Spring 2018

Modelling of oxygen diffusion in cork

Swapnil Shivputra Rane
New Jersey Institute of Technology

ABSTRACT

MODELLING OF OXYGEN DIFFUSION IN CORK

by
Swapnil Shivputra Rane

The primary objective of this project is to better understand gas diffusion through natural cork structures by comparing simulations of gas diffusion through natural cork to experimental measurements of oxygen gas diffusion through the same structure. The cork's internal structure is measured using x-ray computed tomography (CT). From the measured internal structure, a simple mapping is assumed converting the measured x-ray computed tomography images into a 3D map of local diffusion coefficient. Specifically, it is assured that the x-ray CT pixel value is inversely proportional to the diffusion coefficient of oxygen through the cork. A diffusion model for the cork structure was developed using finite commercial finite element method simulation software and the necessary boundary conditions. The primary intention is to create model which can help in understanding the ingress of oxygen through cork. By comparing simulation and experimental results, the role of the cork's internal structure in gas diffusion can be better understood.

.

MODELLING OF OXYGEN DIFFUSION IN CORK

by
Swapnil Shivputra Rane

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mechanical Engineering

Department of Mechanical and Industrial Engineering

May 2018

APPROVAL PAGE

MODELLING OF OXYGEN DIFFUION IN CORK

Swapnil Shivputra Rane

Dr. John F. Federici, Thesis Advisor Date
Distinguished Professor of Department of Physics, NJIT

Dr. Dibakar Datta, Committee Member Date
Assistant Professor of Mechanical and Industrial Engineering, NJIT

Dr. Cristiano Luis Dias, Committee Member Date
Assistant Professor of Physics, NJIT

BIOGRAPHICAL SKETCH

Author: Swapnil Shivputra Rane

Degree: Master of Science in Mechanical Engineering

Date: May 2018

Undergraduate and Graduate Education:

- Master of Science in Mechanical Engineering,
  New Jersey Institute of Technology, Newark, NJ, USA 2018

- Bachelor of Engineering in Mechanical Engineering,
  University of Mumbai, Mumbai, MH, India 2016

Major: Mechanical Engineering

This research is a result of group work by my parents, sisters, friends and me. They have helped me not only for my research, but also for activities which has led me to complete this research. I would like to thank my sister's Snehal and Anjali for their king support. Without the help and sacrifice of my mom and dad, it would have not been possible for me to pursue my master's degree. I'm grateful for my friends Pramod Hanagandi, Prasad Bhatambarekar, Karan Modi for their support. I appreciate the support by Sanjana Patil for helping me with Excel and building my PowerPoint presentation.

ACKNOWLEDGMENT

This work is supported by Drs. John F. Federici and Cristiano Dias (Department of Physics) from New Jersey Institute of Technology and Drs. Herli Surjanhata and Dibakar Datta (Department of Mechanical and Industrial Engineering) from New Jersey Institute of Technology. The use of x-ray tomography data from UAVision in collaboration with Amorium & Irmaos, SA (Portugal) is gratefully acknowledged.

TABLE OF CONTENTS

TABLE OF CONTENTS

(Continued)

LIST OF TABLES

Table **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

LIST OF SYMBOLS

d	Diameter
L	Length
D	Diffusion co-efficient
t	Time
μ	Dynamic viscosity
D_{mean}	Mean diffusion coefficient
B_{mean}	Mean pixel value
C	Concentration

LIST OF DEFINITIONS

Diffusion co-efficient	"Diffusivity or diffusion coefficient is a proportionality constant between the molar flux due to molecular diffusion and the gradient in the concentration of the species (or the driving force for diffusion). Diffusivity is encountered in Fick's law and numerous other equations of physical chemistry." from Wikipedia.
Mass flow rate	The amount of mass flow per unit time

CHAPTER 1

INTRODUCTION TO COMPUTATIONAL FLUID DYNAMICS

Computational fluid dynamics (CFD) is a branch of fluid mechanics that uses numerical analysis and mathematical equations to solve and analyze problems involving fluid flow, heat transfer, and other related physical processes. Computer software is used to perform the calculations required to simulate the interaction of liquids and gases with surfaces defined by boundary conditions. The equations of fluid flow can be solved over a specific region with known boundary conditions- 3D properties of the fluid including flows, flux, as well as mass and heat transfer can be calculated.

1.1 History of CFD

Computers have always been used to solve complex mathematical problems with much less labor, time and cost as compared to similar calculations performed by humans. From the mid-1970s, complex mathematical problems were solved by using general purpose CFD tools. In the early 1980s, powerful computers were developed which were able to handle increasingly more complex problems but required large amounts of time to set up simulations. So, CFD was mostly used in high end research. [6]

Recent advances in computing technology, powerful graphics and interactive 3D models have made the process of creating CFD model and analyzing results with much less labor intensive, reducing time and, hence, cost. Advanced CFD software such as ANSYS, OpenFOAM, COMSOL have tools and features that solve problems of flow and analysis.

Because of these factors, CFD is now a well-established industrial design tool, which helps to reduce design time scales and improve fluid flow processes. CFD is the alternative to scale model testing with variations on simulation done quickly, which offers obvious advantages.

1.2 The Mathematics of CFD

Equations that describe the processes of momentum, heat and mass transfer are known as Navier-Stokes equations. These partial differential equations were derived in the early 19^{th} century and have no known analytical solution but can be discretized and solved numerically. There are many different solution methods that are used in CFD codes. But, the most commonly used solution method is known as the finite volume technique. This technique divides the region of interest into small sub-regions, called control volumes. Equations such as Navier-Stokes fluid flow are solved for these control volumes and the values of unknown variables can be solved. [7]

1.3 Uses of CFD

CFD can be used to solve various design solutions and performance of a component at the design stage to find a new improved design solution. For example, mass flow of a pipe system can be solved using CFD software, to control the flow of fluid through each pipe system. Figure 1.1 shows a schematic pipe system through which liquid can flow with three outlets with different diameters of the piping system according to the needs at the working location.

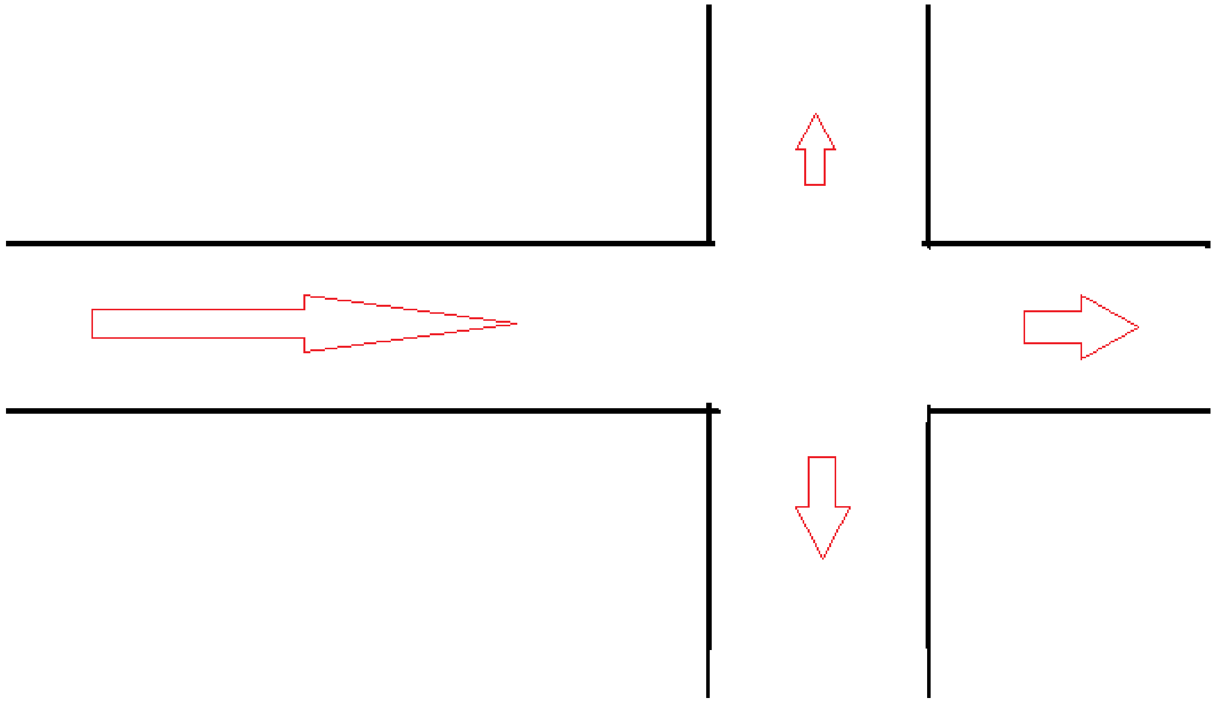

Figure 1.1 Sample flow problem.

The methodology of performing a CFD simulation is carried out in four steps—

1. Generating the Geometry/Mesh

This stage is also known as the pre-processing stage. The first step is to identify the region of interest and then define the geometry or import model geometry. The mesh is to be created then, boundary conditions and fluid properties are to be defined in the preprocessor. Now-a-days the pre-processor stage is mostly automated. In ANSYS CFX, as an example, the geometry can be imported from major CAD (Computer Aided Design) software packages and the mesh is generated automatically. Post-processor produces all the necessary files required by the user which may contain velocity, pressure and any other variables throughout the region of interest. Once the results are obtained, design modifications can be easily applied to obtain the required results or effects in the region if interest.

2. Physics of the model

Fluid properties and boundary conditions are specified. The model to be included in the simulation is automatically imported from the previous stage. The boundary conditions like pressure, velocity, mass flow rate, heat flux, temperature, type of flow are applied to the model for analysis.

3. Solving the CFD Problem

The component known as Solver performs the next step at this stage in a non-interactive/batch process. The steps performed by solver are as follows:

i. For each control volume in the region of interest, partial differential equations are integrated. For example, applying mass or momentum equation to each control volume. Equations such as [9] where u is the fluid velocity, p is the fluid pressure, ρ is the fluid density, and μ is the fluid dynamic viscosity.

$$\underbrace{\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)}_{1}=\underbrace{-\nabla p}_{2}+\underbrace{\nabla\cdot\left(\mu(\nabla\mathbf{u}+(\nabla\mathbf{u})^{T})-\frac{2}{3}\mu(\nabla\cdot\mathbf{u})\mathbf{I}\right)}_{3}+\underbrace{\mathbf{F}}_{4} \tag{1.1}$$

These equations are always solved together with the continuity equation:

$$\frac{\partial \rho}{\partial t}+\nabla\cdot(\rho\mathbf{u})=0 \tag{1.2}$$

ii. By getting the known values from the user, the unknown terms in these equations are solved.

iii. The algebraic equations are solved iteratively.

The degree to which the final solution is close to the exact solution depends on several factors such as size and shape of control volume. The solver produces a result file that is then passed to the post-processor.

4. Postprocessor- Visualize the Results.

   To analyze, visualize and present the results interactively to better understand the solution obtained from the CFD calculations, features of postprocessor are mainly used. Some of the important features of postprocessor are:

   - Visualize design geometry and control volume.
   - Plot the direction and magnitude of flows.
   - Visualize the variation in scalar variable like temperature, pressure and speed in the domain.
   - Animation of flow inside the regime.
   - Charts showing graphical plots of required variables.
   - Report file of the complete analysis.

CHAPTER 2

INTRODUCTION TO ANSYS CFX

There are several commercial CFD software packages which could be used to solve for the diffusion of gas through a natural cork structure. As an example of one such software package, ANSYS CFX is briefly described in this chapter.

2.1 Overview of ANSYS CFX

ANSYS CFX is a powerful Computational Fluid Dynamics (CFD) software with preprocessing and post processing capabilities. [10] Some of the features are:

- Reliable and robust coupled solver.
- Presentation of fully integrated problem definition, analysis and results.
- Capable of modelling:
    - Porous media flow.
    - Subsonic, transonic and supersonic flows.
    - Laminar and turbulent flows
    - Heat transfer and thermal radiation
    - Non-Newtonian flows
    - Buoyancy
    - Multiphase flows
    - Combustion
    - Particle Tracking

2.2 The Structure of ANSYS CFX

In Figure 2.1 below, a flow chart for the operation of ANSYS CFX is shown. In this, section, the operation of each of the step is briefly described.

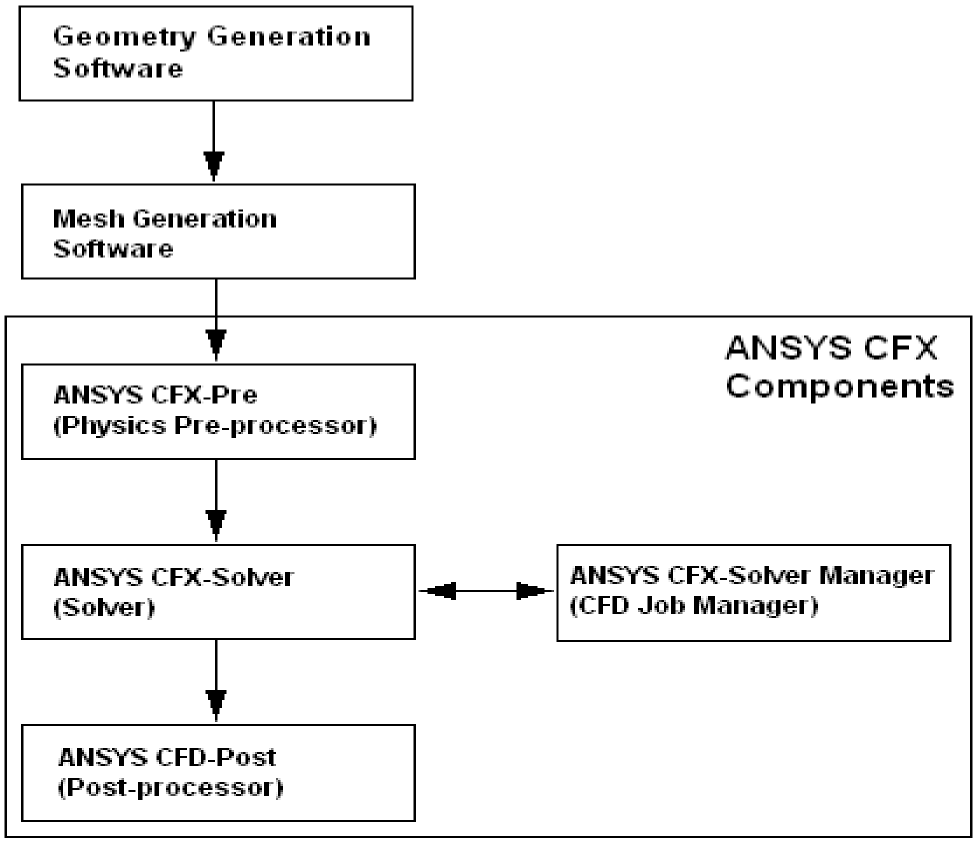

Figure 2.1 Structure of ANSYS CFX.
Source: ANSYS Documentation 17.0

- Geometry generation software.

  ANSYS has two inbuilt software for generating the required geometry for analysis known as Space Claim and Design Modeler. CFX also supports importing the geometry from other third-party software's like SolidWorks and Creo.

- Mesh generation software.

  The general-purpose, intelligent and automated high-performance preinstalled meshing software is well suited for simple and complex geometry with a single mouse click for all parts. Experts can use the fine mesh option to generate more accurate results at the expense of time and energy for computing.

- CFX-Pre

  It is the next-generation physics pre-processor used to define simulations. Capabilities like importing multiple mesh, allowing each section to use the most appropriate mesh. Multiple configuration and control facilitates the setup of complex simulations, such as internal combustion engines with evolving parts and physics.

- CFX-Solver

  Problem specification generated in CFX-Pre-are solved in this solver. By using the coupled solver, which solves all the hydrodynamic equations as a single system to get faster results as compared to the traditional segregated solver to converge the flow solution.

- ANSYS CFX-Solver manager.

  This module provides greater control over the CFD task. Functions include specify input files to CFX-solver, start/stop the program, monitor the solution, setup the CFX-solver for calculations. Although the required convergence level depends on the model and on your requirements, the following guidelines regarding RMS residual levels may be helpful.

  - Values larger than 1e-4 may be sufficient to obtain a qualitative understanding of the flow field.

  - 1e-4 is relatively loose convergence but may be sufficient for many engineering applications. The default target RMS residual value is 1e-4.

  - 1e-5 is good convergence, and usually sufficient for most engineering applications.

- 1e-6 or lower is very tight convergence, and occasionally required for geometrically sensitive problems. It is often not possible to achieve this level of convergence, particularly when using a single precision solver.

- CFD Post

  State-of-the-art interactive graphics software for post processing of the results. Features include report generation, plotting of variable, user-defined variables, function value.

CHAPTER 3

INTRODUCTION TO CORK

3.1 History of Cork.

For many centuries, cork has been used for its natural, renewable, sustainable properties. As a result, the scientific literature on cork is extensive. Cork bottle stoppers have been found in ancient tombs in Egypt and Greece. Romans also used cork in variety of ways, like life jackets for fisherman. Though glass bottles were invented in the fifteenth century, a revolutionary crown cap - a metal lid lined with a disk of natural cork commonly known as bottle cap - was invented in year 1892.

Around 1890, a German company developed a process for adding a clay binder to cork particles and producing sheets of agglomerated (composite) cork for use as insulation. The following year, Americans developed a technique for producing pure-cork agglomerated cut out of waste material. Since then, many other techniques have been developed to produce cork compounds with a variety of properties and uses. [10]

3.2 Natural Occurrence of Cork

Cork [11] is the bark of the cork oak (*Quercus suber L.*). The cork oak is one of the nature's most extraordinary trees which is harvested to produce cork, usually every 9 to 12 years depending on the culture region to produce cork. Cork Oak flourishes only in specific regions of Western Mediterranean countries (Portugal, Spain, Southern France, part of Italy, North Africa) and China. The plant requires huge amount of sunlight and humidity but unusually low rainfall. Europe has 80% of the market share in cork production and exportation. Portugal is the major contributor to this share about three-quarters of all cork.

The cork oak tree has a thick, insulating bark which helps the tree survive in the hot and dry climates. The insulating bark, due to the honeycomb like structures of the cork cells, is a natural barrier to the diffusion of water enabling the tree to remain hydrated even in the warm and dry climates near the Mediterranean Sea. The insulating bark enables the cork oak to be well adapted to survive. [11]

3.3 The Manufacturing Process

No trees are cut down, which makes it a prime example of sustainability and interdependence between the cork industry and the environment. It takes each cork oak 25 years before it can be stripped for the first time and it is only from the third harvest that the cork has reached the high standard of quality for cork production. This means that over 40 years of tree growth is required to produce high-quality cork stoppers from a cork oak. It is a slow growing tree, with a lifespan that can reach 200 years. In its lifetime, a cork oak may be harvested around 17 times, although there are extremely rare cases which have exceeded this target. The cork leaves the forest in form of planks which are stacked for a minimum of six months, to stabilize moisture content. The planks are stacked onto stainless steel structure above around to avoid contamination, in such a way that they maximize water drainage and air circulation. Skilled hands sort the stored planks after the storage period. The thick cork, with the height needed to extract a whole long single stopper, is to be used for natural stoppers. The thin cork shall be transformed into disks, to be used as so called ‘technical’ corks. The cork from the first and second harvest are used to make other products for a wide range of areas such as construction, aeronautics, NASA space shuttle (coating to the shuttle), fashion and design. The planks are boiled in stainless steel tanks, the goal is to remove any organic objects embedded in the pores and enable the cork to reach the ideal moisture content. Those with defects are sent to be ground and shall be used in other products. Although, only good quality planks are selected for cork producing cork stoppers, none of the cork is wasted. Those with defects are sent to be ground and shall be used in other products. The flow chart shows the different types of cork stoppers (Technical, Natural and Granulated) with the byproducts such as floor coverings, composite cork & insulation cork board.

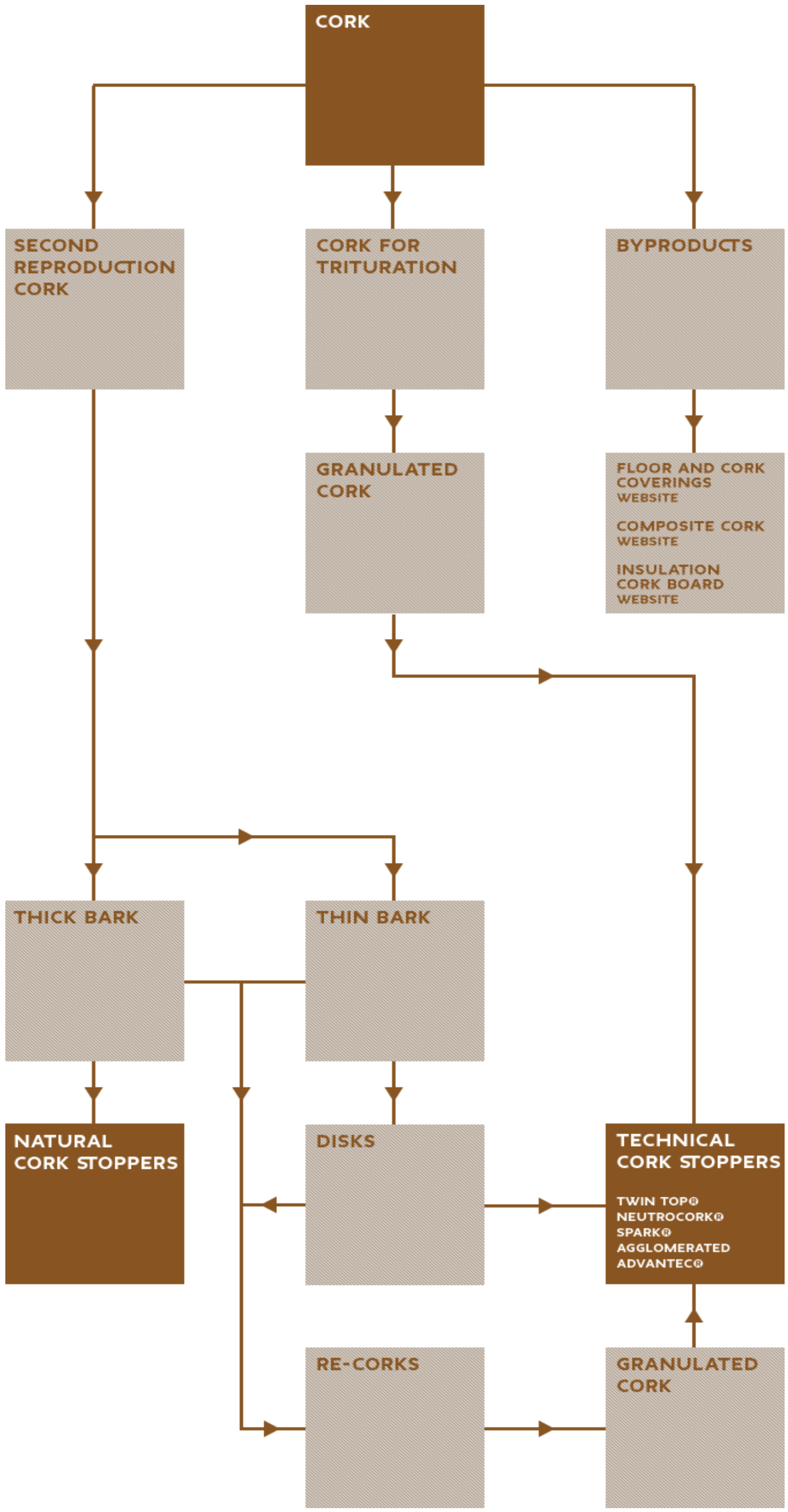

Figure 3.1 Flow chart of cork production.
Source: Perfect by nature https://www.amorimcork.com/en/natural-cork/raw-material-and-production-process/

1) Natural Cork stoppers

To produce natural-cork stoppers, planks of high quality are cut into strips and punched with a machine to extract a cylindrical stopper. Each cylinder is a whole stopper, which results from a semi-automatic cutting process. The manual punching process results in greater consistency in quality, since each worker can choose the best portion of the plank for the natural cork stoppers. [11]

Figure: 3.2 Natural Cork stoppers.
Source: http://www.corkspirit.com

After the punching process, a sample of a batch is analyzed for its lifespan and gas flow. Every Cork manufacturing company has its own quality testing mechanism to get better products. Near the end of this process, the natural stoppers are polished, resulting in a clean, smooth finish. They are then washed in an aqueous hydrogen peroxide (H_2O_2) solution for eliminating the risk of microbial contamination. FDA approved ink is used to brand the corks and are packed into polyethylene bags with preservatives (SO_2) and packages are sealed. [11]

2) Technical Cork stoppers

These are made from granules produced from the by-products of high quality cork and may or may not include natural cork disks. The same boiling process is used for manufacturing disks as that used for natural Cork stoppers.

Figure 3.3 Technical Cork stoppers.
Note the natural cork disks on the top and bottom of the cork
Source: http://www.advancecork.com

The granules to be used for the body are pulverized and screened before agglomeration. Those with a diameter between 3mm and 7mm are selected, as they provide greatest strength and elasticity to the stopper. The stoppers are assembled mechanically. Cork granules are mixed with adhesive and squeezed through a circularly shaped die to define the cylindrical shape. The cylindrical mold is then sliced into short sections that will form the central region of the technical cork stopper. The natural cork disks are then glued to the granulated cork body with the FDA approved glue. After assembly, the stopper is oven-dried for an hour before being stabilized. Technical Cork stoppers undergo same polishing, washing and finishing process to natural stoppers. These are also packed in polyethylene bags with preservatives (SO_2) and are ready to be exported. [11]

3) Granulated Cork Stoppers

Granulated cork is manufactured using the waste left over particles from the production of natural and technical cork stoppers. These corks are of low quality standard and used for lower quality products or products with short shelf-life. These corks are cheaper in price compared to the natural and technical cork. The manufacturing process is same as of the other corks. These corks are not efficient and lack the important properties, a cork should perform, such as slow diffusion of gas. [11]

Figure 3.4 Granulated cork stoppers.
Source: https://corkcho.com

3.4 Properties of Cork

Cork has some unique properties which allows it to exhibit high compressibility and low permeability to air molecules, material perfect in its application for the wine industry. These properties are derived from its distinctive cellular structure. A one-inch cube of natural cork contains more than 200 million tiny air-filled pockets. This results in excellent buoyancy, compressibility, elasticity, impervious to both air and water penetration and low thermal conductivity.

Cork properties are as follows:

- Lightness & Low Density – The cellular structure of cork makes it very lightweight, resulting in cork's celebrated buoyancy.
- Impermeability – Cork is impermeable (or more accurately, very low diffusivity) to both liquids and gases, giving it superior sealing capabilities.
- Elasticity – Cork is pliable and rebounds well to original size and shape
- Low conductivity – Cork has one of the best insulating values of any natural material, with very low conductivity of heat, sound or vibrations.
- Durability – A high friction coefficient means cork will wear and wear.
- Fire resistance – Cork has shown a remarkably high tolerance to heat.

In addition to its ability to ‘seal in’ liquids as a stopper in a bottle, the honeycomb-like structure of the cork cells (they are ~80% air) renders cork a compressible material which makes it ideal as a stopper for wine bottles: Corks are compressed and then inserted into the bottles. The bottom ~2-3 mm of the cork absorbs some of the liquid and expands thereby helping to create a nominally air-tight and liquid-tight seal.

While the seal is nominally air and liquid tight, diffusion of gasses through the cork structure do occur over time scales of months and years. The diffusion of oxygen in wine stoppers is of interest since exposure of wine to oxygen results in an oxidation chemical reaction which destroys the flavor and characteristics of the wine. Typical diffusion rates are 2e-5 m^2/s to 10e-11 m^2/s. [3]. Table 1.1 gives brief information on the physical properties of cork material. The goal of this thesis is to understand the diffusion of gases through a natural cork structure by utilizing 3D reconstructions of a cork structure (acquired through x-ray computed tomography) to construct a 3D map of the internal diffusion coefficient and then model the diffusion of gas through the cork structure.

Table 3.1 Properties of Cork

Properties	Value
Diameter	0.0254 [m]
Length	0.0508 [m]
Density	240 [kg/m^3]
Molar Mass	0.2178529 [kg/kmol]
Porosity	0.56
Specific Heat capacity	1.9E+03 [J kg^{-1} K^{-1}]
Thermal Conductivity	0.051912 [W m^{-1} K^{-1}]
Cell size	6E-8 [m]
Air pressure in cells	1.481 [atm]
Ideal Diffusion co-efficient	1.6E-9 [m^2 s^{-1}]
Maximum Diffusion co-efficient	10E-11 [m^2 s^{-1}]
Minimum Diffusion co-efficient	2E-5 [m^2 s^{-1}]

Source: Sonia Lequin, David Chassagne, Thomas Karbowaik, Jean-Marc Simon, Christian Paulin, & Jean-Pierre Bellat (2012). Diffusion of Oxygen in Cork. *Journal of Agriculture and Food Chemistry,* 2012, 60, 3348–3356

CHAPTER 4

RELATION BETWEEN PIXEL VALUE AND DIFFUSION COEFFICIENT

4.1 Introduction to MATLAB

MATLAB is a programming language software which allows matrix manipulations, plotting of functions, data implementation of algorithms and creation of user interfaces. Although MATLAB is intended primarily for numerical computing, toolbox options can be used for design and model-based simulations. MATLAB supports Graphical User Interfaces (GUIs) to plot graphs, image processing and much more. In this thesis, MATLAB is utilized to (a) process the x-ray CT images to produce a 3D map of the local diffusivity (b) curve fitting the corners of slices to find the interpolation parameters.

4.2 X-Ray Image Processing

To simulate the diffusion of gases through a cork structure, there are several steps including-

Image the internal structure of cork

↓

Map the pixel value of the x-ray CT images to local diffusion coefficient

↓

Translate and import 3D map of diffusion coefficient in ANSYS

↓

Apply boundary conditions to ANSYS model.

↓

Run the ANSYS model to predict the total oxygen gas flux from the cork.

↓

Compare the predicted flux to experimental measurements

In this chapter, steps 1 through 3 above are described. For steps 2 and 3, MATLAB is predominately used. Steps 4 and 5 are described in chapter 5 and 6.

Computed tomography x-ray images were obtained (step 1) from a research team form UAVision (Portugal) lead by Amorim & Irmãos, S.A., a leading cork manufacturing company in Portugal. These images are acquired as 'slices' through the cork sample parallel to the long central axis of the cylindrical cork. Figure 4.1 shows one such slice through a cork. In this gray scale image, regions of low mass density are black, and regions are higher mass density are white. Note for this slice the presence of a larger hole in the interior structure of the cork.

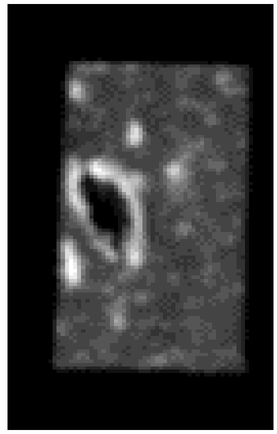

Figure 4.1 X-ray image slice of cork.

MATLAB provides the function of importing images and assigns pixel value based on the intensity. A key feature of the simulation is to relate the pixel intensities of the x-Ray CT images to a localized value for the diffusion coefficient in cork. For this thesis, we will assume a simple relationship: the higher the pixel value (i.e. the denser the cork material), the lower the local diffusion coefficient. As a starting point, this basic relationship makes sense: denser cork usually correlates to smaller cork cells. If the gas diffusion process were dominated by diffusion through the cell walls, a larger number of cell walls in a given volume should translate to a lower local diffusion coefficient.

For initial research the relation between the diffusion coefficient and pixel value was set to be inversely proportional and represented as follows:

$$\frac{D_{mean}}{D_{(x,z)}} = \frac{B_{(x,z)}}{B_{mean}} \tag{4.1}$$

Here,

D_{mean} (Experimental value of diffusion coefficient)

$B_{(x, z)}$ (Pixel value at x, z) & $D_{(x, z)}$ (Diffusion coefficient at x, z)

B_{mean} (Mean pixel value for each image)

The MATLB code for obtaining the diffusion coefficient is shown in Appendix A. Spatial values obtained from the above code are exported from MATLAB into a excel file and imported into ANSYS CFX user function. An example of the format for the Excel file are given in Figure 4.2

[Name]			
Diffusion			
[Spatial Fields]			
X	Y	Z	
[Data]			
X[m]	Y[m]	Z[m]	Diffusion[m^2 s^-1]
0.012064	0.012454	0.01379	5.00E-13
0.012064	0.012409	0.01379	5.00E-13
0.012064	0.012363	0.01379	5.00E-13
0.012064	0.012317	0.01379	5.00E-13
0.012064	0.012272	0.01379	5.00E-13
0.012064	0.012226	0.01379	5.00E-13
0.012064	0.01218	0.01379	5.00E-13
0.012064	0.012135	0.01379	5.00E-13
0.012064	0.012089	0.01379	5.00E-13

Figure 4.2 Example of format for Excel file.

Enabling importing of experimental data into ANSYS simulation software. At each experimentally measured x-ray CT position, the pixel value of the CT image is mathematically mapped into the spatial data of diffusion coefficient.

4.3 Use of Curve Fitting Toolbox

For step 4, the imported 3D Diffusion coefficient map must be spatially translated to coincide with the volume of the cork structure defined in ANSYS. To produce the appropriate mapping, the boundaries of the cork structure must be determined from the x-ray CT images. The MATLAB image processing toolbox imports the image, divides it in small pixels with an intensity value assigned to each pixel. The pixel value outside the four edges of the cork is zero. A MATLAB code was developed to identify the row and column of the cork corners for which the pixel values are zero (Figure 4.3). The code then determines the four corner points of the edges. A description of the code is briefly explained in Appendix B.

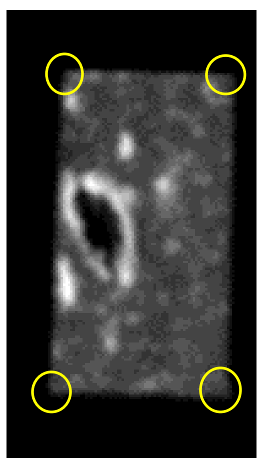

Figure 4.3 Illustration of methodology for determining edges of cork structure.

Once the table of data is created identifying the corners structure in the CT images, the curve fitting toolbox with MATLAB is used to import the data of the corner points.

$$(x - x_0)^2 + (y - y_0)^2 = {r_0}^2 \tag{4.1}$$

Using the curve fitting tool, the edge points are fit the generalized equation for a circle. The best fit parameters of x_0, y_0 and r_0 are then used to translate the diffusion 3D map into ANSYS such that the geometric center of the cork is positioned at the origin in ANSYS with the long axis of the cork cylinder laying along the Y axis. This allows to curve fitting the data to obtain the translation parameters and the center of the cork structure. The best fit circle through the data is shown in Figure 4.4.

Based on the curve fitting results from MATLAB, the best fit parameters are x_0= -0.1016 mm $\pm$ (0.0066), yo=1.888 mm $\pm$ (0.041) and r_o= 0.0128 mm $\pm$ (0.0026). The uncertainties in the best fit parameters are for a 95% confidence level. The adjusted R 2 value for the fit is 0.9918.

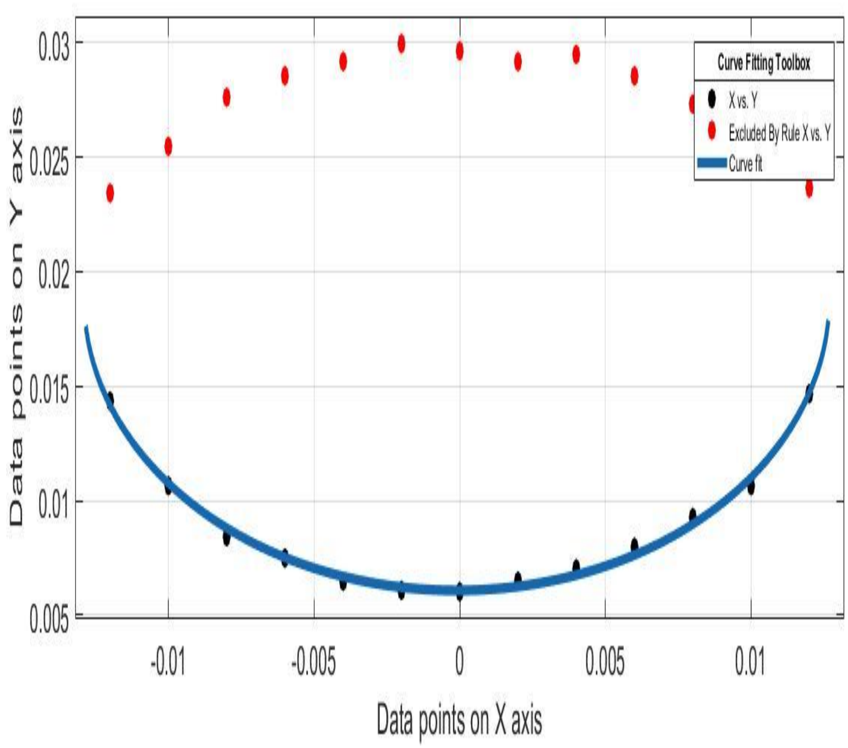

Figure 4.4 Curve fit of data points which determines geometric parameter for translating experimental data to coordinate system of simulation.

CHAPTER 5

SETUP AND DESIGN

In this chapter, the setup and design of the cork structure in ANSYS is described. Included in this discussion is the setup of the boundary conditions.

5.1 Isotropic Diffusion Co-efficient

The analysis of gas flow through cork was performed in a step by step process. A 3D model was developed in ANSYS cfx to demonstrate the basic out flow of oxygen through cork. The boundary conditions were as follows:

- Diameter (determined by the best fit to CT image data described in Chapter 4, but nominally): 24 mm
- Length: 42 mm
- Pressure at Outlet1 & Outlet2: 1 atm (The circular top and bottom of the cork defines the boundaries for calculating the flux of gas. Flux of gas perpendicular to the cylindrically shaped boundary of the cork, corresponding to the interface between the cork and the glass of a wine bottle, is defined to be zero.)
- Pressure inside the cork: 1.481 atm (This value is based on the typical pressure increase inside of the cork when it is compresses prior to insertion of the cork into the neck of a wine bottle.)
- Density of cork: 240 kg/m^3 [3]
- Porosity: 0.056 [3]
- Diffusion co-efficient: 1.6e-10 m^2/s (mean value) [3]
- Equation to be solved:

$$\frac{dC}{dt} = \nabla(D\nabla C) \text{ or } \frac{dP}{dt} = \nabla(D\nabla P) \tag{5.1}$$

A) Geometry of Cork

The first step in designing is to develop a 3D geometry model of the cork in DesignModeler feature with length of 42 mm and diameter of 24 mm.

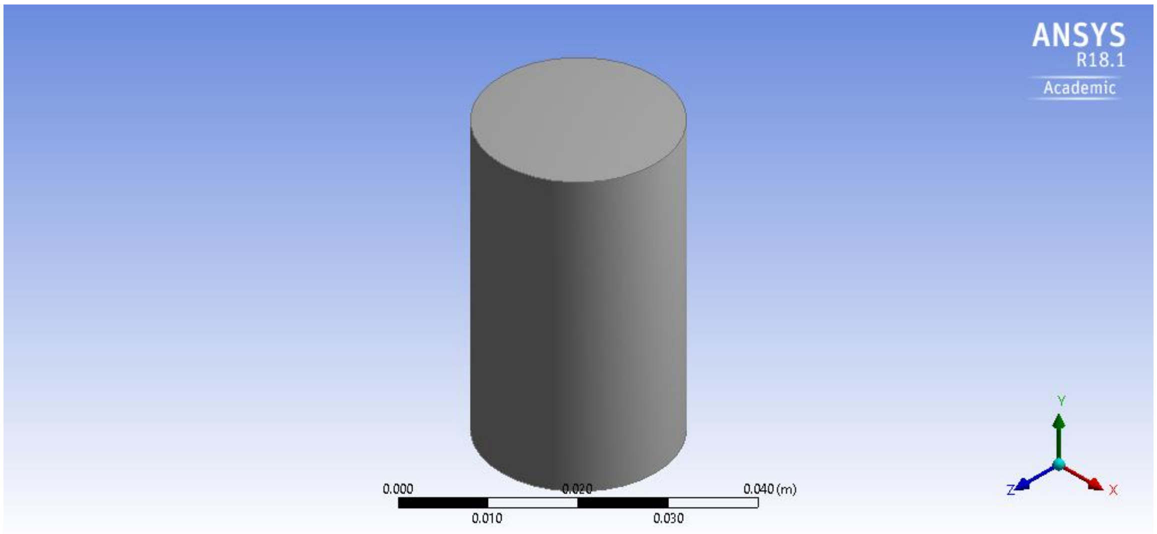

Figure 5.1 Geometry of cork in DesignModeler.

B) Mesh generation.

Mesh is generated using the in-built mesh feature with fine quality for better results.

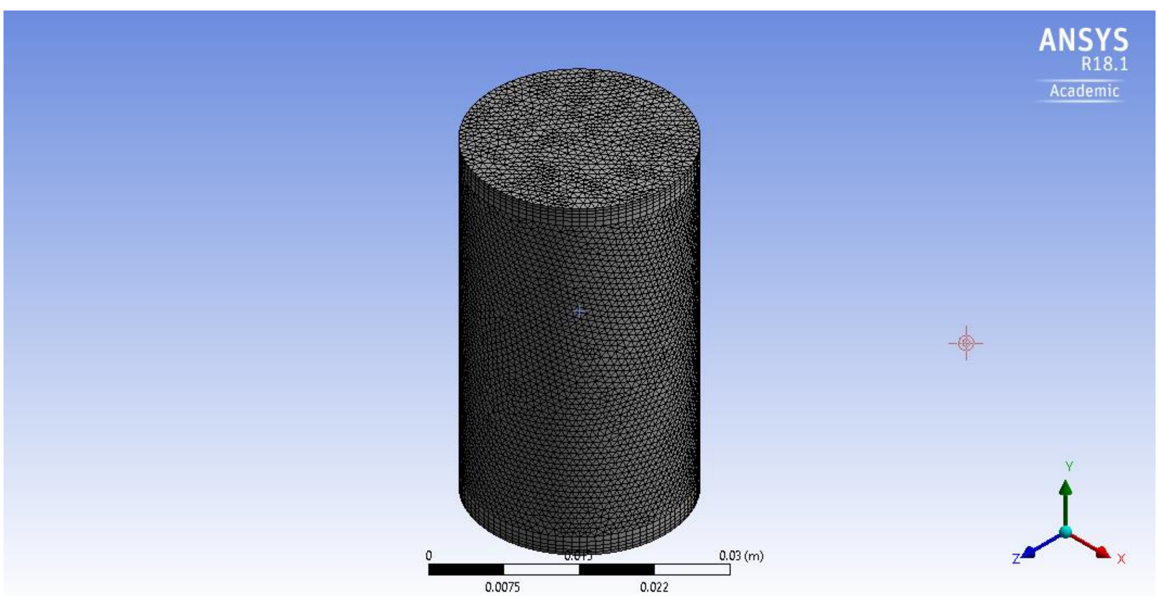

Figure 5.2 Mesh of geometry.

C) Cfx-Pre (Setup boundary conditions)

At this step, all the necessary boundary conditions are specified to the model.

- Pressure at outlets: 1 atm
- Pressure inside the cork: 1.481 atm
- Total run time: 8760 hr. (one year)
- Timestep size: 730 hr. (one month)

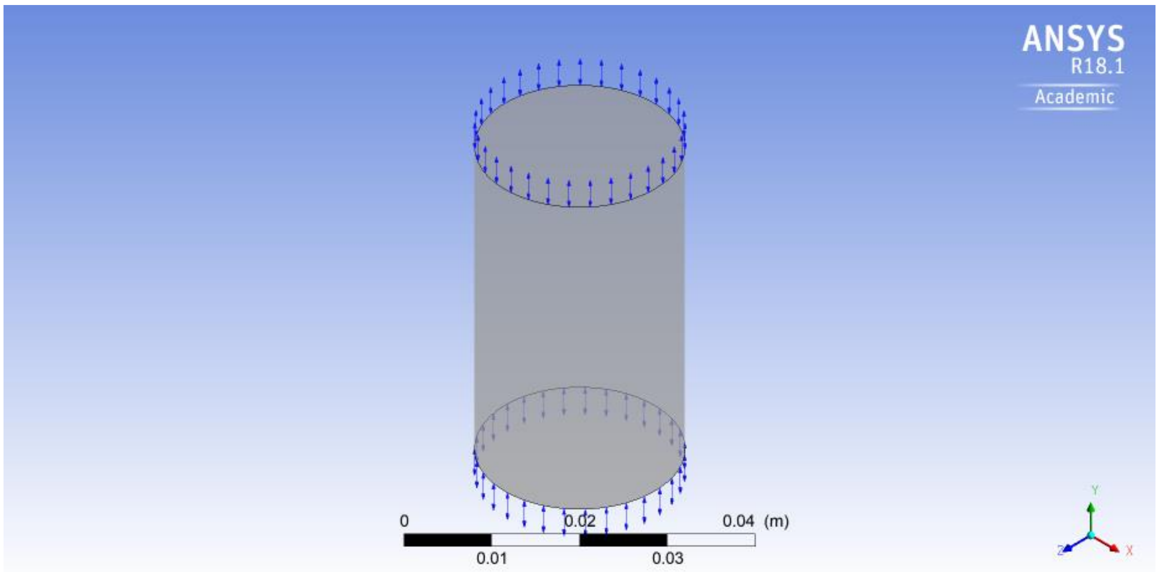

Figure 5.3 CFX-Pre-Setup of model with boundary conditions.

Flow of gas into bottle is through bottom of model & into the atmosphere is through top.

D) CFX- Solver Manger

Convergence history plots and user point plots are shown on monitor tabs. Each monitor tab shows at least one plot of a variable versus time step, where the variable can be an RMS or maximum residual, or a user-defined variable. For example, the Momentum and Mass monitor tab shows plots of the RMS/maximum residuals for pressure and the U, V, and W components of momentum. A legend appears below each plot to show the variable associated with each plot line.

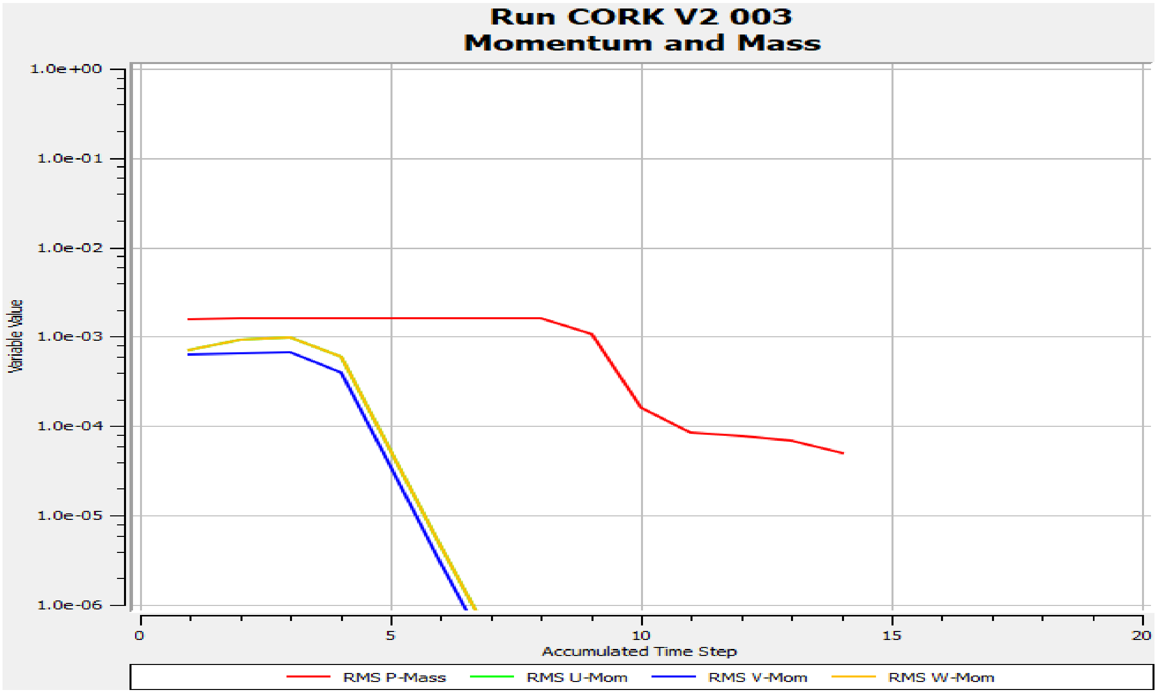

Figure 5.4 CFX Solver manager residuals graph.

E) CFD-Post

This section helps to visualize the results in form of graphs, contours & animations to better understand the flow. Below is a graph of Mass flow vs Time for the flow of oxygen in the bottle. Due to the initial high pressure there is a sudden ingress of oxygen into the bottle, same as the experimental data found. As a simple test of the model, if we assume that the diffusion coefficient condition were isotropic in the cork structure, the mass flow at the top is same as of the bottom (see Figure 4.6 & 4.7) which suggests that the model is working correctly.

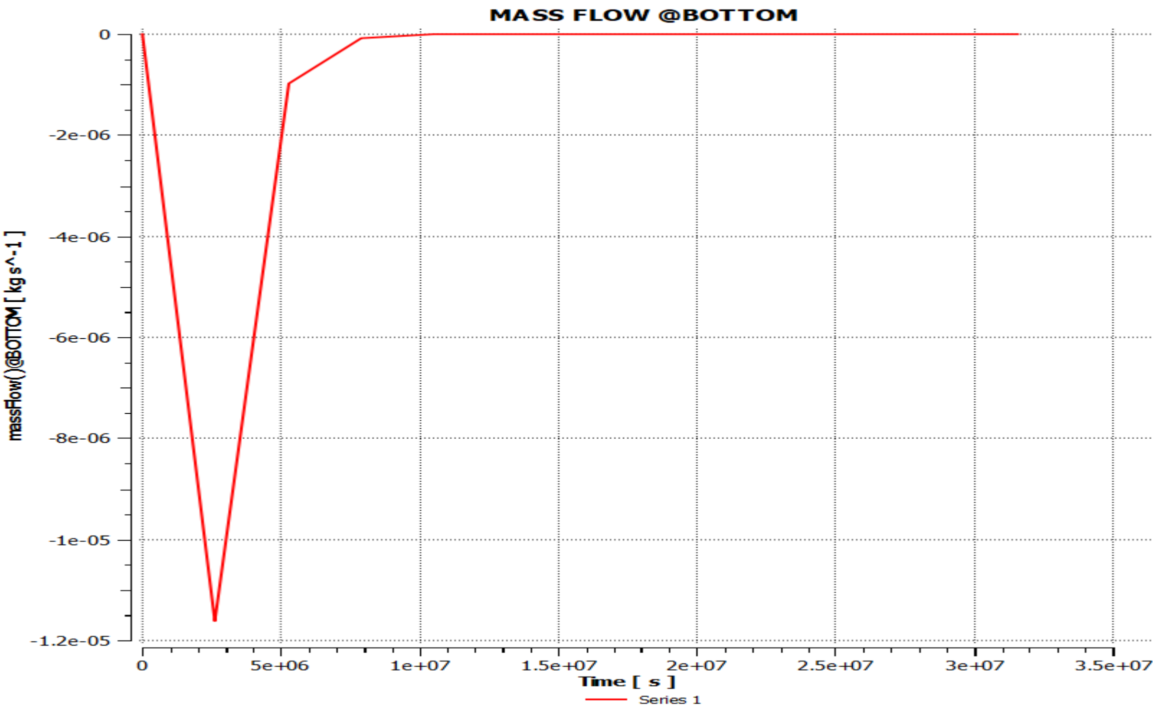

Figure 5.5 Mass flow at the bottom of cork.

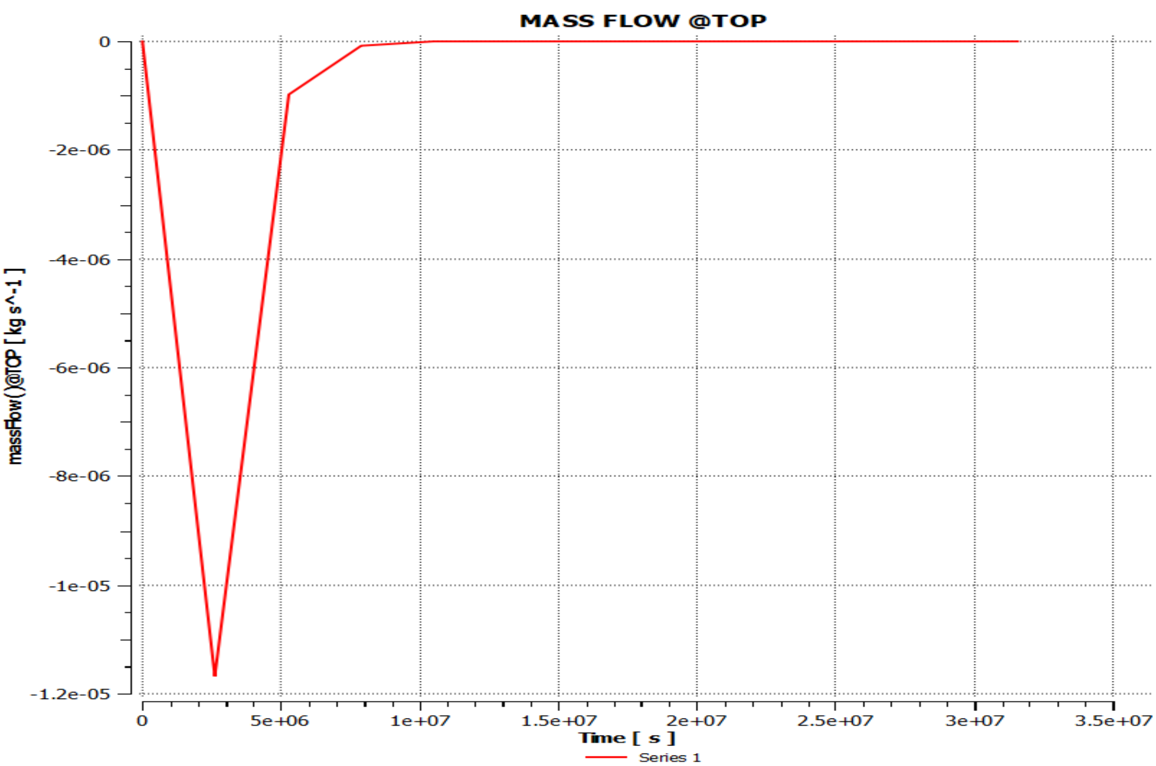

Figure 5.6 Mass flow at the top of cork.

To compare the simulation to experimental results, one needs to convert the simulation output (namely mass flow (kg/s) out of the cork as a function of time) to the total cumulative mass flux (kg) as a function of time. Mathematically one calculates the total cumulative mass flux by integrating the mass flow from initial time to the time of interest.

5.2 Failure of ANSYS CFX

According to the documentation [12], ANSYS cfx solves the Equation (5.2) as an additional variable.

$$\frac{\partial(\rho\phi)}{\partial t} = \nabla.(\rho D_\Phi \nabla\phi) + S_\phi \qquad \textbf{(5.2)}$$

Where,

- ρ is the mixture density, mass per unit volume.
- Φ is the conserved quantity per unit volume, or concentration.
- D_Φ is the kinematic diffusivity for the scalar
- S_ϕ is a volumetric source term, for this problem, the value is zero.

For most applications, the transport of additional variables is both a convective and diffusive process (including both laminar and turbulent diffusion), and one would therefore need to specify the molecular kinematic diffusivity for each additional variable used. This describes how rapidly the scalar quantity would move through the fluid. Ansys solves diffusion of oxygen in natural cork as a convection-dominated flow due to the pressure gradient. The kinematic diffusivity specified to the model can have little effect because convection processes dominate over diffusion processes. The software was never developed to solve these types of problems and consider the boundary conditions which are necessary in the simulation where diffusion causes mass flow in a system.

To illustrate the inability of our ANSYS model to solve oxygen diffusion in a diffusion dominated regime, the AYSYS model was run with two different initial conditions: For both simulations, the initial diffusion coefficient was assumed to be constant throughout the cork. However, one simulation used a smaller diffusion coefficient than another. Based on our simple model for the diffusion process, changing the diffusion coefficient should change the time-scale for the total mass flow from the cork to saturate. However, the final total cumulative mass should be independent of the diffusion coefficient because in both cases, the total amount of initial gas in the cork after compression is the same.

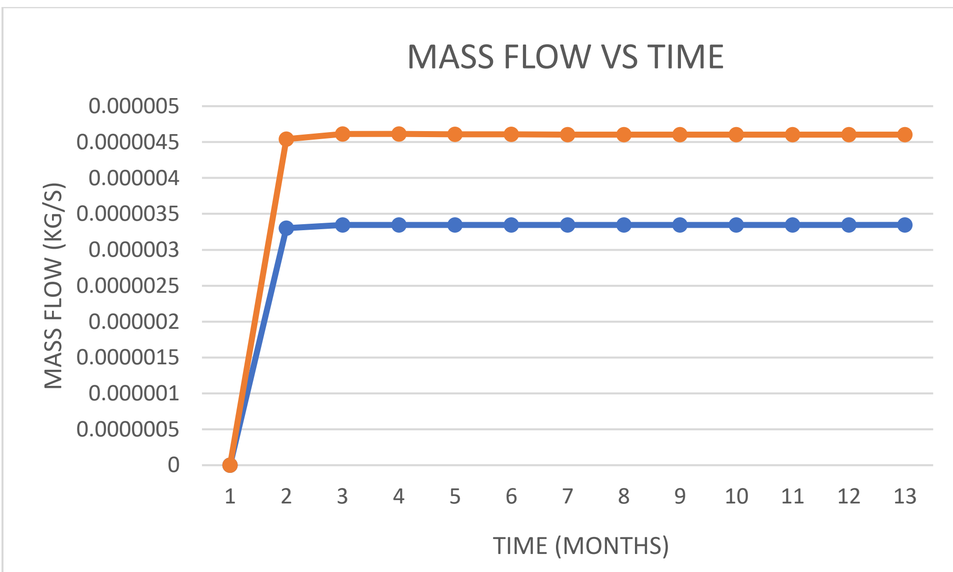

Figure 5.8 Cumulative Mass flow for different diffusion coefficient.

As shown in the ANSYS simulation of Figure 5.8, the total amount of oxygen diffusing out of the cork changed with the value of diffusion coefficient, which is an unphysical answer for our problem. Also, discussions with ANSYS technical support suggested that ANSYS could not appropriately solve equations for the diffusion dominated flow of gas. Therefore, in order to solve for gas diffusion in a strongly diffusion dominated regime, a different finite-element-analysis software system was used. COMSOL (www.comsol.com) solver and multiphysics simulation software can solve for gas diffusion in the diffusion dominated regime. In our implementation of the problem in COMSOL, the physics of model is considered as 'Transport of dilute species' in which the 'gauge' concentration is considered for analysis. The detailed steps on modeling are explained briefly in Chapter 6.

CHAPTER 6

INTRODUCTION TO COMSOL

In this chapter the design, results and post processing steps for modelling of oxygen diffusion in cork are performed in COMSOL software. Included in this section is the setup and the boundary conditions.

6.1 Isotropic Diffusion Coefficient

A 3D geometric model was developed in COMSOL to demonstrate the basic flow of oxygen through cork. For this simulation, we utilized the differential equations for the diffusion of a dilute species. Equation 5.1 was used to model the diffusion of oxygen. For this formulation, the concentration of oxygen is measured relative to the normal concentration in the atmosphere. Essentially, this is analogous to measuring the gauge pressure (i.e. relative to atmospheric pressure).

The boundary conditions were as follows:

- Diameter (determined by the best fit to CT image data described in Chapter 4, but nominally): 24 mm
- Length: 42 mm
- Gauge concentration at top & bottom: 0 mol/m^3
- The circular top and bottom of the cork defines the boundaries for calculating the flux of gas. Flux of gas perpendicular to the cylindrically shaped boundary of the cork, corresponding to the interface between the cork and the glass of a wine bottle, is defined to be zero.
- Gauge concentration inside the cork: 0.481 mol/m^3 (Using the ideal gas law equation-PV=nRT, the typical pressure increases inside of the cork when it is compresses prior to insertion of the cork into the neck of a wine bottle. Corresponds to 1.481 atmospheres)
- Density of cork: 240 kg/m^3 [3]
- Diffusion co-efficient: 1.6e-10 m^2/s (mean value) [3]

6.1.1 Geometry and Mesh of Model

For the first step, the geometry of the cork is defined in COMSOL with length of 42 mm and 24 mm. The geometry is shown in Figure 6.1 and mesh is shown in Figure 6.2.

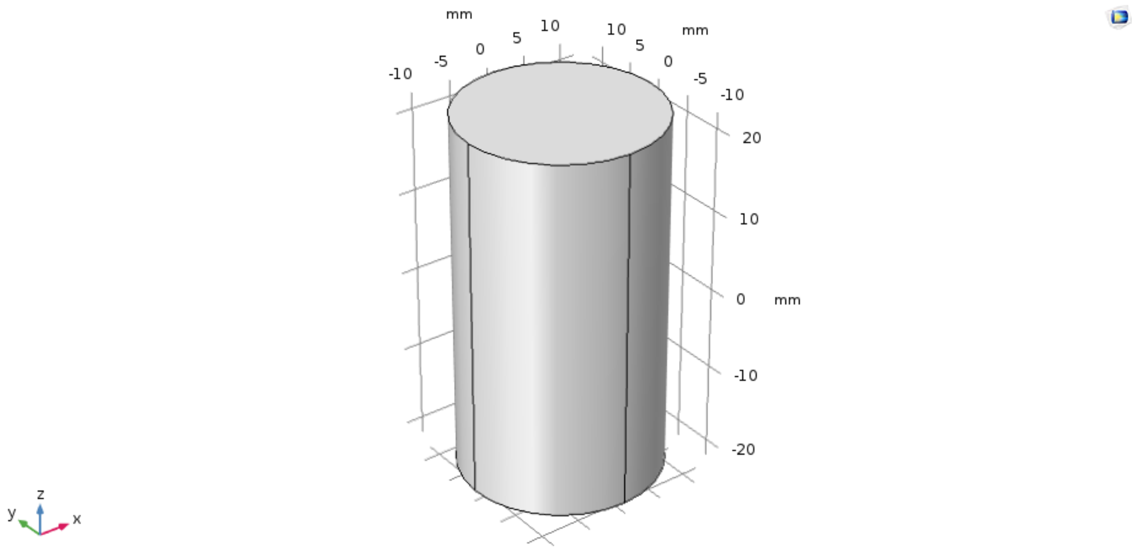

Figure 6.1 Geometry of cork in COMSOL.

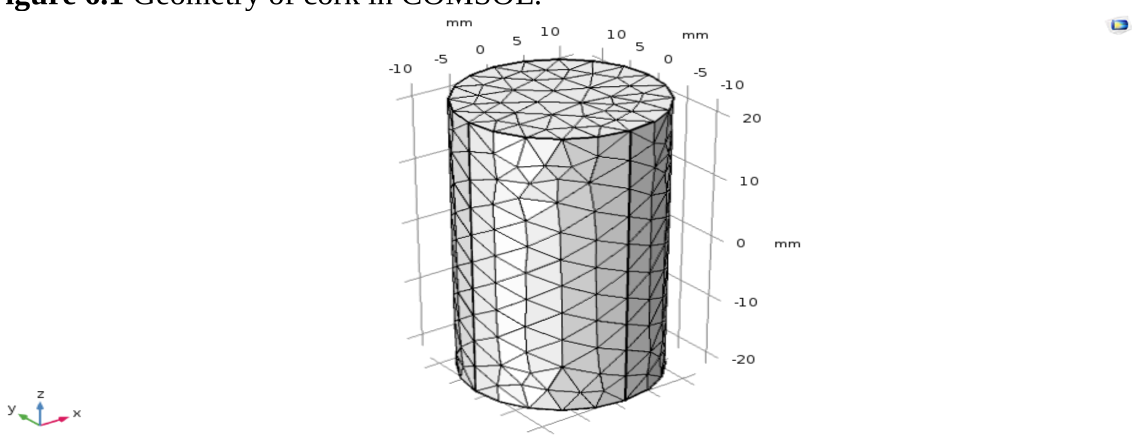

Figure 6.2 Fine mesh of Cork.

6.1.2 Oxygen Concentration in Cork

As an initial test of our model, the diffusion coefficient at each x, y, z value of the imported 3D diffusion coefficient map is set a constant value i.e. 2e-10 m^2/s. Under these circumstances, it is expected that the – oxygen flux will be same at both the outlets of the cork body. Figure 6.3, 6.4, 6.5 & 6.6 shows the concentration of oxygen at time 0, 1, 2 & 12 Ms timesteps. Note that at each time step, the concentration is approximately the same across the circular cross-section of the cork at a fixed height z in the cork as one would expect for a constant diffusion coefficient. Also note that at very long times, the gauge concentration in the cork approaches zero as expected.

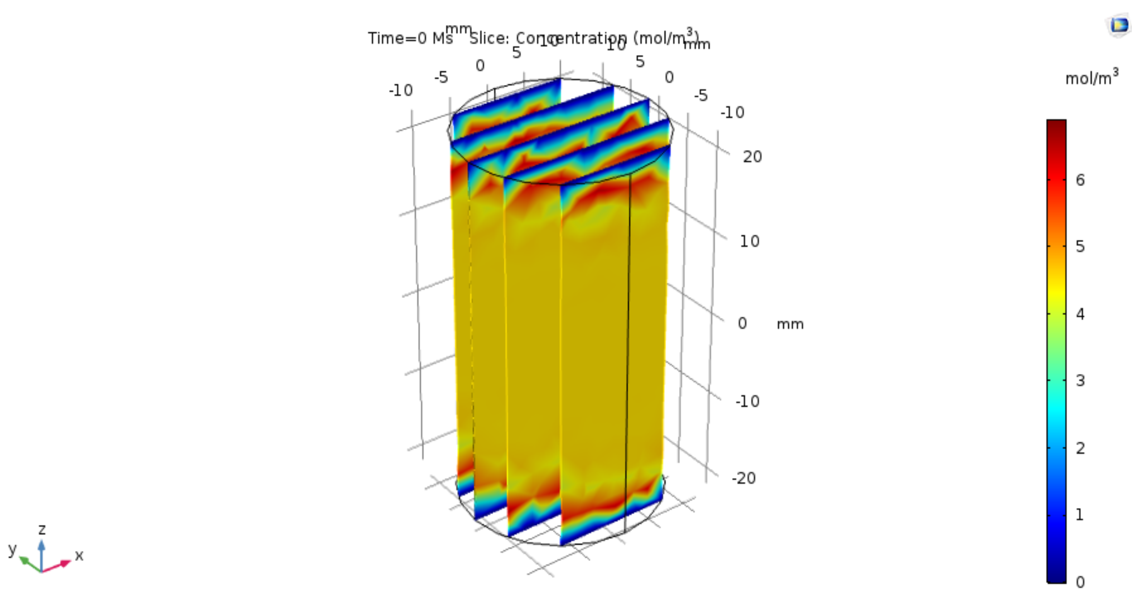

Figure 6.3 Initial gauge concentration in Cork.

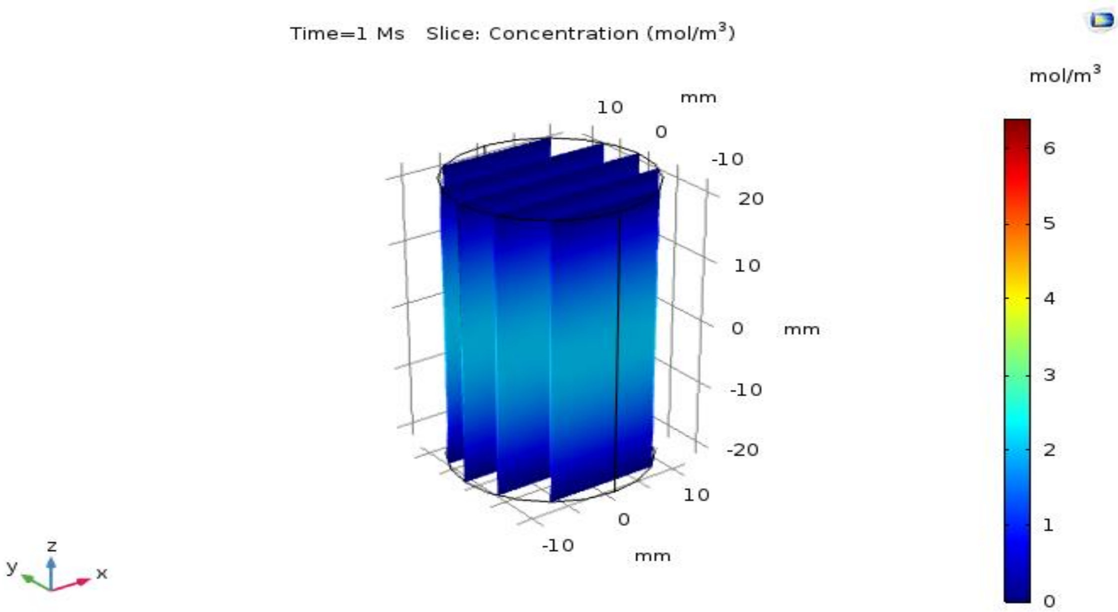

Figure 6.4 Gauge concentration in cork at time (1 Ms).

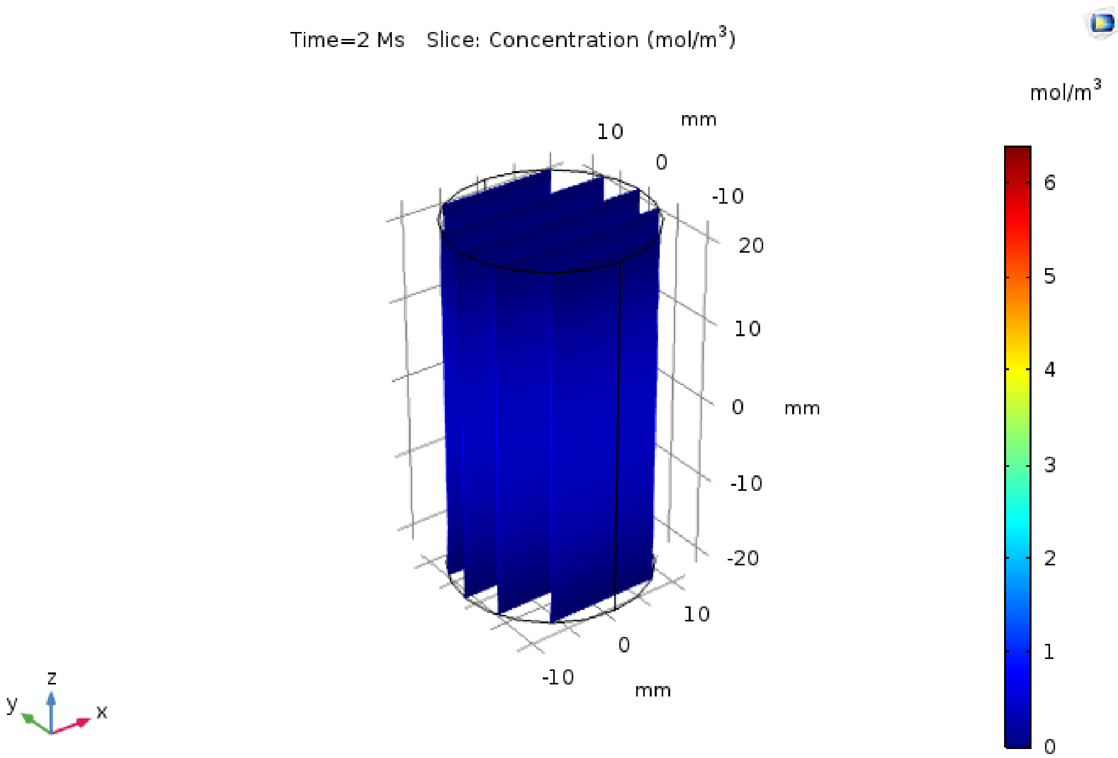

Figure 6.5 Gauge concentration in cork at time (2 Ms).

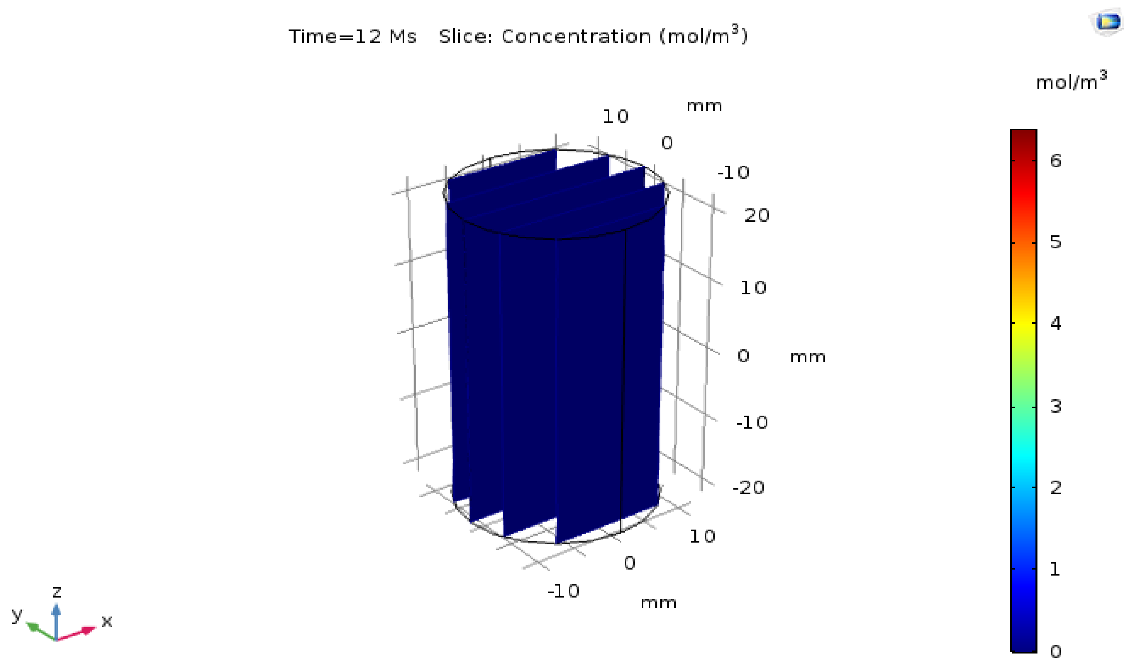

Figure 6.6 Gauge concentration in cork at time (2 Ms).

COMSOL calculates the mass flux at each timestep. This data was imported to Excel and the cumulative total flux was calculated. As the diffusion coefficient is 2e-10 m^2/s (constant), the total cumulative flux at top and bottom outlets is same and is shown in the Figure 6.7 & 6.8. These results verify that the COMSOL model reproduces expected results for simple initial conditions.

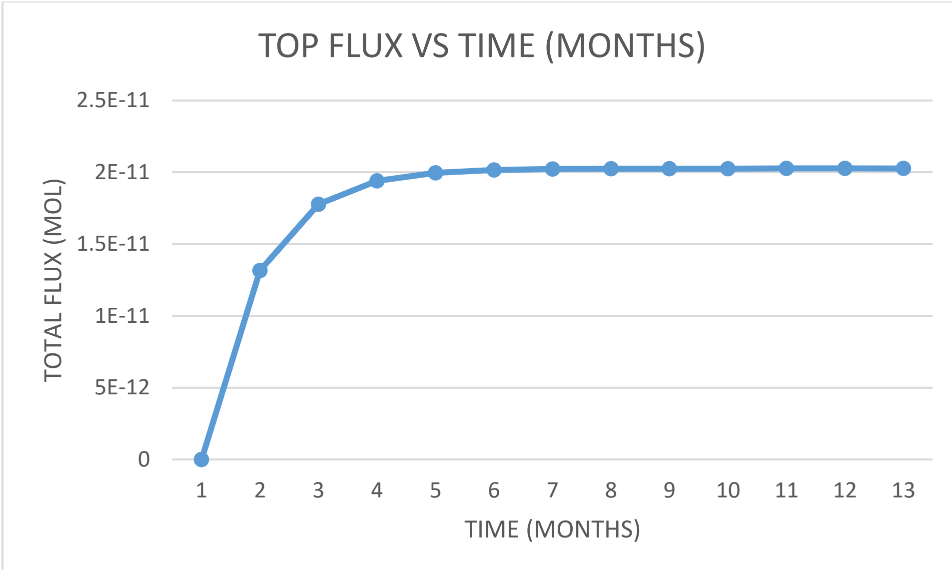

Figure 6.7 Total flux at top outlet.

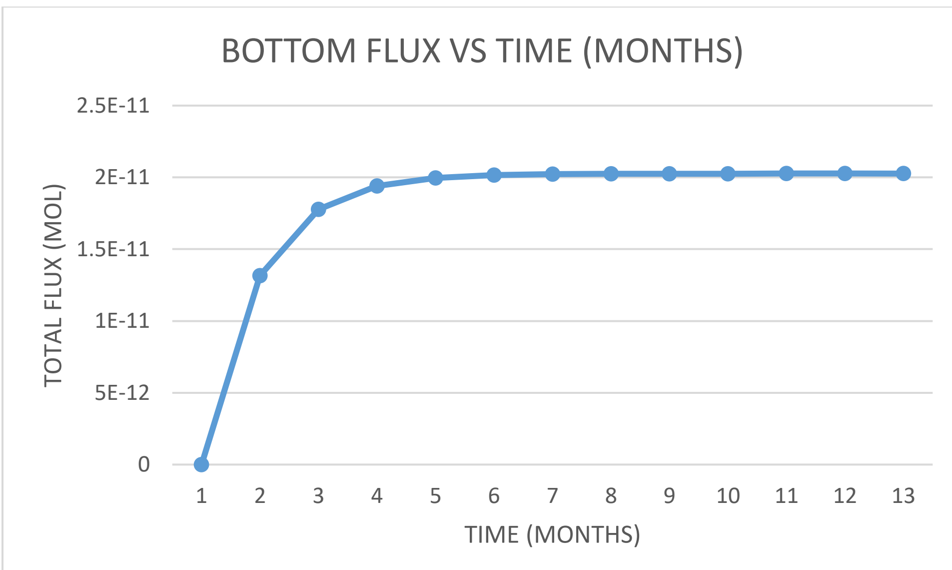

Figure 6.8 Total flux at bottom outlet.

6.2 Diffusion Coefficient Larger at Top

For the next step of the testing process, the diffusion coefficient at the upper section was set to be higher than the lower section.

$$D_0 = D_2(z + \frac{L}{2})(\frac{L}{2}) \quad (6.1)$$

Where,

D_0= Diffusion coefficient variable depending on z value

D_2= Ideal Diffusion coefficient (2e-10 m^2 s^{-1})

From Figures 6.9, 6.10 & 6.11, it can be interpreted that the oxygen diffuses very quickly through the top outlet.

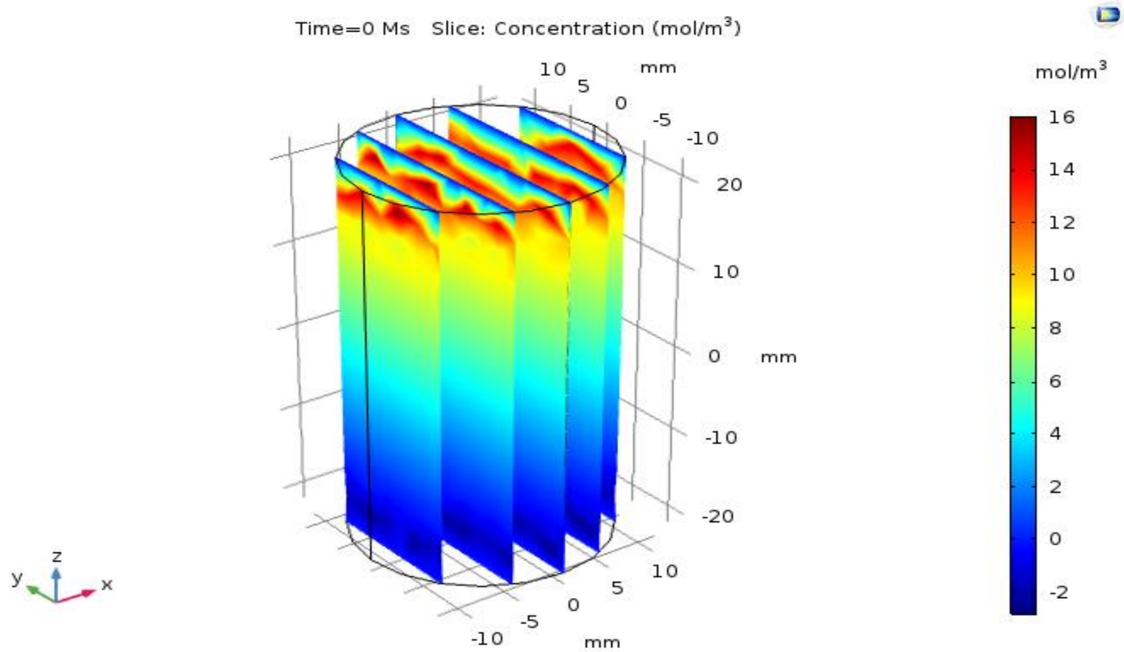

Figure 6.9 Plot of Oxygen gauge concentration inside cork at time= 0 Ms.

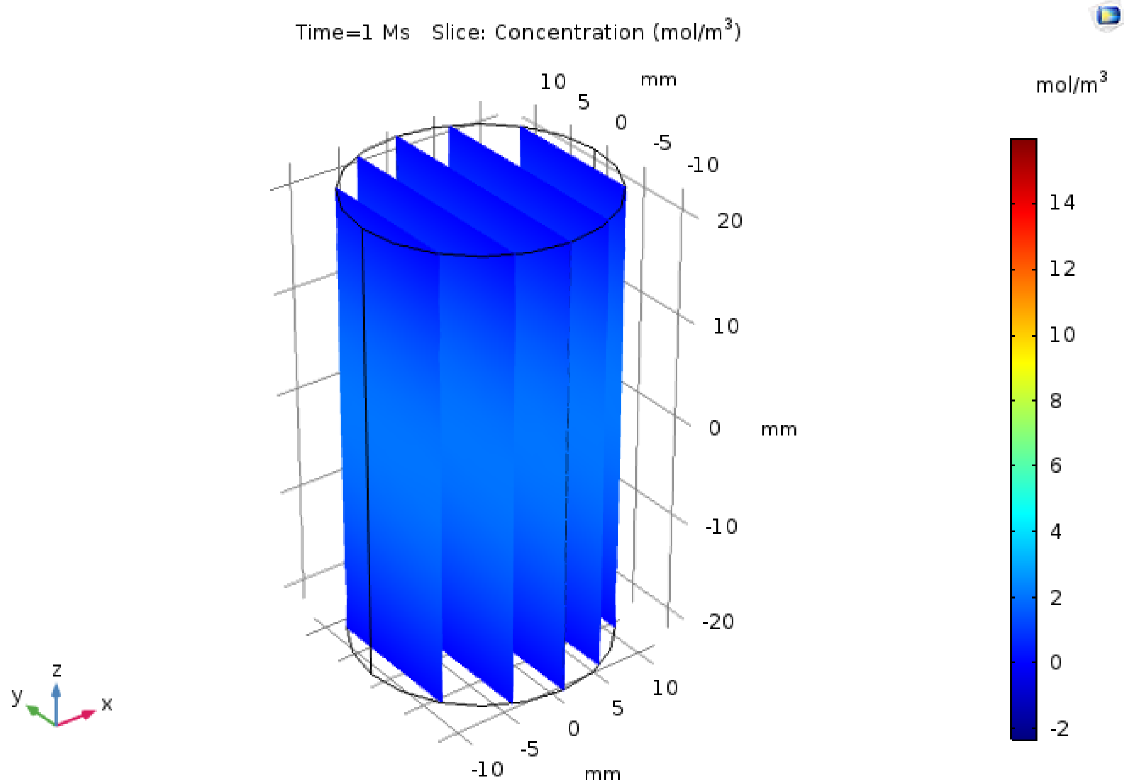

Figure 6.10 Oxygen gauge concentration inside the cork at time =1 Ms.

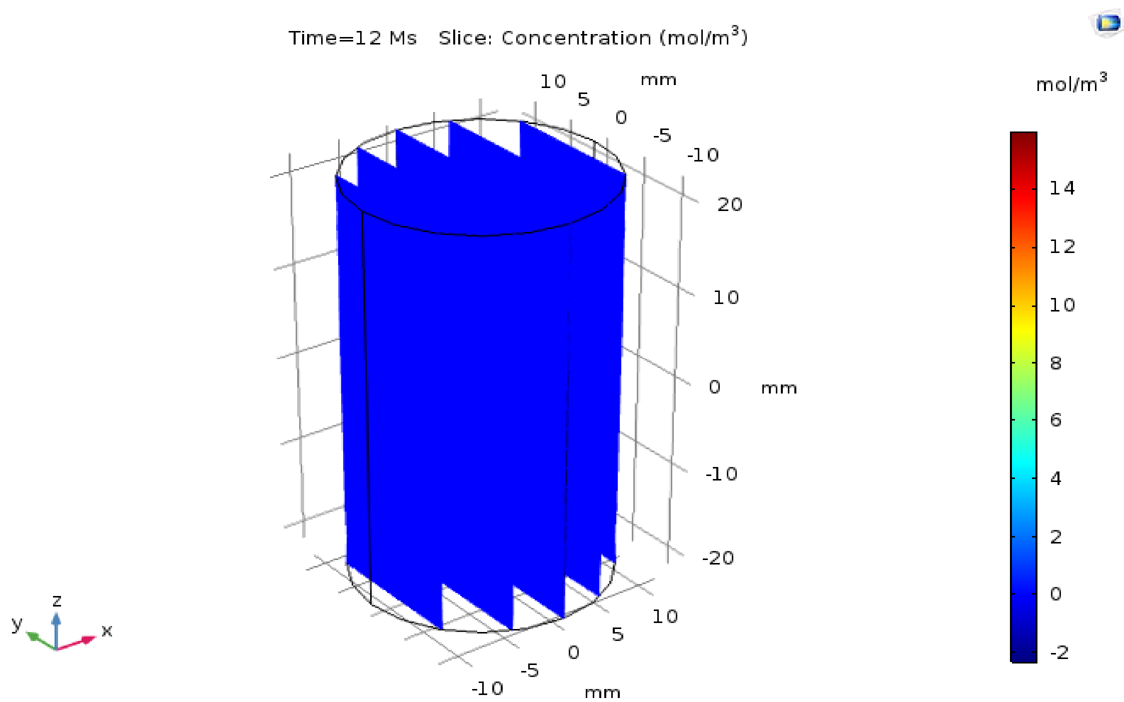

Figure 6.11 Oxygen gauge concentration inside the cork at time =12 Ms.

The total cumulative flux at the top outlet is slightly higher than the bottom as would be expected since the initial concentration is higher near the top of the cork. Figure 6.12 & 6.13 show the plot of total flux vs time at top and bottom outlet respectively.

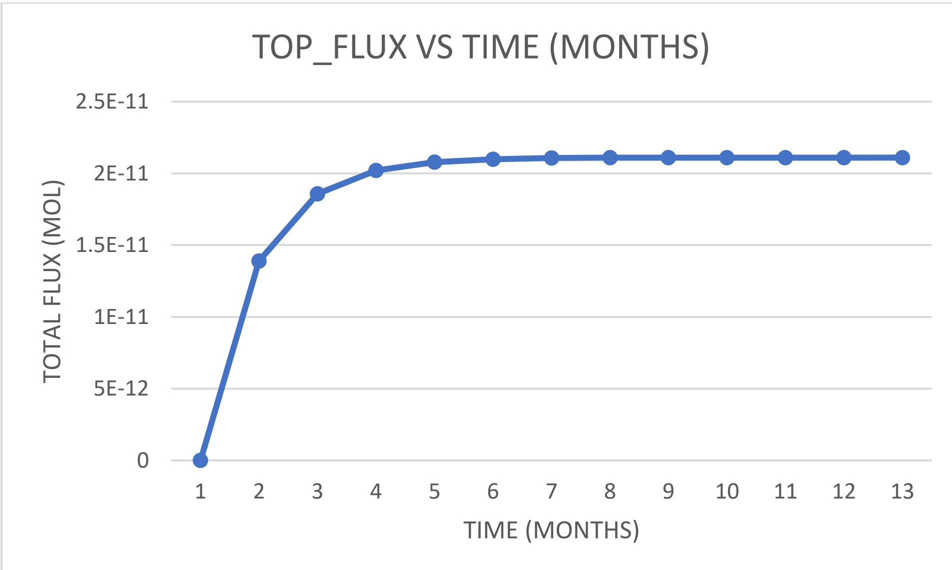

Figure 6.12 Cumulative total flux at the top outlet.

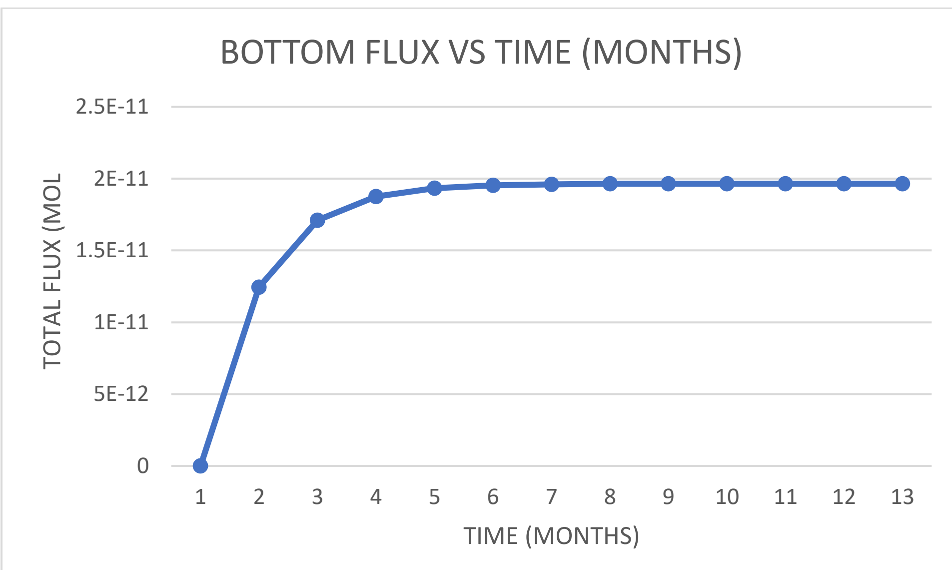

Figure 6.13 Cumulative total flux at the bottom outlet.

6.3 Spatial Variable Diffusion Coefficient

At this step of model testing, the diffusion mapping is imported from MATLAB to COMSOL in a txt file as shown in Figure 6.16. While values are given for the diffusion coefficient only within the spatial domain of the cork, when COMSOL imports the txt file, it determines the values outside the domain by interpolation. COMSOL uses linear interpolation to calculate the unknown value of arguments where data is missing inside the domain. To avoid this issue, the value of diffusion coefficient outside the domain is set at a higher value and a wall with no flux (i.e. the glass-cork interface) is added.

```
-0.003094737    0.006   0.015915789     2.36E-10
-0.003094737    0.006   0.016357895     2.22E-10
-0.003094737    0.006   0.0168  1.95E-10
-0.003094737    0.006   0.017242105     1.67E-10
-0.003094737    0.006   0.017684211     1.51E-10
-0.003094737    0.006   0.018126316     1.50E-10
-0.003094737    0.006   0.018568421     1.59E-10
```

Figure 6.14 Sample of input file for COMSOL which specifies the x, y, z and corresponding diffusion coefficient in the last column.

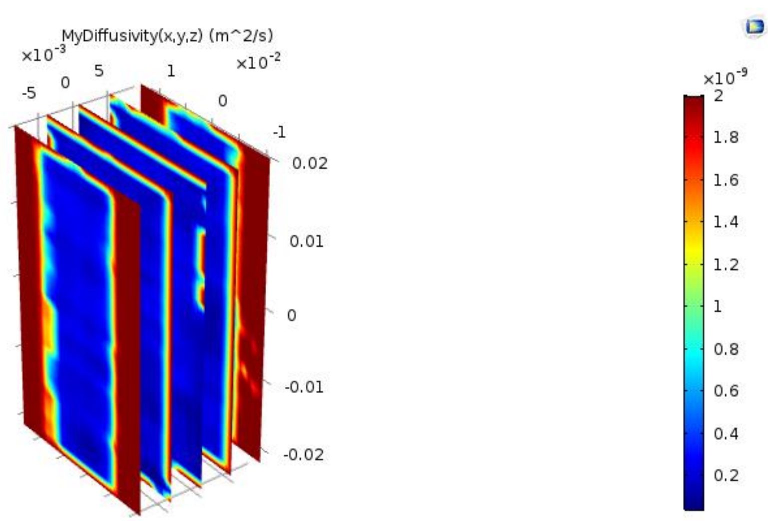

Figure 6.15 Plot of spatial diffusion coefficient mapping.

6.3.1 Concentration Inside the Cork

The concentration of oxygen inside the cork is shown in Figures 6.17- 6.21.at time 0,1,2,12 Ms of timesteps. While the initial concentration (Figure 6.18) is essentially constant in the cork's interior, as the gas diffuses, there is a higher concentration of gas inside of the hole as indicated in Figures 6.19-6.20. The higher concentration of oxygen at the same position of the hole in cork is as expected: As shown in Figure 4.1, the hole is surrounded by dense cork material. The high density translates into a lower diffusion coefficient essentially impeding the flow of gas from the hole. For very long times (Figure 6.21), the concentration of gas in the cork becomes isotropic as expected.

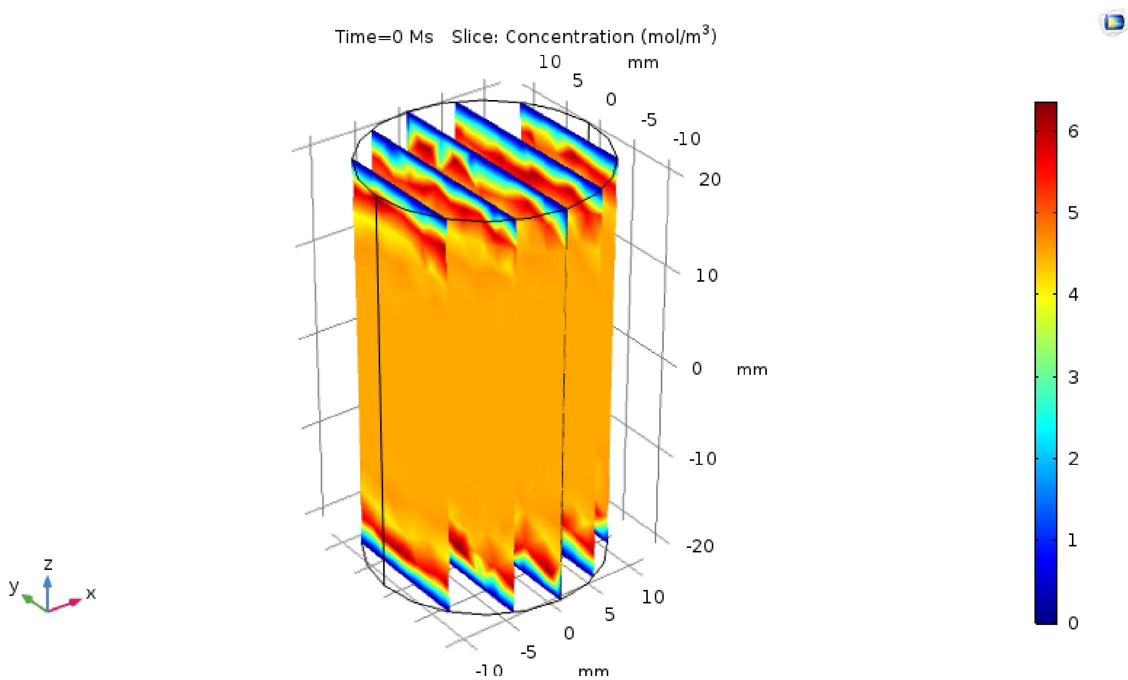

Figure 6.16 Oxygen gauge concentration at time= 0 Ms.

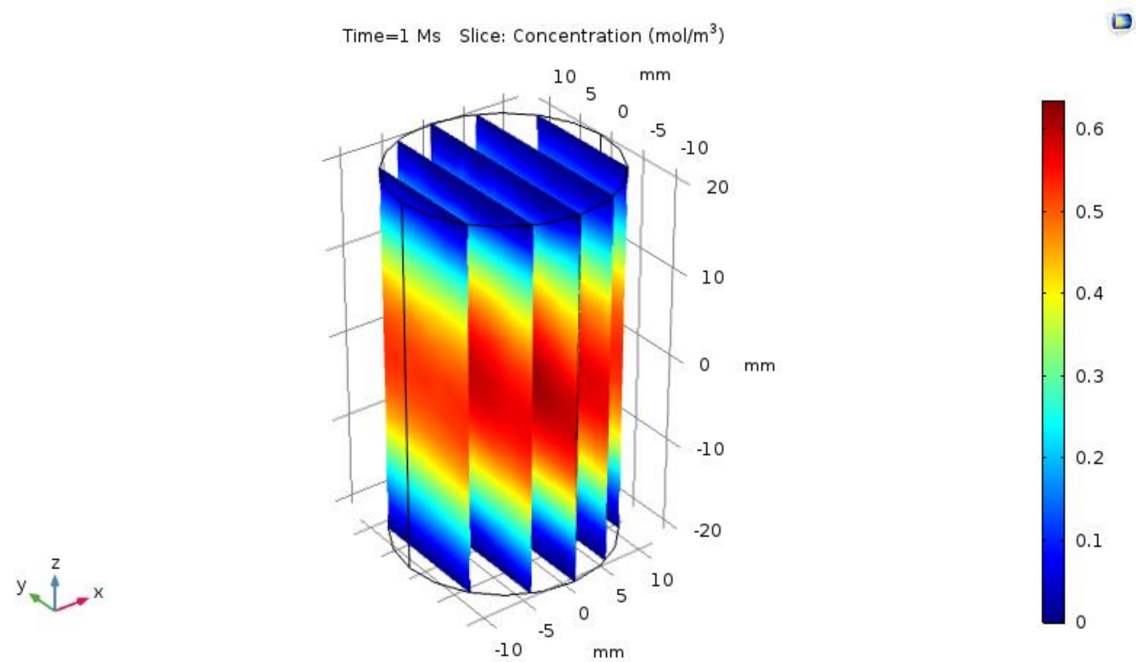

Figure 6.17 Oxygen gauge concentration at time= 1 Ms.

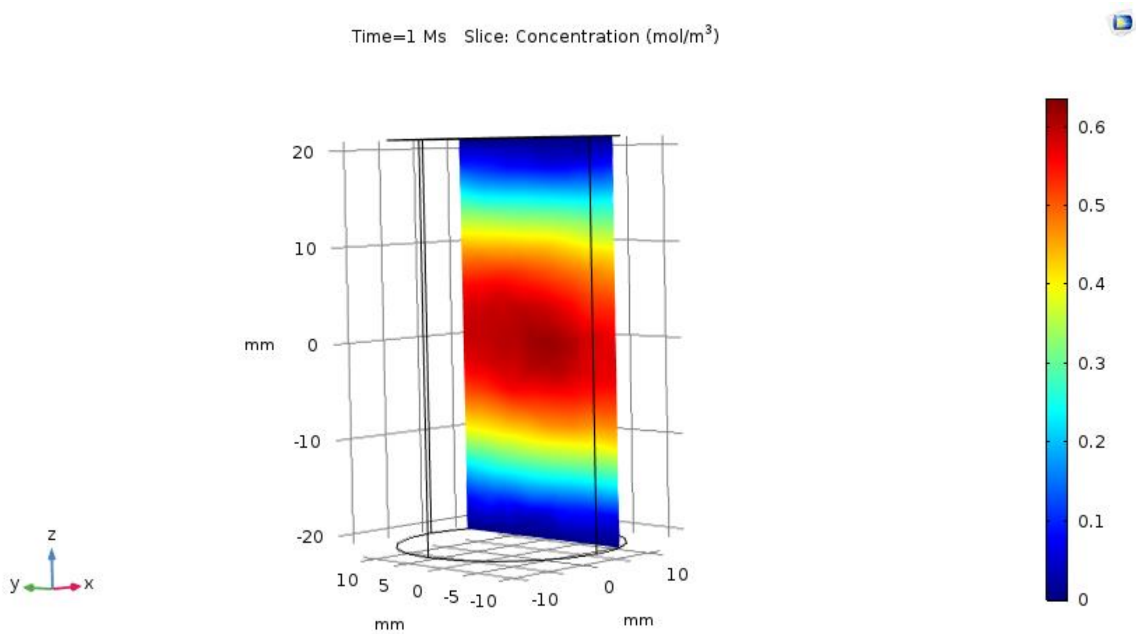

Figure 6.18 Gauge concentration at the hole from the x-ray images.

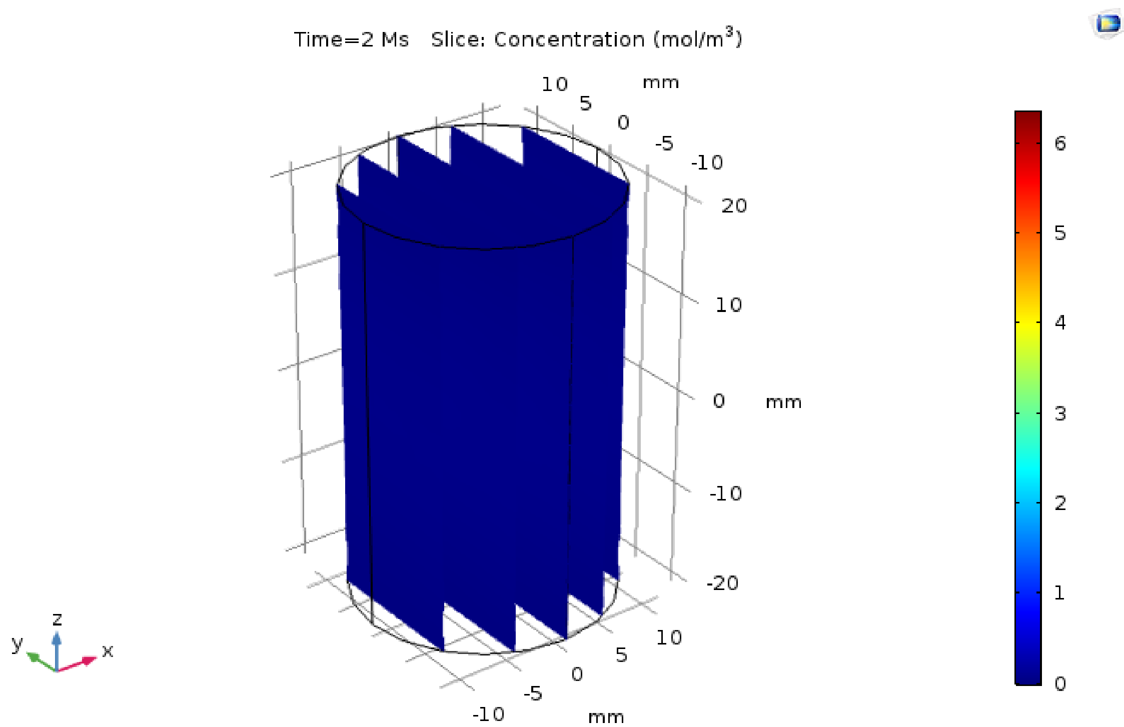

Figure 6.19 Oxygen gauge concentration at time= 2 Ms.

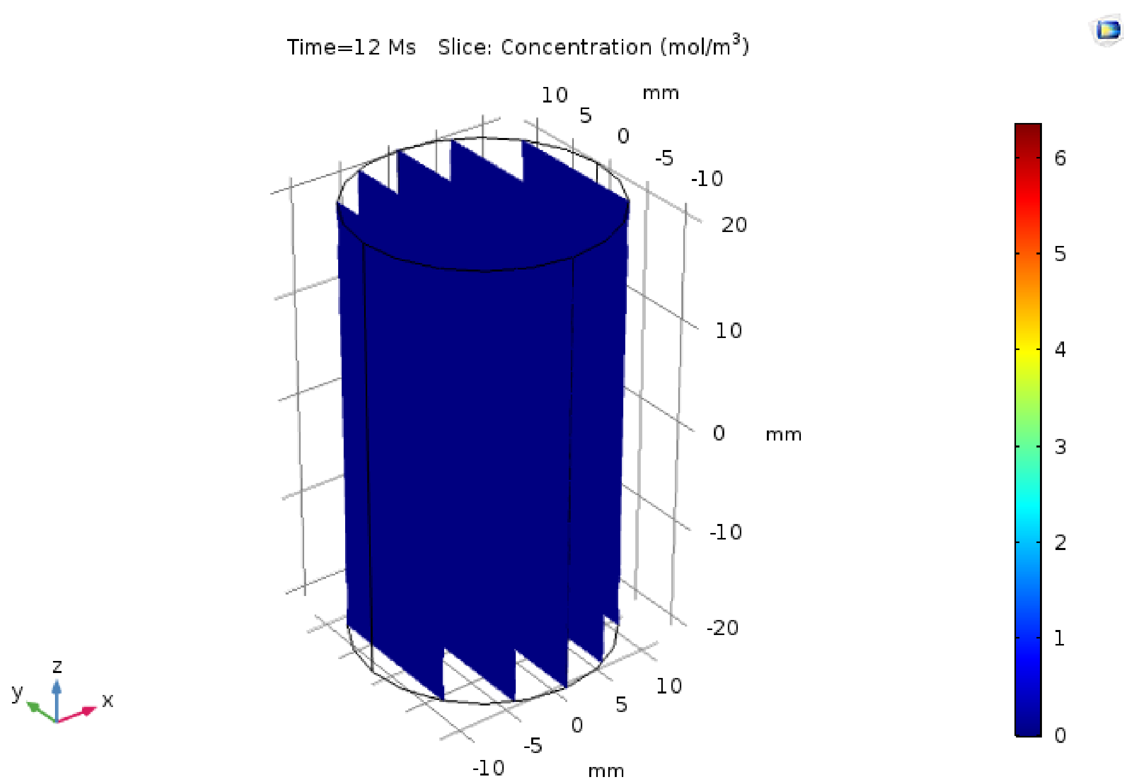

Figure 6.20 Oxygen gauge concentration at time= 12 Ms.

6.4 Shifting of Hole in Model.

While the results in the previous section strongly suggest that a key structure in the cork (namely the hole) affects the diffusion of gas through the cork stopper, additional tests were performed to confirm that conclusion. The cork used for this research, was is designated as cork 'A86' by the Amorim research team, contained a hole in the center as shown in Figure 6.21. To verify that the location of the hole affects the simulation results, the 3D diffusion map was modified by shifting the hole from the center to the top portion as shown in Figure 6.22. By locating the hole near the top of the cork, one would expect to observe a significant difference in the cumulated flux through the top and bottom of the cork.

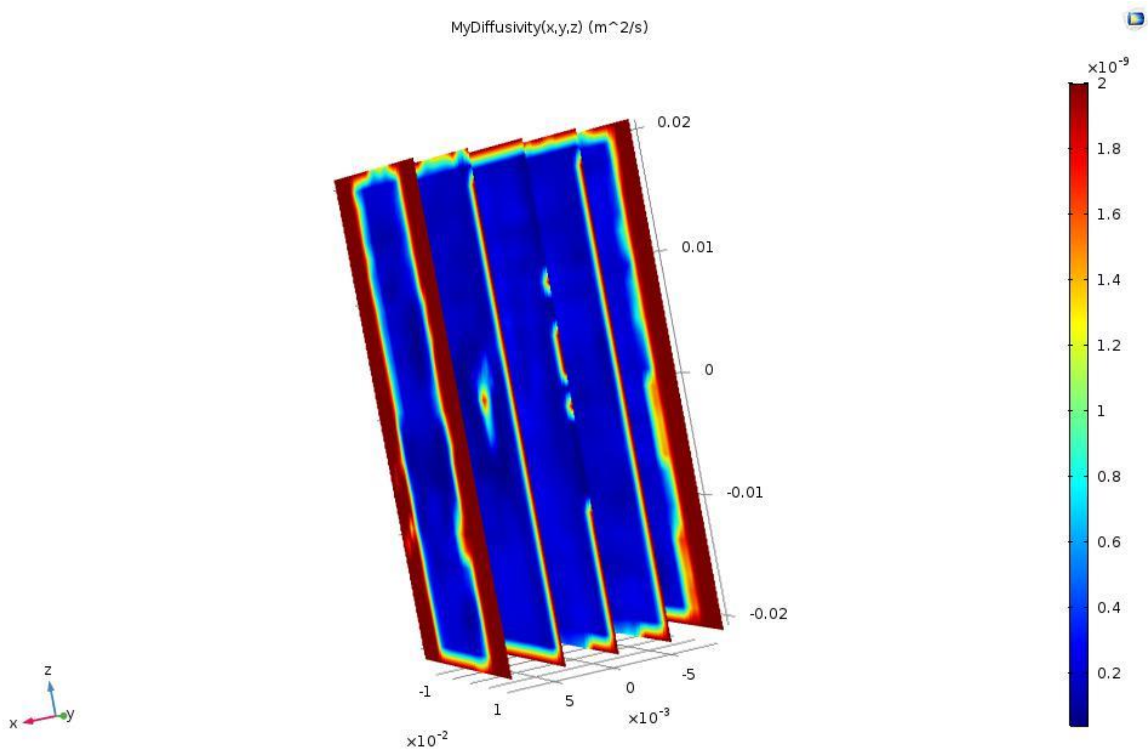

Figure 6.21 Hole in the center of the cork.

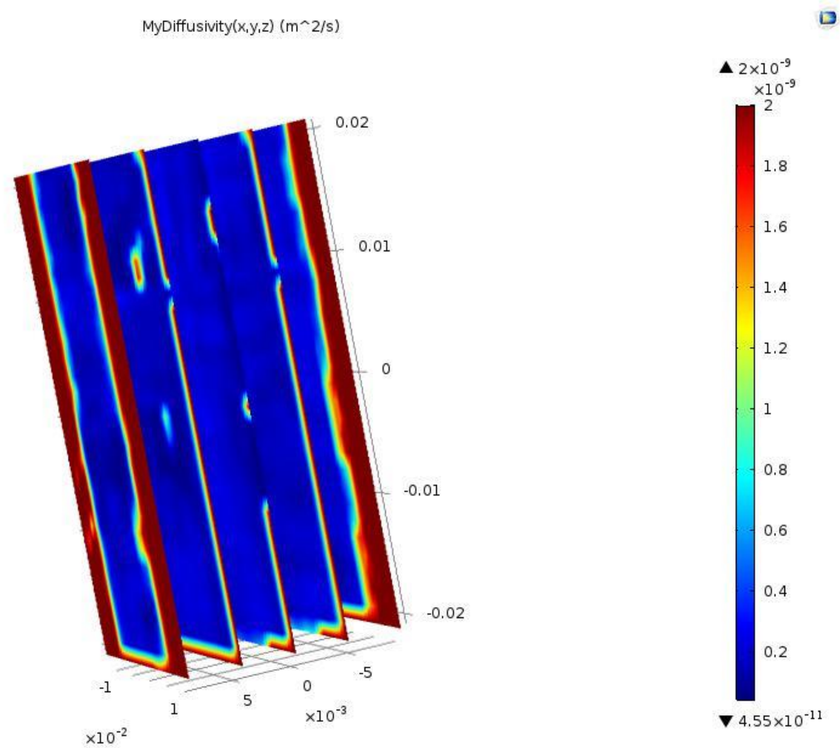

Figure 6.22 Hole shifted at the top portion.

The total flux at the top portion outlet increased as the diffusion coefficient increased at the top section of cork. A comparison of total flux at top outlet before and after shifting of the hole is compared in the graph in Figure 6.23.

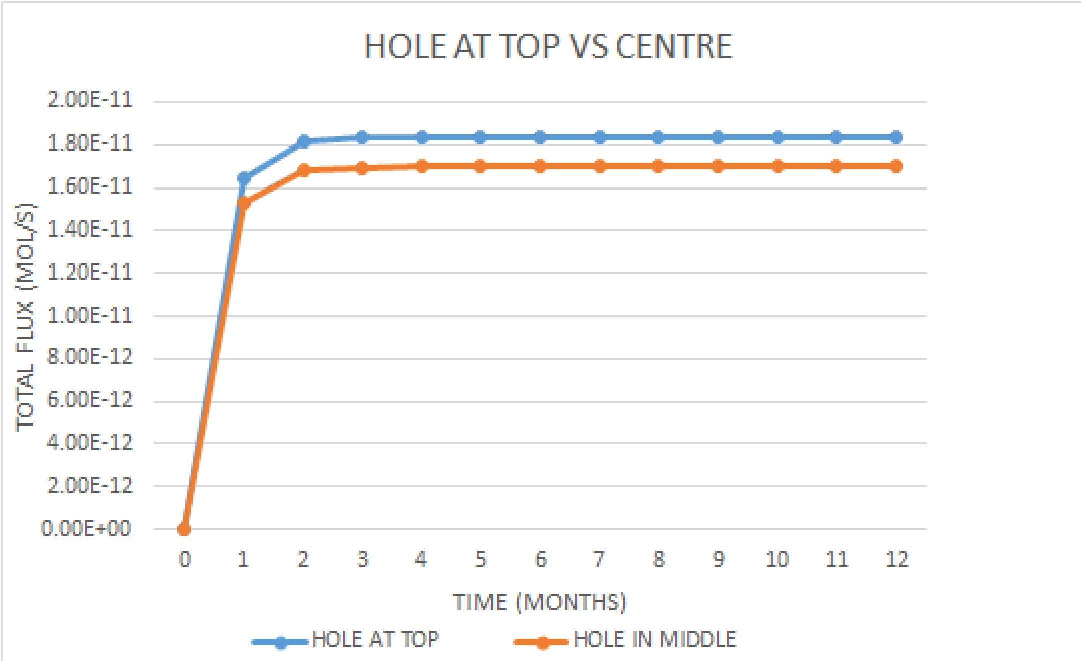

Figure 6.23 Comparison of total flux at the top outlet before and after hole shifting. The total oxygen flux through the top outlet increased. From this test it can concluded that the internal structure of cork affects the fluxes at the outlets.

6.5 Shuffled diffusion mapping import.

In this test, the spatial diffusion coefficient values obtained by mapping of the pixel value was randomly shuffled and imported in COMSOL interpolation function. While the average diffusion coefficient of the cork remained fixed, the connectivity of diffusion coefficient values was randomized. For example, in this test, there is no longer a large cohesive ‘hole’ in the cork. The high diffusion coefficient from the hole has now been randomly distributed throughout the volume of the cork. The plot of the diffusion coefficient mapping is shown in Figure 6.24.

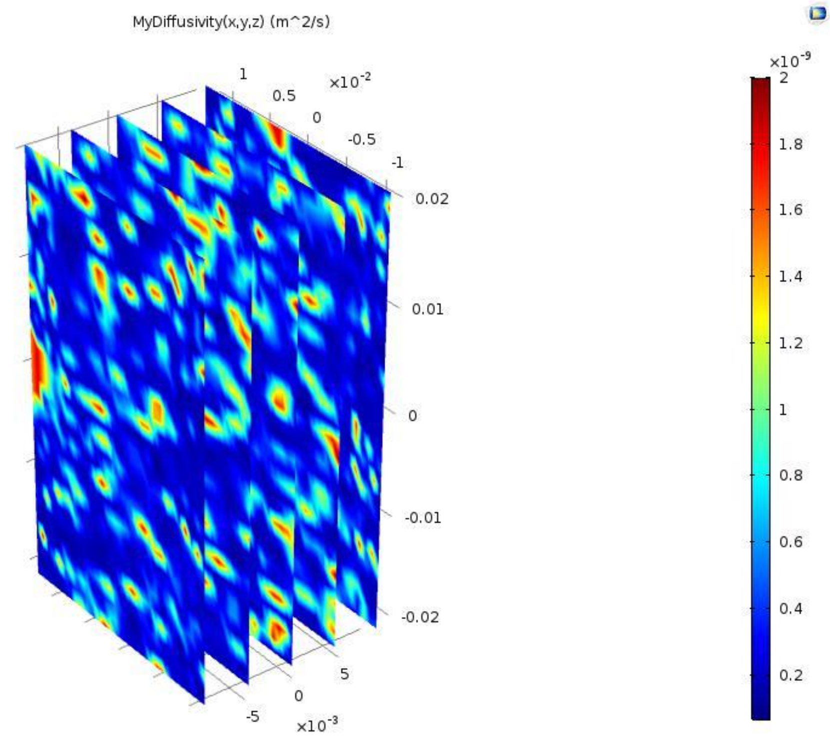

Figure 6.24 Shuffled diffusion coefficient values mapping in COMSOL.

As the internal structure of the model is changed due to the change in the diffusion mapping of the natural cork one would expect the total flux to also change at the bottom and top outlets of the cork. Comparison of total flux at the bottom outlet before and after shuffling the data is shown in Figure 6.24. Note the dramatic change in the predicted flux through the bottom of the cork. This test strongly suggests that the internal structure of the cork affects the diffusion of oxygen through the natural cork.

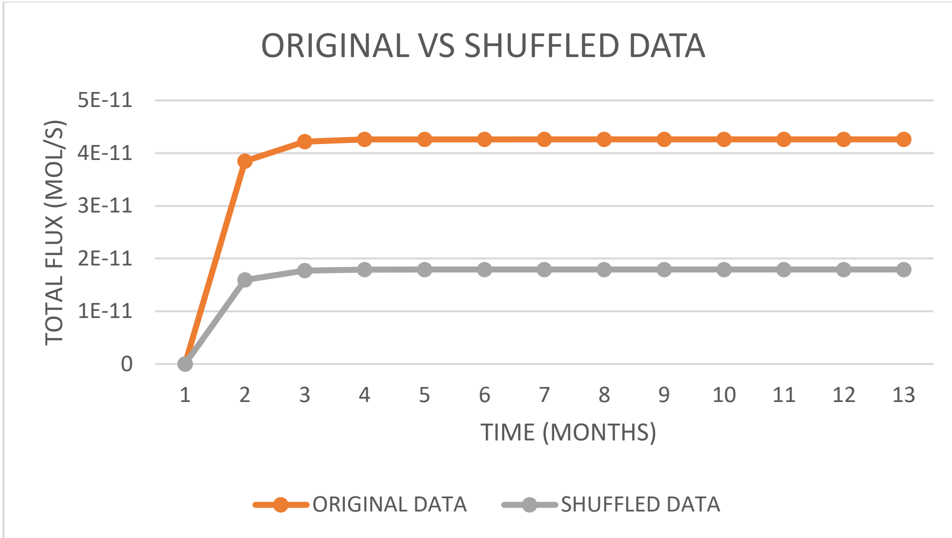

Figure 6.25 Original vs shuffled total flux at bottom.

CHAPTER 7

COMAPARISON OF SIMULATION AND EXPERIEMENTAL RESULTS

The total cumulative flux results were obtained from the experimental setup performed at the research department in Amorim & Irmãos, S.A. The experimental data was obtained in milliliters of oxygen flux through cork as shown in the table 7.1. Specifically, the data is for the cork designated as 'A86'.

Table 7.1 Experimental and simulation total flux

Time (Months)	Total flux (mL)	Experimental total flux (mol)	Simulation total flux (mol)
0	0	0	0
1	1.9	3.23E-11	3.85E-11
2	2.2	3.74E-11	4.22E-11
3	2.2	3.74E-11	4.26E-11
4	2.5	4.25E-11	4.26E-11
5	2.626965	4.46E-11	4.26E-11
6	2.6	4.42E-11	4.26E-11
7	2.583924	4.39E-11	4.26E-11
8	2.6	4.42E-11	4.26E-11
9	2.637339	4.48E-11	4.26E-11
10	2.7	4.59E-11	4.26E-11
11	2.79172	4.74E-11	4.26E-11
12	2.9	4.93E-11	4.26E-11

There are no free fitting parameters other than mapping of the CT intensity to a diffusion coefficient value. The mean diffusion coefficient was chosen such that the experimental and thermotical curves have the same time scale for increase. The shape of the experimental and simulation curves are similar as shown in Figure 7.1. The timescale of the increase in the total cumulated flux is well matched by the simulation, not only is the approximate time rise similar, but the saturated values for the long time are comparable.

Experimentally, the measured cork seems to have a continuous (but slow) increase in total flux at long times. The possibility of a continuous, but slow, increase in total flux in the experimental data could be explained by diffusion of oxygen from the outside atmosphere through the cork into the bottle. Initially, the high pressure in the cork relaxes after the cork is compressed and inserted into the bottle. On a shorter time-scale (about 6 months) the concentration of oxygen eventually becomes uniform as it equilibrates to atmospheric pressure. However, if one assumes that oxygen in the bottle will chemical react with the wine (oxidation of alcohol produces acetic acid), then as oxygen diffuses into the bottle, the concentration of oxygen drops inside the bottle due to chemical reactions. Therefore, there should be a second longer timescale during which the higher oxygen concentration in the atmosphere but zero concentration of oxygen inside the bottle drives diffusion of oxygen from the atmosphere through the cork and into the bottle. So, for long periods, one would expect to see flux of oxygen from the atmosphere into the bottle, which was not considered in this model designed in COMSOL.

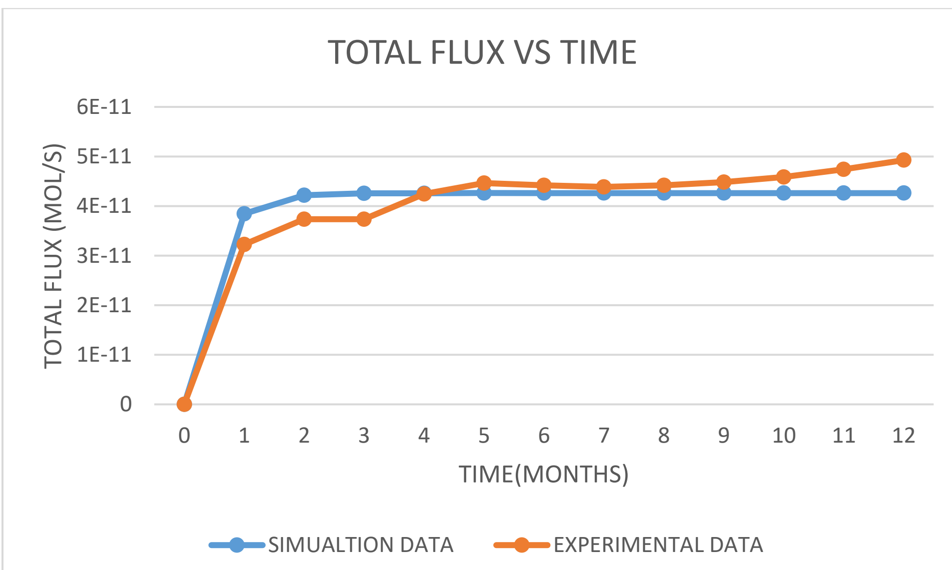

Figure 7.1 Experimental vs simulation total flux graph.

CHAPTER 8

CONCLUSION AND FUTURE WORK

From this research it can be concluded that the simulation model gives appropriate results up to certain timestep. Specifically, there is rough agreement between experimental and simulation results. Moreover, it is clear from the simulations and various tests that the internal structure of the cork affects the total cumulated flux through either the top or bottom of the cork.

While there is rough agreement between simulation and experiment for the specific cork under study, the employed model is very simple: The local diffusion coefficient is assumed to be inversely proportional to the density of material. Moreover, it assumed that the oxygen which flows through the cork and into the wine bottle comes from gas initially trapped in the cork cells.

But there are many other factors which have been ignored which might alter the flow of oxygen inside the bottle including the chemical reaction of oxygen with wine inside the bottle, a process necessary for the maturation of wine. Allowing the chemical reaction inside the bottle changes the boundary conditions inside the bottle: reducing the oxygen concentration inside the bottle below atmospheric concentrations will produce concentration gradient as the atmospheric oxygen concentration is higher than the cork and inside the bottle. This would lead to the flow of atmospheric oxygen flow into the bottle for longer time scales. These additional effects were not considered in this modelling of oxygen diffusion through the cork. By considering these additional factors into the simulation, a full model of oxygen flow inside the bottle could possibly be made.

Improvement in the resolution of the X-ray CT images will result in more accurate pixel value and better diffusion mapping of the internal structure of the natural cork. Decreasing in the spacing of tomography slices (less than 2 mm) would enable more accurate mapping of the CT images into a 3D diffusion coefficient map.

Lastly, the impact of cork internal structure on gas diffusion would be better understood by measuring and simulating a larger number of cork samples and comparing the simulation results to experimental measurements of the total flux. Use of powerful computer systems and finer mesh size gives accurate flux values at the outlets.

APPENDIX A

MATLAB code for diffusion co-efficient extraction from x-ray images.

```
% Code to calculate spatial values of Diffusion co-efficient
% Clear memory values
clc; clear variables;
% Loop for total number of images
for w=1:13
    % Read image files from the home directory
    F= imread (sprintf('slice%03d.bmp', w));
    % Rotate the image file in anti-clockwise direction.
    B=imrotate(F,2);
    % Create new variable to store original image
    C=B;
    % Remove zero rows form the matrix i.e. crop the upper & lower edges
    B (all (~B,2),) = [];
    % Remove zero columns from the matrix i.e. crop left & right edges
    B (: all(~B,1)) = [];
    % Extract number of rows and columns from the cropped image
    [row, col] =size(B);
    % Calibrate the pixel size by dividing the length of cork with number
    % of rows
    Calibration_Factor=0.045/row;
    % Mean of pixel values and ideal Diffusion Co-efficient.
    B_Avg=mean(mean(B)); D=zeros (row, col); D_Ideal=1.6E-11;
    % Loop to calculate Diffusion at each pixel value
    for i=1: row
        for j=1: col
            if B (i, j) ==0
                D (i, j) =2E-9;
            else
                % Diffusion co-efficient is Inversely proportional to the
                % pixel value.
                D (i, j)=D_Ideal*(B_Avg/double(B(i,j)));
            end
        end
    end
    Y=zeros(row,1); YY=zeros(row,col);X=zeros(1,col);XX=zeros(row,col);
    % Row values in Y direction
    for i=1:row
        Y(i,1)=Calibration_Factor*i;
    end
    % Y matrix formation
    for ii=1:col
        YY(:,ii)=Y(:,1);
```

```
    end
    % Column values in X direction
    for j=1:col
        X(1,j)=Calibration_Factor*j;
    end
    % X matrix formation
    for jj=1:row
        XX(jj,:)=X(1,:);
    end
    ZZ=ones(row,col)*(-0.012);
    old_data=[XX(:) YY(:) ZZ(:) D(:)];
    new_x=XX-0.2211;new_y=YY-0.2250;new_z=ZZ-0.2399;
    new_data=[new_x(:),new_y(:),new_z(:),D(:)];
    % Export data to a excel file with X,Y,Z,Diffusion co-efficient data
    xlsappend('TO_ANSYS.csv',new_data)
end

% Code from https://www.mathworks.com/matlabcentral/fileexchange/28600-xlsappend

function [success, message] = xlsappend(file,data,sheet)

% XLSAPPEND Stores numeric array or cell array to the end of specified Excel sheet.
%
%   REQUIRES ONLY ONE CALL TO THE EXCEL ACTXSERVER, so the overhead
is less
%   than for successive xlsread/xlswrite calls.
%
% [SUCCESS, MESSAGE] =XLSAPPEND (FILE, ARRAY, SHEET) writes ARRAY to
the Excel
%   workbook, FILE, into the area beginning at COLUMN A and FIRST UNUSED
%   ROW, in the worksheet specified in SHEET.
%   FILE and ARRAY must be specified. If either FILE or ARRAY is empty, an
%   error is thrown and XLSAPPEND terminates. The first worksheet of the
%   workbook is the default. If SHEET does not exist, a new sheet is added at
%   the end of the worksheet collection. If SHEET is an index larger than the
%   number of worksheets, new sheets are appended until the number of worksheets
%   in the workbook equals SHEET. The success of the operation is
%   returned in SUCCESS and any accompanying message, in MESSAGE. On error,
%   MESSAGE shall be a struct, containing the error message and message ID.
%   See NOTE 1.
%
%  [SUCCESS,MESSAGE]=XLSAPPEND(FILE,ARRAY) writes ARRAY to the Excel
%   workbook, FILE, in the first worksheet.
%
%   [SUCCESS,MESSAGE]=XLSAPPEND(FILE,ARRAY) writes ARRAY to the Excel
%   workbook, FILE, starting at cell A[firstUnusedRow] of the first
```

```
%   worksheet. The return values are as for the above example.
%
%   XLSAPPEND ARRAY FILE, is the command line version of the above example.
%
%   INPUT PARAMETERS:
%      file:   string defining the workbook file to write to.
%              Default directory is pwd; default extension 'xls'.
%      array:  m x n numeric array or cell array.
%      sheet:  string defining worksheet name;
%              double, defining worksheet index.
%
%   RETURN PARAMETERS:
%      SUCCESS: logical scalar.
%      MESSAGE: struct containing message field and message_id field.
%
%   EXAMPLES:
%
%   SUCCESS = XLSAPPEND('c:\matlab\work\myworkbook.xls',A) will write A to
%   the sheet 1 of workbook file, myworkbook.xls. On success, SUCCESS will
%   contain true, while on failure, SUCCESS will contain false.
%
%   NOTE 1: The above functionality depends upon Excel as a COM server. In
%   absence of Excel, ARRAY shall be written as a text file in CSV format. In
%   this mode, the SHEET argument shall be ignored.
%
% See also XLSREAD, XLSWRITE.
%

% Written by Brett Shoelson, PhD, 8/30/2010.
% Copyright 1984-2010 The MathWorks, Inc.
%=============================================================
================


% Set default values.
lastwarn('');
Sheet1 = 1;

if nargin < 3
   sheet = Sheet1;
end

if nargout > 0
   success = true;
   message = struct('message',{''},'identifier',{''});
end
```

```
% Handle input.
try
   % handle requested Excel workbook filename.
   if ~isempty(file)
      if ~ischar(file)
         error('MATLAB:xlswrite:InputClass','Filename must be a string.');
      end
      % check for wildcards in filename
      if any(strfind('*', file))
         error('MATLAB:xlswrite:FileName', 'Filename must not contain *.');
      end
      [Directory,file,ext]=fileparts(file);

      if isempty(ext) % add default Excel extension;
         ext = '.xls';
      end
      file = abspath(fullfile(Directory,[file ext]));
      [a1, a2] = fileattrib(file);
      if a1 && ~(a2.UserWrite == 1)
         error('MATLAB:xlswrite:FileReadOnly', 'File cannot be read-only.');
      end
   else % get workbook filename.
      error('MATLAB:xlswrite:EmptyFileName','Filename is empty.');
   end

   % Check for empty input data
   if isempty(data)
      error('MATLAB:xlswrite:EmptyInput','Input array is empty.');
   end

   % Check for N-D array input data
   if ~ismatrix(data)
      error('MATLAB:xlswrite:InputDimension',...
         'Dimension of input array cannot be higher than two.');
   end

   % Check class of input data
   if ~(iscell(data) || isnumeric(data) || ischar(data)) && ~islogical(data)
      error('MATLAB:xlswrite:InputClass',...
         'Input data must be a numeric, cell, or logical array.');
   end


   % convert input to cell array of data.
    if iscell(data)
```

```
        A=data;
      else
         A=num2cell(data);
      end

    if nargin > 2
        % Verify class of sheet parameter.
        if ~(ischar(sheet) || (isnumeric(sheet) && sheet > 0))
            error('MATLAB:xlswrite:InputClass',...
                'Sheet argument must be a string or a whole number greater than 0.');
        end
        if isempty(sheet)
            sheet = Sheet1;
        end
        % Parse sheet
        if ischar(sheet) && contains(sheet,':')
            sheet = Sheet1;% Use default sheet.
        end
    end

catch exception
    if ~isempty(narginchk(2,4))
        error('MATLAB:xlswrite:InputArguments',nargchk(2,4,nargin));
    else
        success = false;
        message = exceptionHandler(nargout, exception);
    end
    return;
end
%---------------------------------------------------------------------------
% Attempt to start Excel as ActiveX server.
try
    Excel = actxserver('Excel.Application');
    % open workbook
    Excel.DisplayAlerts = 0;

    %Workaround for G313142.  For certain files, unless a workbook is
    %opened prior to opening the file, various COM calls return an error:
    %0x800a9c64.  The line below works around this flaw.  Since we have
    %seen only one example of such a file, we have decided not to incur the
    %time penalty involved here.
    %     aTemp = Excel.workbooks.Add(); aTemp.Close();

    ExcelWorkbook = Excel.workbooks.Open(file);
    %ExcelWorkbook.ReadOnly
    format = ExcelWorkbook.FileFormat;
```

```
    if  strcmpi(format, 'xlCurrentPlatformText') == 1
        error('MATLAB:xlsread:FileFormat', 'File %s not in Microsoft Excel Format.', file);
    end

    %Sheets = Excel.ActiveWorkBook.Sheets;
    activate_sheet(Excel,sheet);
    readinfo = get(Excel.Activesheet,'UsedRange');
    %get(Excel.Activesheet,'Name')
    if numel(readinfo.value)== 1 && isnan(readinfo.value)
        m2 = 0;
    else
        [m2,~] = size(readinfo.value);
    end

catch exception %#ok<NASGU>
    warning('MATLAB:xlswrite:NoCOMServer',...
        ['Could not start Excel server for export.\n' ...
        'XLSWRITE will attempt to write file in CSV format.']);
    if nargout > 0
        [message.message,message.identifier] = lastwarn;
    end
    % write data as CSV file, that is, comma delimited.
    file = regexprep(file,'(\.xls[^.]*+)$','.csv');
    try
        dlmwrite(file,data,','); % write data.
    catch exception
        exceptionNew = MException('MATLAB:xlswrite:dlmwrite', 'An error occurred on
data export in CSV format.');
        exceptionNew = exceptionNew.addCause(exception);
        if nargout == 0
            % Throw error.
            throw(exceptionNew);
        else
            success = false;
            message.message = exceptionNew.getReport;
            message.identifier = exceptionNew.identifier;
        end
    end
    return;
end
%--------------------------------------------------------------------------------
try
    % Construct range string
    % Range was partly specified or not at all. Calculate range.
    % Select range of occupied cells in active sheet.
    % Activate indicated worksheet.
```

```
    message = activate_sheet(Excel,sheet);
    [m,n] = size(A);
    range = calcrange('',m,n,m2);
catch exception
    success = false;
    message = exceptionHandler(nargout, exception);
    return;
end

%--------------------------------------------------------------------------
try
    bCreated = ~exist(file,'file');
    ExecuteWrite;
catch exception
    if (bCreated && exist(file, 'file') == 2)
        delete(file);
    end
    success = false;
    message = exceptionHandler(nargout, exception);
end
    function ExecuteWrite
            cleanUp = onCleanup(@()cleaner(Excel, file));
            if bCreated
                % Create new workbook.
                %This is in place because in the presence of a Google Desktop
                %Search installation, calling Add, and then SaveAs after adding data,
                %to create a new Excel file, will leave an Excel process hanging.
                %This workaround prevents it from happening, by creating a blank file,
                %and saving it.  It can then be opened with Open.

                %ExcelWorkbook = Excel.workbooks.Add;
                switch ext
                    case '.xls' %xlExcel8 or xlWorkbookNormal
                       xlFormat = -4143;
                    case '.xlsb' %xlExcel12
                       xlFormat = 50;
                    case '.xlsx' %xlOpenXMLWorkbook
                       xlFormat = 51;
                    case '.xlsm' %xlOpenXMLWorkbookMacroEnabled
                       xlFormat = 52;
                    otherwise
                       xlFormat = -4143;
                end
                ExcelWorkbook.SaveAs(file, xlFormat);
                ExcelWorkbook.Close(false);
            end
```

```
        %Open file
        %ExcelWorkbook = Excel.workbooks.Open(file);
        if ExcelWorkbook.ReadOnly ~= 0
            %This means the file is probably open in another process.
            error('MATLAB:xlswrite:LockedFile', 'The file %s is not writable.  It may be
locked by another process.', file);
        end
        try % select region.
            % Select range in worksheet.
            Select(Range(Excel,sprintf('%s',range)));

        catch exceptionInner % Throw data range error.
            throw(MException('MATLAB:xlswrite:SelectDataRange', sprintf('Excel
returned: %s.', exceptionInner.message)));
        end

        % Export data to selected region.
        set(Excel.selection,'Value',A);
        ExcelWorkbook.Save
    end
end
function cleaner(Excel, filePath)
    try
        %Turn off dialog boxes as we close the file and quit Excel.
        Excel.DisplayAlerts = 0;
        %Explicitly close the file just in case.  The Excel API expects
        %just the filename and not the path.  This is safe because Excel
        %also does not allow opening two files with the same name in
        %different folders at the same time.
        [~, n, e] = fileparts(filePath);
        fileName = [n e];
        Excel.Workbooks.Item(fileName).Close(false);
    catch exception %#ok<NASGU>
        %If something fails in closing, there is nothing to do but attempt
        %to quit.
    end
    Excel.Quit;
end
%-------------------------------------------------------------------------
function message = activate_sheet(Excel,Sheet)
% Activate specified worksheet in workbook.

% Initialise worksheet object
WorkSheets = Excel.sheets;
message = struct('message',{''},'identifier',{''});
```

```
% Get name of specified worksheet from workbook
try
   TargetSheet = get(WorkSheets,'item',Sheet);
catch exception  %#ok<NASGU>
   % Worksheet does not exist. Add worksheet.
   TargetSheet = addsheet(WorkSheets,Sheet);
   warning('MATLAB:xlswrite:AddSheet','Added specified worksheet.');
   if nargout > 0
      [message.message,message.identifier] = lastwarn;
   end
end

% activate worksheet
Activate(TargetSheet);
end
%-------------------------------------------------------------------------------
function newsheet = addsheet(WorkSheets,Sheet)
% Add new worksheet, Sheet into worsheet collection, WorkSheets.

if isnumeric(Sheet)
   % iteratively add worksheet by index until number of sheets == Sheet.
   while WorkSheets.Count < Sheet
      % find last sheet in worksheet collection
      lastsheet = WorkSheets.Item(WorkSheets.Count);
      newsheet = WorkSheets.Add([],lastsheet);
   end
else
   % add worksheet by name.
   % find last sheet in worksheet collection
   lastsheet = WorkSheets.Item(WorkSheets.Count);
   newsheet = WorkSheets.Add([],lastsheet);
end
% If Sheet is a string, rename new sheet to this string.
if ischar(Sheet)
   set(newsheet,'Name',Sheet);
end
end
%-------------------------------------------------------------------------------
function [absolutepath]=abspath(partialpath)

% parse partial path into path parts
[pathname, filename, ext] = fileparts(partialpath);
% no path qualification is present in partial path; assume parent is pwd, except
% when path string starts with '~' or is identical to '~'.
if isempty(pathname) && isempty(strmatch('~',partialpath))
```

```
    Directory = pwd;
elseif isempty(regexp(partialpath,'(.:|\\\\)','once')) && ...
        isempty(strmatch('/',partialpath)) && ...
        isempty(strmatch('~',partialpath))
    % path did not start with any of drive name, UNC path or '~'.
    Directory = [pwd,filesep,pathname];
else
    % path content present in partial path; assume relative to current directory,
    % or absolute.
    Directory = pathname;
end

% construct absulute filename
absolutepath = fullfile(Directory,[filename,ext]);
end
%--------------------------------------------------------------------------
function range = calcrange(range,m,n,offset)
% Calculate full target range, in Excel A1 notation, to include array of size
% m x n

range = upper(range);
cols = isletter(range);
rows = ~cols;
% Construct first row.
if ~any(rows)
    firstrow = offset+1; % Default row.
else
    firstrow = str2double(range(rows)); % from range input.
end
% Construct first column.
if ~any(cols)
    firstcol = 'A'; % Default column.
else
    firstcol = range(cols); % from range input.
end
try
    lastrow = num2str(firstrow+m-1);   % Construct last row as a string.
    firstrow = num2str(firstrow);      % Convert first row to string image.
    lastcol = dec2base27(base27dec(firstcol)+n-1); % Construct last column.

    range = [firstcol firstrow ':' lastcol lastrow]; % Final range string.
catch exception
    error('MATLAB:xlswrite:CalculateRange', 'Invalid data range: %s.', range);
end
end
%-------------------------------------------------------------------
```

```
function string = index_to_string(index, first_in_range, digits)

letters = 'A':'Z';
working_index = index - first_in_range;
outputs = cell(1,digits);
[outputs{1:digits}] = ind2sub(repmat(26,1,digits), working_index);
string = fliplr(letters([outputs{:}]));
end
%-------------------------------------------------------------------
function [digits, first_in_range] = calculate_range(num_to_convert)

digits = 1;
first_in_range = 0;
current_sum = 26;
while num_to_convert > current_sum
    digits = digits + 1;
    first_in_range = current_sum;
    current_sum = first_in_range + 26.^digits;
end
end
%--------------------------------------------------------------------------
function s = dec2base27(d)

%   DEC2BASE27(D) returns the representation of D as a string in base 27,
%   expressed as 'A'..'Z', 'AA','AB'...'AZ', and so on. Note, there is no zero
%   digit, so strictly we have hybrid base26, base27 number system.  D must be a
%   negative integer bigger than 0 and smaller than 2^52.
%
%   Examples
%       dec2base(1) returns 'A'
%       dec2base(26) returns 'Z'
%       dec2base(27) returns 'AA'
%-------------------------------------------------------------------------

d = d(:);
if d ~= floor(d) || any(d(:) < 0) || any(d(:) > 1/eps)
    error('MATLAB:xlswrite:Dec2BaseInput',...
        'D must be an integer, 0 <= D <= 2^52.');
end
[num_digits, begin] = calculate_range(d);
s = index_to_string(d, begin, num_digits);
end
%--------------------------------------------------------------------------
function d = base27dec(s)
%   BASE27DEC(S) returns the decimal of string S which represents a number in
%   base 27, expressed as 'A'..'Z', 'AA','AB'...'AZ', and so on. Note, there is
```

```
%   no zero so strictly we have hybrid base26, base27 number system.
%
%   Examples
%       base27dec('A') returns 1
%       base27dec('Z') returns 26
%       base27dec('IV') returns 256
%-----------------------------------------------------------------------------

if length(s) == 1
  d = s(1) -'A' + 1;
else
   cumulative = 0;
   for i = 1:numel(s)-1
      cumulative = cumulative + 26.^i;
   end
   indexes_fliped = 1 + s - 'A';
   indexes = fliplr(indexes_fliped);
   indexes_in_cells = num2cell(indexes, 1, ones(1,numel(indexes)));
   d = cumulative + sub2ind(repmat(26, 1,numel(s)), indexes_in_cells{:});
end
end
%-------------------------------------------------------------------------------

function messageStruct = exceptionHandler(nArgs, exception)
   if nArgs == 0
      throwAsCaller(exception);
   else
      messageStruct.message = exception.message;
      messageStruct.identifier = exception.identifier;
   end
end
```

APPENDIX B

MATLAB code for getting the corner points in x-ray images.

```
% Code to extract the corner points of the cork in images

% Clear all memory values

clear variables; clc; REQUIRED_VALUES=zeros(2,13);

for w=1:13

    % Read the image files

    F= imread(sprintf('slice%03d.bmp',w));

    % Rotate the image files

    B=imrotate(F,2);C=B;

    % Remove zero rows form the matrix i.e. crop the upper & lower edges

    B( all(~B,2), : ) = [];

    % Remove zero columns from the matrix i.e. crop left & right edges

    B( :, all(~B,1) ) = [];

    % Extract number of rows and column in the cropped matrix

    [row,col]=size(B);

    % Calibrate the pixel size by dividing the length of cork with number

    Calibration_Factor=0.045/row;

    % Extracted the wanted rows and columns

    [wanted_rows,wanted_cols]=find(C~=0);

    % Position of the corner points in the image

    RU= min(wanted_rows);RB= max(wanted_rows);

    CL= min(wanted_cols);CR= max(wanted_cols);
```

```
    CORNERS=[RU CL;RU CR;RB CL;RB CR];
    % Convert corner points to the physical co-ordinates
    REAL_CORNERS=Calibration_Factor.*CORNERS;
    % Required upper corner values of the images
    REQUIRED_VALUES(:,w)=[REAL_CORNERS(1,2);REAL_CORNERS(2,2)];
end
out=reshape(REQUIRED_VALUES,[],1);
% Extract the corner value data to a txt file
dlmwrite('out.txt',out);
```

REFERENCES

Sonia Lequin, David Chassagne, Thomas Karbowaik, Jean-Marc Simon, Christian Paulin, & Jean-Pierre Bellat (2012). Diffusion of Oxygen in Cork. *Journal of Agriculture and Food Chemistry,* 2012, 60, 3348–3356.

David P. Faria, Ana L. Fonseca, Helen Pereira, & Orlando M. N. D. Teodoro (2011). Permeability of cork. *Journal of Agriculture and Food Chemistry,* 2011, 59, 3590–3597.

Paulo Lopes, Cedric Saucier, Pierre-Louis Teissedre, & Yves Glories (2007). Main routes of Oxygen ingress through different closures into Wine bottles. *Journal of Agriculture and Food Chemistry,* 2007, 62, 351–33405.

ANSYS CFX Documentation 17.0 Chapter 1, *https://www.sharcnet.ca/Software/Ansys/17.0/en-us/help/cfx_intr/i1301903.html*

ANSYS CFX Documentation 17.0 Chapter 1.2, *https://www.sharcnet.ca/Software/Ansys/17.0/en-us/help/cfx_intr/i1301908.html*

Diffusion through a membrane, *http://mathbench.umd.edu/modules/cell-processes_diffmembrane/page05.htm conversion p to D*

ANSYS CFX Documentation 17.0 Chapter 1.2, *https://www.comsol.com/multiphysics/navier-stokes-equations*

ANSYS CFX Documentation 17.0 Chapter 1.2, *https://www.sharcnet.ca/Software/Ansys/17.0/en-us/help/cfx_intr/i1301908.html*

Santos Pereira, J., Bugalho, M. N., and Caldeira, M. D. (2008). From the cork oak to cork: A sustainable ecosystem *APCOR: Portuguese cork Association.*

Helena Pereira, Elsevier (New York),2007. Cork: Biology, Production and Uses