) are injected at the inlet.

The deposition efficiency is calculated in function of the Stokes number, which was explained more in detail in the previous chapter in section 5.4. In this problem the characteristic length L, defined in Stokes number equation 5.38, is the radius of the tube R_t. Thus, St will be described by

$$St = \frac{C_{slip}\rho_p d_p^2 U_0}{18\mu R_t}. \tag{8.3}$$

As shown in chart 8.7 the convergence in time for deposition is reached using $\delta t_p = 10^{-6}s$ (blue line in the figure 8.7). Lower time steps don't

considerably change the results (purple line in figure 8.7). Although the option $\delta t_p = 10^{-4}s$ (red line in figure 8.7) is found far from the solution obtained by smaller time steps, if the adaptive time strategy is applied, as shown in figure 8.8, the deposition results get much closer to the converged solution. Two different adaptive time steps are used here. Although the characteristic length in both is d_p, the ϵ_{err} chosen (described in chapter 6, section 6.2) is different. The higher error option takes $\epsilon_{err} = 10^{-2}$, obtaining a very similar solution with fixed time step $\delta t_p = 10^{-5}s$, which has not converged yet. If however, a lower error is selected ($\epsilon_{err} = 10^{-4}$), the results obtained are much more similar to $\delta t_p = 10^{-6}s$ configuration. It is noticeable that the main difference in function of the time step strategy used in the deposition results is obtained in the smallest particles, which have a smaller relaxation time, and for that reason require a smaller time step.

Finally, the results obtained are compared with experimental data in figure 8.9.

In conclusion, using the adaptive time step strategy allows saving computational time and obtains very accurate results (compared to experimental data and more computational cost simulations). In addition, this option takes a weight off the user's shoulder as the particle time step selection is no longer in his/her hands.

To include CPU times, a single time step of this bend pipe experiment with 12.000 particles, without using adaptive strategy, consumes on average 0.6s. The adaptive time step strategy with higher errors requires 3 seconds, on average. The solution obtained with this last configuration and $\delta t_p = 10^{-4}s$ was close to non-adaptive strategy using $\delta t_p = 10^{-5}s$, thus 10 extra steps are needed, which implies a total of 6s, whereas with the adaptive is halved. These differences get larger if ϵ_{err} is decreased. In the case of the lowest error presented in figures 8.8 and 8.9, the time required on average was 4s, obtaining a result similar to $\delta t_p = 10^{-6}s$ which would need 60s.

8.4 Real patient simulation with brownian diffusivity

Next and last experiment uses a real patient nasal cavity and our goal is to validate the deposition of small particles affected by Brownian diffusion.

We consider a transient and unsteady airflow. A time window of $0.02s$ is chosen to compute the mean flow and the turbulence measures. This time window was taken as sufficiently long due to the fine temporal resolution available and short-scale transients.

Physiologically, the airflow through the respiratory system is driven by the pressure drop. Figure 8.10 presents the static pressure drop with Pressure

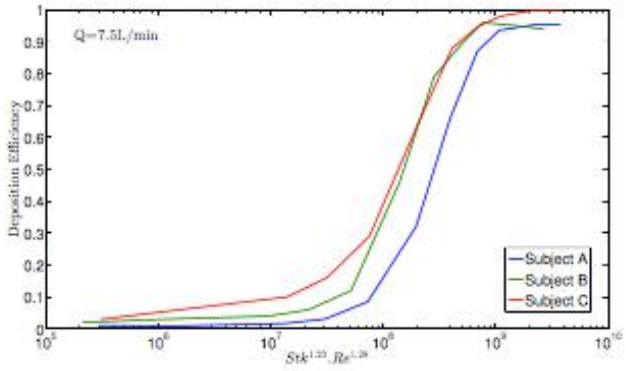
distribution in the human nasal cavity wall when inhalation flow rate is $20L/min$.

When the airflow is considered transient and unsteady, the mean velocity provides an overview of the dominant persistent flow features. Figure 8.11 shows the mean velocity in five different cross sections of the airway, see the location in figure 8.10.

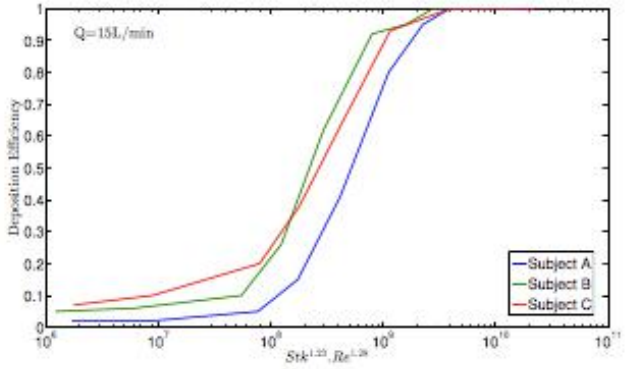
Nanoparticle deposition on the ideal wall condition, i.e. perfectly absorbing wall, are presented. Particles with in effective diameter range $1nm < d_p < 150nm$ are compared to the experimental data reported by (Cheng et al., 1995) and numerical result produced (Shi et al., 2008b). The configuration of the Brownian diffusion is done according to the results obtained in 7.5.1. According to this, the simulation will be done using a $\delta t = 10^{-5}s$ applying the diffusion over the displacement and using the proposed parallel pseudorandom numbers generator (PPRNG).

The final result is plotted in Figure 8.12 (a). We can observe good agreement with the results from the literature.

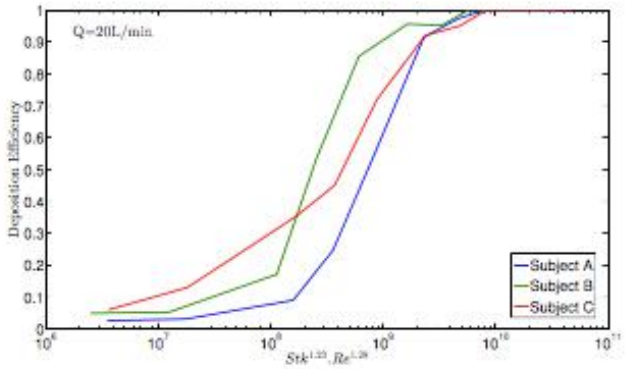
In addition, accurate particle tracking and deposition of nanoparticles give information of the precise location of deposition through the nasal cavity, Figure 8.12 (b).



(a)



(b)



(c)

Figure 8.5: Particle deposition efficiency in different real subjects (a), (b) and (c)

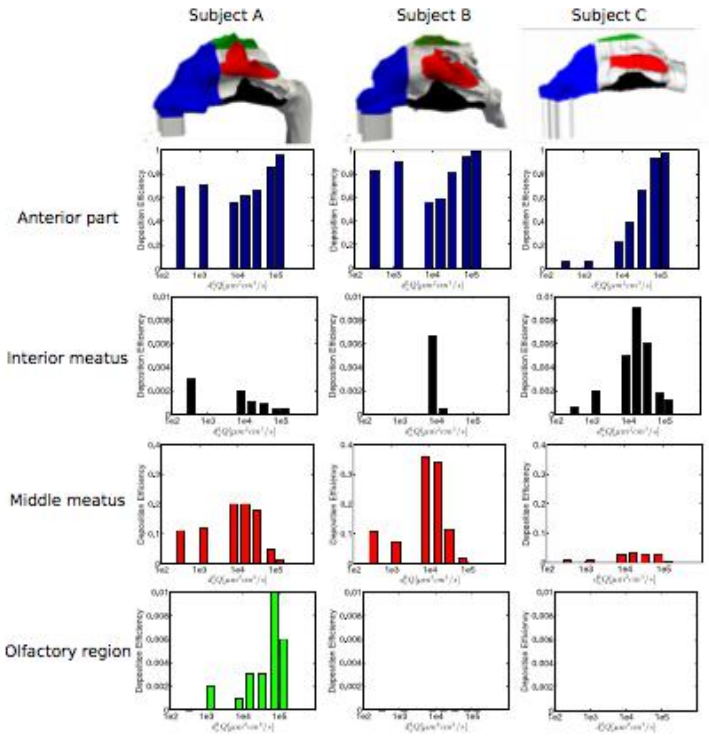


Figure 8.6: Particle deposition efficiencies in different critical regions for the three different subjects (a), (b) and (c)

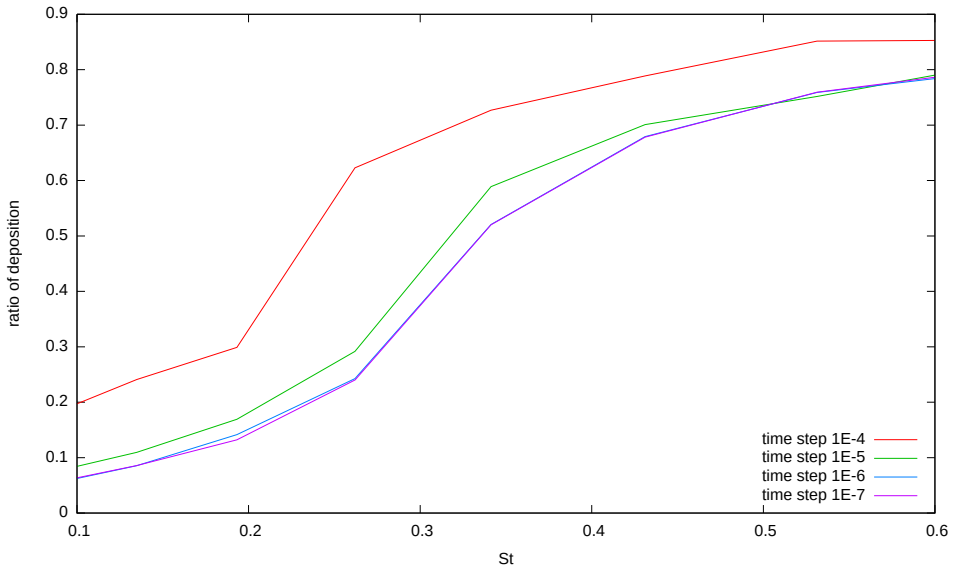


Figure 8.7: Comparison of deposition ratio in function of Stokes number obtained using different fixed time steps until reaching the convergence with $\delta t_p = 10^{-6}s$ in blue.

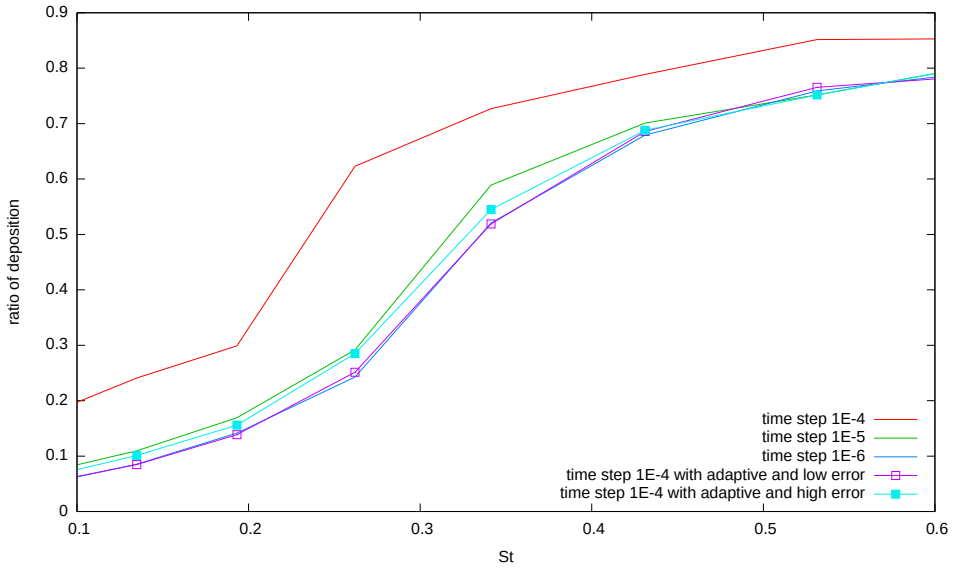


Figure 8.8: Deposition ratio in function of the Stokes number obtained using an adaptive time step compared to fixed time steps results.

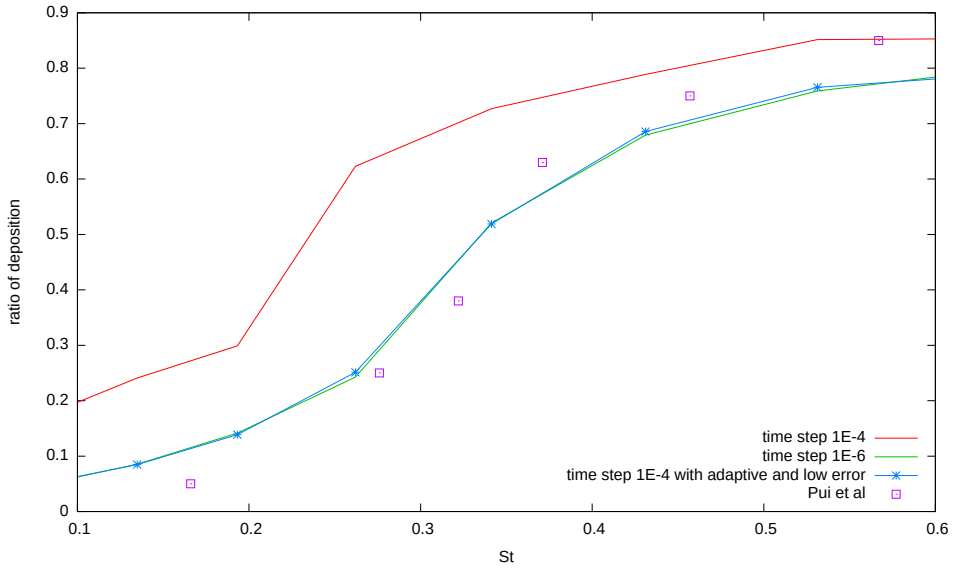


Figure 8.9: Deposition ratio in function of Stokes number obtained using computational simulations by using different time step strategies compared to experimental data (Pui et al., 1987).

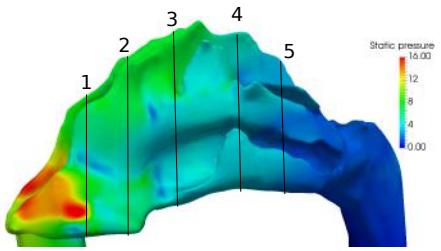


Figure 8.10: Pressure distribution in the human nasal cavity wall when inhalation flow rate is $20L/min$.

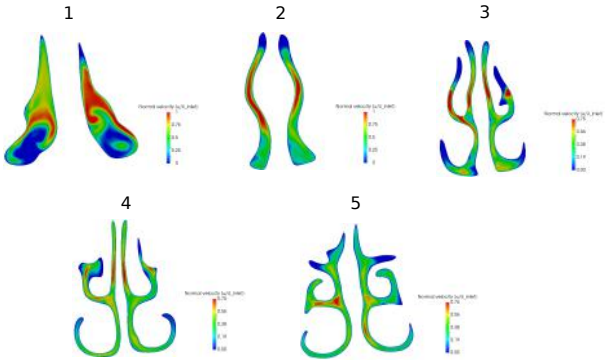
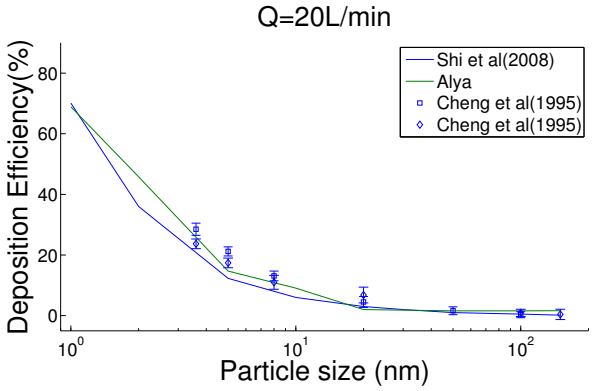
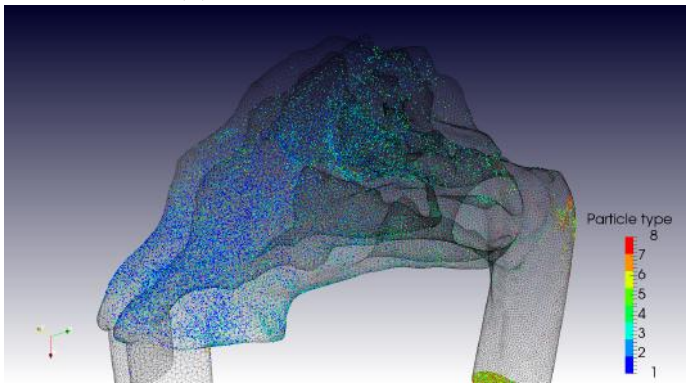


Figure 8.11: Velocity fields in human nasal cavity at a constant inlet flow rate of 20 L/min. Mean velocity contours in five selected slices, see Figure 8.10.



(a)



(b)

Figure 8.12: (a) Model validation of nanoparticle transport and deposition in a human nasal cavity. (b) Nanoparticle deposition in a human nasal cavity.

Chapter 9

Conclusions

The world is a stage and the play is badly cast.

O. Wilde

Summary: In this chapter, main conclusions achieved during this thesis and future lines of work are presented. In this thesis, a Lagrangian particle transport model, applied to respiratory system, has been developed. First, an efficient method to solve the point-location problem has been introduced. This is an essential beginning to locate and track a particle inside a mesh. Next, a study of the flow and forces affecting particles has been done and, after that, a versatile integration scheme within an adaptive time step has been shown. Once the main tools are built, the code is adapted to HPC parallelizing it. Finally, simulations applied to respiratory system and results obtained, comparing them to benchmarks, have been included to validate our algorithm.

9.1 Particle transport overview

Particle transport simulation is a very extensive topic which can involve many different applications, as for example the study of the respiratory system. Independently of this application, in this thesis we focused on the robustness of the integration scheme. The core of the work should not vary too much from one application to another. If the integration scheme is robust enough, it should converge and suit to any application, without being determined by particle velocities, accelerations or their size.

The inclusion of an adaptive time step is also important because can beef up the convergence event with abrupt changes of directions (e.g., small

particles in a turbulent flow), but mainly improves the accuracy of the results or, in other words, adaptive time step deeply improves the efficiency compared to a fix time step strategy, reducing the computational time given the same accuracy.

Finally, the parallelization scheme is necessary to be able to work with a supercomputer and focused on HPC. An hybrid parallelization using MPI tasks and OpenMP threads is proposed. On the one hand, MPI tasks divide the work in subdomains and once each particle has finished all the required calculations along a given time step δt, an *all–reduce* is done in order to count the total number of particles in the system. This process can become unbalanced if not all the particles require a similar computational time. On the other hand, OpenMP threads are casted in the main loop of the code, i.e., for each particle do whatever calculations are required. This process comes as a simple and efficient solution if only particles are computed, but it can get unbalanced when these particles are coupled with the fluid. The balance within this two parallel approaches is smartly obtained by using a DLB library.

Unlike the integration scheme, the parallelization approach could vary in the future, depending on the application. Including any new stochastic process or changing an existing one (e.g., Brownian motion) must be done carefully, as has been detailed on chapter 7. In addition, if future work implies particle interaction (in a deterministic way), such as collision, particle information from different subdomains should be required during the main loop calculation, which would unavoidably alter the existing parallelization scheme.

Of course, the choice of focusing on a single application like the respiratory system, has been helpful for an easier structure of the steps and objectives of this thesis. Firstly, reducing the number of physics affecting particles and being able to deeply study them and, secondly, allowing us to meet the concrete computational needs to make the code functional for this purpose. In addition, respiratory system application bestows a more biological character to this thesis, which can facilitate its comprehension because a familiar context for everyone is generated (breathing, inhalation, drug delivery...) and the numerous applications for medical purposes adds encouraging reasons to keep working and improving our developments.

9.2 Main goals achieved

This thesis has presented a method to simulate particles transport in a flow given an HPC context and applying it to the respiratory system airways. According to the main goals proposed at beginning of this thesis, the most relevant progress achieved are:

- Studying which are the main forces affecting particle transport inside respiratory system airways.
- According to involved forces, being able to neglect or not each one depending on the situation. Herein, the particle size can determine the most suitable configuration, but also how much computational cost we can assume.
- Modelling particle transport by using a robust integration scheme with an inner adaptive time step, capable to run very distinct types of particles and sizes.
- Parallelizing the code to make it suitable to HPC according to the different requirements of the flow and particles, by using a dynamic hybrid MPI tasks+OpenMP threads solution balanced by the DLB library. This approach has allowed us to simulate extremely expensive computational simulation fine meshes (up to 550 millions of elements) of the whole respiratory system, injecting millions of particles.
- Simulating respiratory system airways facing medical or pharmaceutical challenges and benchmarking our results.

Furthermore, during the development of aforementioned achieved goals, new or unexpected aims have been carried out, such as:

- During the parallelization of our code, we have found out that special attention must be taken into the parallelization of Brownian diffusion, specifically with an hybrid parallelization.
- During the thesis, a general problem has appeared recurrently. The point (or particle) test inclusion test presented different and new complexities when a new mesh or a new improvement was done. For this reason, a robust test inclusion test has been presented. It is important to highlight that this test has been specially modified to become suitable to higher order methods by adding a NR that allows to solve the ray intersection with the high order face, generating an innovative method for this purpose.

9.3 Future work

As future work, a deeper analysis must be done about lift force, because of its complexity in crossed terms of its Jacobian (using only the main diagonal may not be the best option when the lift force becomes important) and the requirement of a major development on SGS modellization. Coding the two way coupling opens a window for combustion models, although the most

suitable configuration to implement particle interaction should be studied too. Indeed, efficiently parallelizing millions of particle interactions (e.g., which can be provoked by particles from two different subdomains) is not straight forward.

In terms of respiratory system airways models, coupling flow-structure interaction would improve the precision of the simulations in the trachea. Herein, an improvement on lift force, above mentioned, near wall and for largest particles could also condition deposition results. Moreover, inclusion test for high order elements must be carefully studied with special attention in the convergence of the NR.

Finally, we feel absolutely confident about the possibility of using this same algorithm for the opposite scale (e.g., astronomical bodies movement and orbits). Another possibility is applying a different external field, such as an electromagnetic field, which would transport particles by Lorentz forces. This option makes feasible the application to fusion energy, simulating particle behaviour in tokamaks and stelllarators, as it is being done in current work.

Bibliography

Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.

Miguel de Cervantes Saavedra

A.Einstein. Ann. vol. 17, página 549, 1903.

Aluru, S. Lagged fibonacci random number generators for distributed memory parallel computers. *Journal of Parallel and Distributed Computing*, vol. 45(1), páginas 1 – 12, 1997. ISSN 0743-7315.

Arastoopour, H., Wang, C.-H. y Weil, S. A. Particle-particle interaction force in a dilute gas-solid system. *Chemical Engineering Science*, vol. 37(9), páginas 1379 – 1386, 1982. ISSN 0009-2509.

Artigues, A., Cucchietti, F. M., Montes, C. T., Vicente, D., Calmet, H., Marin, G., Houzeaux, G. y Vazquez, M. Scientific big data visualization: a coupled tools approach. *Supercomputing Frontiers and Innovations*, vol. 1(3), páginas 4–18, 2015.

Asgharian, B., Price, O. T., Schroeter, J. D., Kimbell, J. y Singal, M. A lung dosimetry model of vapor uptake and tissue disposition. *Inhalation toxicology*, vol. 24(3), páginas 182–193, 2012.

Aspden, A., Nikiforakis, N., Dalziel, S. y Bell, J. Analysis of implicit les methods. *Communications in Applied Mathematics and Computational Science*, vol. 3(1), páginas 103–126, 2009.

Babuska, I., Szabo, B. A. y Katz, I. N. The p-version of the finite element method. *SIAM journal on numerical analysis*, vol. 18(3), páginas 515–545, 1981.

Basu, S., Frank-Ito, D. O. y Kimbell, J. S. On computational fluid dynamics models for sinonasal drug transport: relevance of nozzle subtraction and nasal vestibular dilation. *arXiv preprint arXiv:1705.08989*, 2017.

BATES, A., CETTO, R., DOORLY, D., SCHROTER, R., TOLLEY, N. y COMERFORD, A. The effects of curvature and constriction on airflow and energy loss in pathological tracheas. *Respiratory Physiology & Neurobiology*, vol. 234(Supplement C), páginas 69 – 78, 2016. ISSN 1569-9048.

BATES, A. J., DOORLY, D. J., CETTO, R., CALMET, H., GAMBARUTO, A., TOLLEY, N., HOUZEAUX, G. y SCHROTER, R. Dynamics of airflow in a short inhalation. *Journal of the Royal Society Interface*, vol. 12, 2014. ISSN 1742-5689.

BOOZER, A. H. y KUO-PETRAVIC, G. Monte carlo evaluation of transport coefficients. *The Physics of Fluids*, vol. 24(5), páginas 851–859, 1981.

BRADSHAW, P. *An introduction to turbulence and its measurement: thermodynamics and fluid mechanics series*. Elsevier, 2013.

BUCKINGHAM, E. On physically similar systems; illustrations of the use of dimensional equations. *Physical review*, vol. 4(4), página 345, 1914.

BUCKLEY, A., HODGSON, A., WARREN, J., GUO, C. y SMITH, R. Size-dependent deposition of inhaled nanoparticles in the rat respiratory tract using a new nose-only exposure system. *Aerosol Science and Technology*, vol. 50(1), páginas 1–10, 2016.

CALMET, H., GAMBARUTO, A. M., BATES, A. J., VÁZQUEZ, M., HOUZEAUX, G. y DOORLY, D. J. Large-scale cfd simulations of the transitional and turbulent regime for the large human airways during rapid inhalation. *Computers in biology and medicine*, vol. 69, páginas 166–180, 2016.

CALMET, H., KLEINSTREUER, C., HOUZEAUX, G., KOLANJIYIL, A., LEHMKUHL, O., OLIVARES, E. y VÁZQUEZ, M. Subject-variability effects on micron particle deposition in human nasal cavities. *Journal of Aerosol Science*, vol. 115, páginas 12–28, 2018.

CANTWELL, C., SHERWIN, S., KIRBY, R. y KELLY, P. From h to p efficiently: Strategy selection for operator evaluation on hexahedral and tetrahedral elements. *Computers & Fluids*, vol. 43(1), páginas 23–28, 2011.

CHANG, T.-J., KAO, H.-M. y YAM, R. S.-W. Lagrangian modeling of the particle residence time in indoor environment. *Building and Environment*, vol. 62, páginas 55 – 62, 2013. ISSN 0360-1323.

CHENG, K.-H., CHENG, Y.-S., YEH, H.-C., GUILMETTE, R. A., SIMPSON, S. Q., YANG, Y.-H. y SWIFT, D. L. In vivo measurements of nasal airway dimensions and ultrafine aerosol deposition in the human nasal and oral airways. *Journal of Aerosol Science*, vol. 27(5), páginas 785–801, 1996.

CHENG, K.-H., CHENG, Y.-S., YEH, H.-C. y SWIFT, D. L. Deposition of ultrafine aerosols in the head airways during natural breathing and during simulated breath holding using replicate human upper airway casts. *Aerosol Science and Technology*, vol. 23(3), páginas 465–474, 1995.

CHENG, N.-S. y NGUYEN, H. T. Hydraulic radius for evaluating resistance induced by simulated emergent vegetation in open-channel flows. *Journal of hydraulic engineering*, vol. 137(9), páginas 995–1004, 2010.

CHENG, Y., HOLMES, T., GAO, J., GUILMETTE, R., LI, S., SURAKITBANHARN, Y. y ROWLINGS, C. Characterization of nasal spray pumps and deposition pattern in a replica of the human nasal airway. *Journal of Aerosol Medicine*, vol. 14(2), páginas 267–280, 2001.

CHENG, Y.-S., YEH, H. y SWIFT, D. Aerosol deposition in human nasal airway for particles 1nm to 20 μm: A model study. *Radiation Protection Dosimetry*, vol. 38(1-3), páginas 41–47, 1991.

CHUNG, J. y HULBERT, G. A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized alpha method. vol. 60, páginas 371–375, 1993.

CODDINGTON, P. D. Random number generators for parallel computers. *Computer Physics Communications*, vol. 13, 1997.

CROWE, C. T., SCHWARZKOPF, J. D., SOMMERFELD, M. y TSUJI, Y. *Multiphase flows with droplets and particles*. CRC press, 2011.

CSANADY, G. Turbulent diffusion of heavy particles in the atmosphere. *Journal of the Atmospheric Sciences*, vol. 20(3), páginas 201–208, 1963.

DANCKWERTS, P. Continuous flow systems. *Chemical Engineering Science*, vol. 2(1), páginas 1 – 13, 1953. ISSN 0009-2509.

DARQUENNE, C. y PAIVA, M. One-dimensional simulation of aerosol transport and deposition in the human lung. *Journal of applied Physiology*, vol. 77(6), páginas 2889–2898, 1994.

DEARDORFF, J. W. A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics*, vol. 41, páginas 453–480, 1970.

DHURIA, S. V., HANSON, L. R. y FREY, W. H. Intranasal delivery to the central nervous system: mechanisms and experimental considerations. *Journal of pharmaceutical sciences*, vol. 99(4), páginas 1654–1673, 2010.

DJUPESLAND, P. G. Nasal drug delivery devices: characteristics and performance in a clinical perspective?a review. *Drug delivery and translational research*, vol. 3(1), páginas 42–62, 2013.

DOISNEAU, F., DUPAYS, J., MURRONE, A., LAURENT, F. y MASSOT, M. Eulerian versus lagrangian simulation of unsteady two-way coupled coalescing two-phase flows in solid propellant combustion. *Comptes Rendus Mécanique*, vol. 341(1-2), páginas 44–54, 2013.

DREW, D. A. y LAHEY, R. T. *In Particulate Two-Phase Flow*. Butterworth-Heinemann. Boston, 1993.

DZIUGYS, A. y PETERS, B. An approach to simulate the motion of spherical and non-spherical fuel particles in combustion chambers. *Granular Matter*, vol. 3:4, páginas 231–266, 2001.

ELGHOBASHI, S. On predicting particle-laden turbulent flows. *Applied scientific research*, vol. 52(4), páginas 309–329, 1994.

FILIPPIDI, E., MICHAILIDOU, V., LOPPINET, B., RÃ¼HE, J. y FYTAS, G. Brownian diffusion close to a polymer brush. *Langmuir*, vol. 23(9), páginas 5139–5142, 2007.

FOLCH, A., COSTA, A. y MACEDONIO, G. A computational model for transport and deposition of volcanic ash. *Computers and Geosciences*, vol. 35, páginas 1334–1342, 2009.

FUNG, T. Numerical dissipation in time-step integration algorithms for structural dynamic analysis. *Structural Analysis and CAD*, vol. 5:3, página 167?180, 2003.

GAMBARUTO, A., OLIVARES, E., CALMET, H., HOUZEAUX, G., BATES, A. y DOORLY, D. Transport and deposition in the upper human airways during a sniff. páginas 13–16. 2014.

GANSER, G. H. A rational approach to drag prediction of spherical and nonspherical particles. *Powder Technology*, vol. 77, páginas 143–152, 1993.

GARCIA, G. J., BAILIE, N., MARTINS, D. A. y KIMBELL, J. S. Atrophic rhinitis: a cfd study of air conditioning in the nasal cavity. *Journal of applied physiology*, vol. 103(3), páginas 1082–1092, 2007.

GARCIA, G. J., SCHROETER, J. D. y KIMBELL, J. S. Olfactory deposition of inhaled nanoparticles in humans. *Inhalation toxicology*, vol. 27(8), páginas 394–403, 2015.

GARCIA, G. J., TEWKSBURY, E. W., WONG, B. A. y KIMBELL, J. S. Interindividual variability in nasal filtration as a function of nasal cavity geometry. *Journal of aerosol medicine and pulmonary drug delivery*, vol. 22(2), páginas 139–156, 2009a.

GARCIA, M., CORBALAN, J. y LABARTA, J. Lewi: A runtime balancing algorithm for nested parallelism. En *Parallel Processing, 2009. ICPP '09. International Conference on*, páginas 526–533. 2009b. ISSN 0190-3918.

GARCIA-GASULLA, M., HOUZEAUX, G., FERRER, R., ARTIGUES, A., LÓPEZ, V., LABARTA, J. y VÁZQUEZ, M. Task-based parallelization and dynamic load balance of finite element assembly. *J. Comput. Sci., In preparation*, 2018.

GARGALLO PEIRÓ, A. Validation and generation of curved meshes for high-order unstructured methods. 2014.

GHIRELLI, F. y LECKNER, B. Transport equation for the local residence time of a fluid. *Chemical Engineering Science*, vol. 59(3), páginas 513 – 523, 2004. ISSN 0009-2509.

GOSMAN, A. y LOANNIDES, E. Aspects of computer simulation of liquid-fueled combustors. *Journal of Energy*, vol. 7(6), páginas 482–490, 1983.

GRINSTEIN, F. F., MARGOLIN, L. G. y RIDER, W. J. *Implicit large eddy simulation: computing turbulent fluid dynamics*. Cambridge university press, 2007.

GUERRA, G., ZIO, S., CAMATA, J., ROCHINHA, F., ELIAS, R., PARAIZO, P. y COUTINHO, A. Numerical simulation of particle-laden flows by the residual-based variational multiscale method. *Int. J. Numer. Meth. Fluids*, vol. 73, páginas 729–749, 2013.

HABASHI, W. G. ET AL. Development of a second generation in-flight icing simulation code. *Journal of fluids engineering*, vol. 128(2), páginas 378–387, 2006.

HEYDER, J. Deposition of inhaled particles in the human respiratory tract and consequences for regional targeting in respiratory drug delivery. *Proceedings of the American Thoracic Society*, vol. 1(4), páginas 315–320, 2004.

HILBER, H., HUGHES, T. y TAYLOR, R. Improved numerical dissipation for the time integration algorithms in structural dynamics. *Earthquake Engineering and Structural Dynamics*, vol. 5, páginas 283–292, 1977.

HOUZEAUX, G., AUBRY, R., y VAZQUEZ, M. Extension of fractional step techniques for incompressible flows: The preconditioned orthomin(1) for the pressure schur complement. *Computers and Fluids*, vol. 44, páginas 297–313, 2011a.

HOUZEAUX, G., AUBRY, R. y VÁZQUEZ, M. Extension of fractional step techniques for incompressible flows: The preconditioned orthomin (1) for

the pressure schur complement. *Computers & Fluids*, vol. 44(1), páginas 297–313, 2011b.

HOUZEAUX, G. y CODINA, R. A chimera method based on a dirichlet/neumann (robin) coupling for the navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, vol. 192(31), páginas 3343–3377, 2003.

HOUZEAUX, G., GARCIA-GASULLA, M., CAJAS, J., ARTIGUESA, A., OLIVARES, E., LABARTA, J. y VÁZQUEZ, M. Dynamic load balance applied to particle transport in fluids. *INTERNATIONAL JOURNAL OF COMPUTATIONAL FLUID DYNAMICS*, vol. 30(6), páginas 408–418, 2016.

HOUZEAUX, G., VÁZQUEZ, M., AUBRY, R. y CELA, J. M. A massively parallel fractional step solver for incompressible flows. *Journal of Computational Physics*, vol. 228(17), páginas 6316–6332, 2009.

HUERTA, A., ROCA NAVARRO, F. J., ANGELOSKI, A. y PERAIRE GUITART, J. Are high-order and hybridizable discontinuous galerkin methods competitive. En *Oberwolfach Reports, scientific programme 2012*, páginas 28–30. 2012.

HUGHES, T. J. Multiscale phenomena: Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, vol. 127(1), páginas 387 – 401, 1995. ISSN 0045-7825.

ILLUM, L. Nasal drug delivery¿possibilities, problems and solutions. *Journal of controlled release*, vol. 87(1), páginas 187–198, 2003.

INTHAVONG, K., GE, Q., SE, C. M., YANG, W. y TU, J. Simulation of sprayed particle deposition in a human nasal cavity including a nasal spray device. *Aerosol Science and Technology*, vol. 42, páginas 100–113, 2011.

INTHAVONG, K., TIAN, Z., LI, H., TU, J., YANG, W., XUE, C. y LI, C. G. A numerical study of spray particle deposition in a human nasal cavity. *Aerosol Science and Technology*, vol. 40(11), páginas 1034–1045, 2006.

INTHAVONG, K., TIAN, Z., TU, J., YANG, W. y XUE, C. Optimising nasal spray parameters for efficient drug delivery using computational fluid dynamics. *Computers in biology and medicine*, vol. 38(6), páginas 713–726, 2008.

JANSEN, K., WHITING, C. y HULBER, G. A generalized alpha method for integrating the filtered navier-stokes equations with a stabilized finite element method. *International Journal for Numerical Methods in Engineering*, vol. 190, páginas 305–319, 2000.

Keeler, J. A., Patki, A., Woodard, C. R. y Frank-Ito, D. O. A computational study of nasal spray deposition pattern in four ethnic groups. *Journal of aerosol medicine and pulmonary drug delivery*, vol. 29(2), páginas 153–166, 2016.

Kelly, J., Asgharian, B., Kimbell, J. y Wong, B. Particle deposition in human nasal airway replicas manufactured by different methods. part i: Inertial regime particles. *Aerosol Science*, vol. 38, páginas 1072–1079, 2004a.

Kelly, J. T., Asgharian, B., Kimbell, J. S. y Wong, B. A. Particle deposition in human nasal airway replicas manufactured by different methods. part ii: Ultrafine particles. *Aerosol science and technology*, vol. 38(11), páginas 1072–1079, 2004b.

Kesavan, J., Bascom, R., Laube, B. y Swift, D. L. The relationship between particle deposition in the anterior nasal passage and nasal passage characteristics. *Journal of aerosol medicine*, vol. 13(1), páginas 17–23, 2000.

Kesavanathan, J. y Swift, D. L. Human nasal passage particle deposition: the effect of particle size, flow rate, and anatomical factors. *Aerosol Science and Technology*, vol. 28(5), páginas 457–463, 1998.

Keyhani, K., Scherer, P. W. y Mozell, M. M. A numerical model of nasal odorant transport for the analysis of human olfaction. *Journal of Theoretical Biology*, vol. 186(3), páginas 279–301, 1997.

Kim, J. W., Dennis, P. G., Sankar, L. N. y Kreeger, R. E. Ice accretion modeling using an eulerian approach for droplet impingement. *AIAA Paper 2013*, vol. 246, 2013.

Kimbell, J. S., Segal, R. A., Asgharian, B., Wong, B. A., Schroeter, J. D., Southall, J. P., Dickens, C. J., Brace, G. y Miller, F. J. Characterization of deposition from nasal spray devices using a computational fluid dynamics model of the human nasal passages. *Journal of aerosol medicine*, vol. 20(1), páginas 59–74, 2007.

Kleinstreuer, C., Feng, Y. y Childress, E. Drug-targeting methodologies with applications: a review. *World Journal of Clinical Cases: WJCC*, vol. 2(12), página 742, 2014.

Kleinstreuer, C. y Zhang, Z. Airflow and particle transport in the human respiratory system. *Annual Review of Fluid Mechanics*, vol. 42, páginas 301–334, 2010.

KOLANJIYIL, A. V. y KLEINSTREUER, C. Computationally efficient analysis of particle transport and deposition in a human whole-lung-airway model. part i: Theory and model validation. *Computers in biology and medicine*, vol. 79, páginas 193–204, 2016.

KOLANJIYIL, A. V., KLEINSTREUER, C. y SADIKOT, R. T. Computationally efficient analysis of particle transport and deposition in a human whole-lung-airway model. part ii: Dry powder inhaler application. *Computers in biology and medicine*, vol. 84, páginas 247–253, 2017.

KULICK, J. D., FESSLER, J. R. y EATON, J. K. Particle response and turbulence modification in fully developed channel flow. *Journal of Fluid Mechanics*, vol. 277, páginas 109–134, 1994.

KUNUTH, D. The art of computer programming vol. 2 seminumerical algorithms. 1998.

KUZNETSOV, A. y NIELD, D. Natural convective boundary-layer flow of a nanofluid past a vertical plate. *International Journal of Thermal Sciences*, vol. 49(2), páginas 243 – 247, 2010. ISSN 1290-0729.

LECUYER, P., MUNGER, D., ORESHKIN, B. y SIMARD, R. Random numbers for parallel computers: Requirements and methods, with emphasis on {GPUs}. *Mathematics and Computers in Simulation*, vol. 135, páginas 3 – 17, 2017. ISSN 0378-4754. Special Issue: 9th Seminar on Monte Carlo Methods.

LI, A. y AHMADI, G. Dispersion and deposition of spherical particles from point sources in a turbulent channel flow. *Aerosol science and technology*, vol. 16(4), páginas 209–226, 1992.

LÖHNER, R. Error and work estimates for high-order elements. *International Journal for Numerical Methods in Fluids*, vol. 67(12), páginas 2184–2188, 2011.

LONGEST, P. W. y XI, J. Computational investigation of particle inertia effects on submicron aerosol deposition in the respiratory tract. *Journal of Aerosol Science*, vol. 38(1), páginas 111–130, 2007.

LOTH, E. Numerical approaches for motion of dispersed particles, droplets and bubbles. *Progress in Energy and Combustion Science*, vol. 26(3), páginas 161–223, 2000.

MACINNES, J. y BRACCO, F. Stochastic particle dispersion modeling and the tracer-particle limit. *Physics of Fluids A: Fluid Dynamics*, vol. 4(12), páginas 2809–2824, 1992.

McCUTCHAN, J. W. y TAYLOR, C. L. Respiratory heat exchange with varying temperature and humidity of inspired air. *Journal of Applied Physiology*, vol. 4(2), páginas 121–135, 1951.

MEI, R., KLAUSNER, J. F. y LAWRENCE, C. J. A note on the history force on a spherical bubble at finite reynolds number. *Physics of Fluids*, vol. 6(1), páginas 418–420, 1994.

MISTRY, A., STOLNIK, S. y ILLUM, L. Nanoparticles for direct nose-to-brain delivery of drugs. *International journal of pharmaceutics*, vol. 379(1), páginas 146–157, 2009.

MYERS, T. G. Extension to the messinger model for aircraft icing. *AIAA*, vol. 39:2, páginas 211–218, 2001.

NEUMANN, P. y ROHRMANN, T. Lattice boltzmann simulations in the slip and transition flow regime with the peano framework. *Open Journal of Fluid Dynamics*, vol. 2(3), página 101, 2012.

NEWMARK, N. M. A method of computation for structural dynamics. *Journal of Engineering Mechanics*, vol. 85, páginas 67–94, 1959.

NIELD, D. y KUZNETSOV, A. The cheng-minkowycz problem for natural convective boundary-layer flow in a porous medium saturated by a nanofluid. *International Journal of Heat and Mass Transfer*, vol. 52(25), páginas 5792 – 5795, 2009. ISSN 0017-9310.

NOROUZI, H. R., ZARGHAMI, R., SOTUDEH-GHAREBAGH, R. y MOSTOUFI, N. *Coupled CFD-DEM Modeling: Formulation, Implementation and Application to Multiphase Flows*. John Wiley & Sons, 2016.

PARK, K. C. An improved stiffly stable method for direct integration of nonlinear structural dynamic equations. *Journal of Applied Mechanics*, vol. 42(2), páginas 464–470, 1975.

POZORSKI, J. y APTE, S. V. Filtered particle tracking in isotropic turbulence and stochastic modeling of subgrid-scale dispersion. *International Journal of Multiphase Flow*, vol. 35(2), páginas 118 – 128, 2009. ISSN 0301-9322.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T. y FLANNERY, B. P. Numerical recipes in fortran 77, the art of scientific computing. 1992.

PUI, D. Y. H., ROMAY-NOVAS, F. y LIU, B. Y. H. Experimental study of particle deposition in bends of circular cross section. *Aerosol Science and Technology*, vol. 7, páginas 301–315, 1987.

RAMSEY, S. D., POTTER, K. y HANSEN, C. Ray bilinear patch intersections. *Journal of Graphics Tools*, vol. 9(3), páginas 41–47, 2004.

RASMUSSEN, T. R., ANDERSEN, A. y PEDERSEN, O. Particle deposition in the nose related to nasal cavity geometry. *Rhinology*, vol. 38(3), páginas 102–107, 2000.

RENNIE, C. E., GOUDER, K. A., TAYLOR, D. J., TOLLEY, N. S., SCHROTER, R. C. y DOORLY, D. J. Nasal inspiratory flow: at rest and sniffing. En *International forum of allergy & rhinology*, vol. 1, páginas 128–135. Wiley Online Library, 2011.

REYNOLDS, O. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical Transactions of the Royal Society of London. A*, vol. 186, páginas 123–164, 1895. ISSN 02643820.

RHEE, J. S., PAWAR, S. S., GARCIA, G. J. y KIMBELL, J. S. Toward personalized nasal surgery using computational fluid dynamics. *Archives of facial plastic surgery*, vol. 13(5), páginas 305–310, 2011.

SAEZ, X., SOBA, A., CELA, J. M., SANCHEZ, E. y CASTEJON, F. Particle-in-cell algorithms for plasma simulations on heterogeneous architectures. En *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, páginas 385–389. IEEE, 2011.

SAFFMAN, P. G. The lift on a small sphere in a slow shear flow. *Journal of Fluid Mechanics*, vol. 22, páginas 385–400, 1965.

SCHROETER, J. D., GARCIA, G. J. y KIMBELL, J. S. A computational fluid dynamics approach to assess interhuman variability in hydrogen sulfide nasal dosimetry. *Inhalation toxicology*, vol. 22(4), páginas 277–286, 2010.

SCHROETER, J. D., GARCIA, G. J. y KIMBELL, J. S. Effects of surface smoothness on inertial particle deposition in human nasal models. *Journal of aerosol science*, vol. 42(1), páginas 52–63, 2011.

SCHROETER, J. D., KIMBELL, J. S. y ASGHARIAN, B. Analysis of particle deposition in the turbinate and olfactory regions using a human nasal computational fluid dynamics model. *Journal of Aerosol Medicine*, vol. 19(3), páginas 301–313, 2006.

SCHROETER, J. D., KIMBELL, J. S., GROSS, E. A., WILLSON, G. A., DORMAN, D. C., TAN, Y.-M. y CLEWELL III, H. J. Application of physiological computational fluid dynamics models to predict interspecies nasal dosimetry of inhaled acrolein. *Inhalation toxicology*, vol. 20(3), páginas 227–243, 2008.

SCHROETER, J. D., TEWKSBURY, E. W., WONG, B. A. y KIMBELL, J. S. Experimental measurements and computational predictions of regional particle deposition in a sectional nasal model. *Journal of aerosol medicine and pulmonary drug delivery*, vol. 28(1), páginas 20–29, 2015.

SCHWEIZERHOF, K., NEUMANN, J. y KIZIO, S. On time integration error estimation and adaptive time stepping in structural dynamics. *Proceedings in Applied Mathematics and Mechanics*, vol. 4, páginas 35–38, 2004a.

SCHWEIZERHOF, K., NEUMANN, J. y KIZIO, S. Robust time integration schemes for durability analyses. 2004b.

SEGAL, R. A., KEPLER, G. M. y KIMBELL, J. S. Effects of differences in nasal anatomy on airflow distribution: a comparison of four individuals at rest. *Annals of biomedical engineering*, vol. 36(11), páginas 1870–1882, 2008.

SHI, H., KLEINSTREUER, C. y ZHANG, Z. Modeling of inertial particle transport and deposition in human nasal cavities with wall roughness. *Journal of Aerosol Science*, vol. 38(4), páginas 398–419, 2007a.

SHI, H., KLEINSTREUER, C. y ZHANG, Z. Modeling of inertial particle transport and deposition in human nasal cavities with wall roughness. *Aerosol Science*, vol. 38, páginas 398–419, 2007b.

SHI, H., KLEINSTREUER, C. y ZHANG, Z. Dilute suspension flow with nanoparticle deposition in a representative nasal airway model. *Physics of Fluids*, vol. 20(1), página 013301, 2008a.

SHI, H., KLEINSTREUER, C. y ZHANG, Z. Dilute suspension flow with nanoparticle deposition in a representative nasal airway model. *Physics of Fluids*, vol. 20(1), página 013301, 2008b.

SHIROLKAR, J., COIMBRA, C. y MCQUAY, M. Q. Fundamental aspects of modeling turbulent particle dispersion in dilute flows. *Progress in Energy and Combustion Science*, vol. 22(4), páginas 363–399, 1996.

SMAGORINSKY, J. General Circulation Experiments with the Primitive Equations. *Monthly Weather Review*, vol. 91, página 99, 1963.

SOVINEC, C., GLASSER, A., GIANAKON, T., BARNES, D., NEBEL, R., KRUGER, S., SCHNACK, D., PLIMPTON, S., TARDITI, A., CHU, M. ET AL. Nonlinear magnetohydrodynamics simulation using high-order finite elements. *Journal of Computational Physics*, vol. 195(1), páginas 355–386, 2004.

SRINIVASAN, A., MASCAGNI, M. y CEPERLEY, D. Testing parallel random number generators. *Parallel Computing*, vol. 29(1), páginas 69 – 94, 2003. ISSN 0167-8191.

STURM, R. Theoretical diagnosis of emphysema by aerosol bolus inhalation. *Annals of translational medicine*, vol. 5(7), 2017.

SZABO, B. A. y BABUSKA, I. *Finite element analysis*. John Wiley & Sons, 1991.

TEIJEIRO, C., SUTMANN, G., TABOADA, G. y TOURIO, J. Parallel brownian dynamics simulations with the message-passing and pgas programming models. *Computer Physics Communications*, vol. 184(4), páginas 1191 – 1202, 2013. ISSN 0010-4655.

TEITZEL, C., GROSSO, R. y ERTL, T. Efficient and reliable integration methods for particle tracing in unsteady flows on discrete meshes. En *Visualization in Scientific Computing?97*, páginas 31–41. Springer, 1997.

THORNE, R., PRONK, G., PADMANABHAN, V. y FREY, W. N. Delivery of insulin-like growth factor-i to the rat brain and spinal cord along olfactory and trigeminal pathways following intranasal administration. *Neuroscience*, vol. 127(2), páginas 481–496, 2004.

THORNE, R. G., EMORY, C. R., ALA, T. A. y FREY, W. H. Quantitative analysis of the olfactory pathway for drug delivery to the brain. *Brain research*, vol. 692(1), páginas 278–282, 1995.

TURTON, R. y LEVENSPIEL, O. A short note on the drag correlation for spheres. *Powder Technology*, vol. 47(1), páginas 83 – 86, 1986. ISSN 0032-5910.

UDWADIA, F. E. y KALABA, R. E. *Analytical dynamics: a new approach*. Cambridge University Press, 2007.

VÁZQUEZ, M., HOUZEAUX, G., KORIC, S., ARTIGUES, A., AGUADO-SIERRA, J., ARÍS, R., MIRA, D., CALMET, H., CUCCHIETTI, F., OWEN, H., TAHA, A., BURNESS, E. D., CELA, J. M. y VALERO, M. Alya: Multiphysics engineering simulation towards exascale. *J. Comput. Sci.*, vol. 14, páginas 15–27, 2016.

VILLEDIEU, P., TRONTIN, P., GUFFOND, D. y BOBO, D. Sld lagrangian modeling and capability assessment in the frame of onera 3d icing suite. En *4th AIAA Atmospheric and Space Environments Conference*, página 3132. 2012.

VOS, P. E., SHERWIN, S. J. y KIRBY, R. M. From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low-and high-order discretisations. *Journal of Computational Physics*, vol. 229(13), páginas 5161–5181, 2010.

Wang, Z. J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H. T. et al. High-order cfd methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, vol. 72(8), páginas 811–845, 2013.

Wilson, L. y Huang, T. The influence of shape on the atmospheric settling velocity of volcanic ash particles. *Earth and Planetary Science Letters*, vol. 44(2), páginas 311 – 324, 1979. ISSN 0012-821X.

Wood, W., Bossak, M. y Zienkiewicz, O. An alpha modification of newmark's methodg. *International Journal for Numerical Methods in Engineering*, vol. 15, páginas 1562–1566, 1981.

Yang, H., Fan, M., Liu, A. y Dong, L. General formulas for drag coefficient and settling velocity of sphere based on theoretical law. *International Journal of Mining Science and Technology*, vol. 25(2), páginas 219–223, 2015.

Zhang, Z. y Cheng, Q. Comparison of the eulerian and lagrangian methods for predicting particle transport in enclosed spaces. *Atmospheric Environment*, vol. 41, páginas 5236–5248, 2007.

Zhang, Z. y Kleinstreuer, C. Computational analysis of airflow and nanoparticle deposition in a combined nasal–oral–tracheobronchial airway model. *Journal of Aerosol Science*, vol. 42(3), páginas 174–194, 2011.

Zhao, B., Zhang, Y., Li, X., Yang, X. y Huang, D. Comparison of indoor aerosol particle concentration and deposition in different ventilated rooms by numerical method. *Building and Environment*, vol. 39(1), páginas 1 – 8, 2004. ISSN 0360-1323.

Zheng, X. y Silber-Li, Z. The influence of saffman lift force on nanoparticle concentration distribution near a wall. *Applied physics letters*, vol. 95(12), página 124105, 2009.